Atmic Networks

CERTIFICATION EXAM SUPPORT

EXAM PREPARATION MATERIAL

Practice files designed to help you understand exam patterns, question styles and the overall assessment format.

IMPORTANT NOTE

> **Please note that these files may not be up to date. However, the questions will help you understand the exam format and typical question patterns.**

Stay connected with our support team
for the latest file updates and exam-related information.

www.atmicnetworks.com

Knowledge that moves beyond memorization.

www.atmicnetworks.com

Question: 1

A data engineer is configuring an AWS Glue job to read data from an Amazon S3 bucket. The data engineer has set up the necessary AWS Glue connection details and an associated IAM role. However, when the data engineer attempts to run the AWS Glue job, the data engineer receives an error message that indicates that there are problems with the Amazon S3 VPC gateway endpoint. The data engineer must resolve the error and connect the AWS Glue job to the S3 bucket.
Which solution will meet this requirement?

A. Update the AWS Glue security group to allow inbound traffic from the Amazon S3 VPC gateway endpoint.
B. Configure an S3 bucket policy to explicitly grant the AWS Glue job permissions to access the S3 bucket.
C. Review the AWS Glue job code to ensure that the AWS Glue connection details include a fully qualified domain name.
D. Verify that the VPC's route table includes inbound and outbound routes for the Amazon S3 VPC gateway endpoint.

Answer: D

Explanation:
The error message indicates that the AWS Glue job cannot access the Amazon S3 bucket through the VPC endpoint. This could be because the VPC's route table does not have the necessary routes to direct the traffic to the endpoint. To fix this, the data engineer must verify that the route table has an entry for the Amazon S3 service prefix (com.amazonaws.region.s3) with the target as the VPC endpoint ID. This will allow the AWS Glue job to use the VPC endpoint to access the S3 bucket without going through the internet or a NAT gateway. For more information, see Gateway endpoints. Reference:
Troubleshoot the AWS Glue error "VPC S3 endpoint validation failed"
Amazon VPC endpoints for Amazon S3
[AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

Question: 2

A retail company has a customer data hub in an Amazon S3 bucket. Employees from many countries use the data hub to support company-wide analytics. A governance team must ensure that the company's data analysts can access data only for customers who are within the same country as the analysts.
Which solution will meet these requirements with the LEAST operational effort?

A. Create a separate table for each country's customer data. Provide access to each analyst based on the country that the analyst serves.
B. Register the S3 bucket as a data lake location in AWS Lake Formation. Use the Lake Formation rowlevel security features to enforce the company's access policies.
C. Move the data to AWS Regions that are close to the countries where the customers are. Provide access to each analyst based on the country that the analyst serves.
D. Load the data into Amazon Redshift. Create a view for each country. Create separate 1AM roles for each country to provide access to data from each country. Assign the appropriate roles to the analysts.

Answer: B

Explanation:
AWS Lake Formation is a service that allows you to easily set up, secure, and manage data lakes. One of the features of Lake Formation is row-level security, which enables you to control access to specific rows or columns of data based on the identity or role of the user. This feature is useful for scenarios where you need to restrict access to sensitive or regulated data, such as customer data from different countries. By registering the S3 bucket as a data lake location in Lake Formation, you can use the Lake Formation console or APIs to define and apply row-level security policies to the data in the bucket. You can also use Lake Formation blueprints to automate the ingestion and transformation of data from various sources into the data lake. This solution requires the least operational effort compared to the other options, as it does not involve creating or moving data, or managing multiple tables, views, or roles. Reference:
AWS Lake Formation
Row-Level Security
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Lakes and Data Warehouses, Section 4.2: AWS Lake Formation

Question: 3

A media company wants to improve a system that recommends media content to customer based on user behavior and preferences. To improve the recommendation system, the company needs to incorporate insights from third-party datasets into the company's existing analytics platform. The company wants to minimize the effort and time required to incorporate third-party datasets. Which solution will meet these requirements with the LEAST operational overhead?

A. Use API calls to access and integrate third-party datasets from AWS Data Exchange.
B. Use API calls to access and integrate third-party datasets from AWS
C. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from AWS CodeCommit repositories.
D. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from Amazon Elastic Container Registry (Amazon ECR).

Answer: A

Explanation:
AWS Data Exchange is a service that makes it easy to find, subscribe to, and use third-party data in the cloud. It provides a secure and reliable way to access and integrate data from various sources, such as data providers, public datasets, or AWS services. Using AWS Data Exchange, you can browse and subscribe to data products that suit your needs, and then use API calls or the AWS Management Console to export the data to Amazon S3, where you can use it with your existing analytics platform. This solution minimizes the effort and time required to incorporate third-party datasets, as you do not need to set up and manage data pipelines, storage, or access controls. You also benefit from the data quality and freshness provided by the data providers, who can update their data products as frequently as needed12.
The other options are not optimal for the following reasons:
B . Use API calls to access and integrate third-party datasets from AWS. This option is vague and does not specify which AWS service or feature is used to access and integrate third-party datasets. AWS offers a variety of services and features that can help with data ingestion, processing, and analysis, but not all of them are suitable for the given scenario. For example, AWS Glue is a serverless data integration service that can help you discover, prepare, and combine data from various sources, but it requires you to create and run data extraction, transformation, and

loading (ETL) jobs, which can add operational overhead3.

C . Use Amazon Kinesis Data Streams to access and integrate third-party datasets from AWS CodeCommit repositories. This option is not feasible, as AWS CodeCommit is a source control service that hosts secure Git-based repositories, not a data source that can be accessed by Amazon Kinesis Data Streams. Amazon Kinesis Data Streams is a service that enables you to capture, process, and analyze data streams in real time, such as clickstream data, application logs, or IoT telemetry. It does not support accessing and integrating data from AWS CodeCommit repositories, which are meant for storing and managing code, not data .

D . Use Amazon Kinesis Data Streams to access and integrate third-party datasets from Amazon Elastic Container Registry (Amazon ECR). This option is also not feasible, as Amazon ECR is a fully managed container registry service that stores, manages, and deploys container images, not a data source that can be accessed by Amazon Kinesis Data Streams. Amazon Kinesis Data Streams does not support accessing and integrating data from Amazon ECR, which is meant for storing and managing container images, not data .

Reference:

1: AWS Data Exchange User Guide

2: AWS Data Exchange FAQs

3: AWS Glue Developer Guide

: AWS CodeCommit User Guide

: Amazon Kinesis Data Streams Developer Guide

: Amazon Elastic Container Registry User Guide

: Build a Continuous Delivery Pipeline for Your Container Images with Amazon ECR as Source

---

Question: 4

A financial company wants to implement a data mesh. The data mesh must support centralized data governance, data analysis, and data access control. The company has decided to use AWS Glue for data catalogs and extract, transform, and load (ETL) operations.

Which combination of AWS services will implement a data mesh? (Choose two.)

A. Use Amazon Aurora for data storage. Use an Amazon Redshift provisioned cluster for data analysis.

B. Use Amazon S3 for data storage. Use Amazon Athena for data analysis.

C. Use AWS Glue DataBrewfor centralized data governance and access control.

D. Use Amazon RDS for data storage. Use Amazon EMR for data analysis.

E. Use AWS Lake Formation for centralized data governance and access control.

Answer: B E

Explanation:

A data mesh is an architectural framework that organizes data into domains and treats data as products that are owned and offered for consumption by different teamsl. A data mesh requires a centralized layer for data governance and access control, as well as a distributed layer for data storage and analysis. AWS Glue can provide data catalogs and ETL operations for the data mesh, but it cannot provide data governance and access control by itself2. Therefore, the company needs to use another AWS service for this purpose. AWS Lake Formation is a service that allows you to create, secure, and manage data lakes on AWS3. It integrates with AWS Glue and other AWS services to provide centralized data governance and access control for the data mesh. Therefore, option E is correct.

For data storage and analysis, the company can choose from different AWS services depending on their needs and preferences. However, one of the benefits of a data mesh is that it enables data to be stored and processed in a decoupled and scalable way1. Therefore, using serverless or managed services that can handle large volumes and varieties of data is preferable. Amazon S3 is a highly scalable, durable, and secure object storage service that can

store any type of data. Amazon Athena is a serverless interactive query service that can analyze data in Amazon S3 using standard SQL. Therefore, option B is a good choice for data storage and analysis in a data mesh. Option A, C, and D are not optimal because they either use relational databases that are not suitable for storing diverse and unstructured data, or they require more management and provisioning than serverless services. Reference:

1: What is a Data Mesh? - Data Mesh Architecture Explained - AWS

2: AWS Glue - Developer Guide

3: AWS Lake Formation - Features

[4] : Design a data mesh architecture using AWS Lake Formation and AWS Glue

[5] : Amazon S3 - Features

[6] : Amazon Athena - Features

Question: 5

A data engineer maintains custom Python scripts that perform a data formatting process that many AWS Lambda functions use. When the data engineer needs to modify the Python scripts, the data engineer must manually update all the Lambda functions.

The data engineer requires a less manual way to update the Lambda functions.

Which solution will meet this requirement?

A. Store a pointer to the custom Python scripts in the execution context object in a shared Amazon S3 bucket.

B. Package the custom Python scripts into Lambda layers. Apply the Lambda layers to the Lambda functions.

C. Store a pointer to the custom Python scripts in environment variables in a shared Amazon S3 bucket.

D. Assign the same alias to each Lambda function. Call reach Lambda function by specifying the function's alias.

Answer: B

Explanation:

Lambda layers are a way to share code and dependencies across multiple Lambda functions. By packaging the custom Python scripts into Lambda layers, the data engineer can update the scripts in one place and have them automatically applied to all the Lambda functions that use the layer. This reduces the manual effort and ensures consistency across the Lambda functions. The other options are either not feasible or not efficient. Storing a pointer to the custom Python scripts in the execution context object or in environment variables would require the Lambda functions to download the scripts from Amazon S3 every time they are invoked, which would increase latency and cost. Assigning the same alias to each Lambda function would not help with updating the Python scripts, as the alias only points to a specific version of the Lambda function code. Reference:

AWS Lambda layers

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 3: Data Ingestion and Transformation, Section 3.4: AWS Lambda

Question: 6

A company created an extract, transform, and load (ETL) data pipeline in AWS Glue. A data engineer must crawl a table that is in Microsoft SQL Server. The data engineer needs to extract, transform, and load the output of the crawl to an Amazon S3 bucket. The data engineer also must orchestrate the data pipeline.

Which AWS service or feature will meet these requirements MOST cost-effectively?

A. AWS Step Functions
B. AWS Glue workflows
C. AWS Glue Studio
D. Amazon Managed Workflows for Apache Airflow (Amazon MWAA)

Answer: B

Explanation:
AWS Glue workflows are a cost-effective way to orchestrate complex ETL jobs that involve multiple crawlers, jobs, and triggers. AWS Glue workflows allow you to visually monitor the progress and dependencies of your ETL tasks, and automatically handle errors and retries. AWS Glue workflows also integrate with other AWS services, such as Amazon S3, Amazon Redshift, and AWS Lambda, among others, enabling you to leverage these services for your data processing workflows. AWS Glue workflows are serverless, meaning you only pay for the resources you use, and you don't have to manage any infrastructure.
AWS Step Functions, AWS Glue Studio, and Amazon MWAA are also possible options for orchestrating ETL pipelines, but they have some drawbacks compared to AWS Glue workflows. AWS Step Functions is a serverless function orchestrator that can handle different types of data processing, such as real-time, batch, and stream processing. However, AWS Step Functions requires you to write code to define your state machines, which can be complex and error-prone. AWS Step Functions also charges you for every state transition, which can add up quickly for large-scale ETL pipelines.
AWS Glue Studio is a graphical interface that allows you to create and run AWS Glue ETL jobs without writing code. AWS Glue Studio simplifies the process of building, debugging, and monitoring your ETL jobs, and provides a range of pre-built transformations and connectors. However, AWS Glue Studio does not support workflows, meaning you cannot orchestrate multiple ETL jobs or crawlers with dependencies and triggers. AWS Glue Studio also does not support streaming data sources or targets, which limits its use cases for real-time data processing.
Amazon MWAA is a fully managed service that makes it easy to run open-source versions of Apache Airflow on AWS and build workflows to run your ETL jobs and data pipelines. Amazon MWAA provides a familiar and flexible environment for data engineers who are familiar with Apache Airflow, and integrates with a range of AWS services such as Amazon EMR, AWS Glue, and AWS Step Functions. However, Amazon MWAA is not serverless, meaning you have to provision and pay for the resources you need, regardless of your usage. Amazon MWAA also requires you to write code to define your DAGs, which can be challenging and time-consuming for complex ETL pipelines.
Reference:
AWS Glue Workflows
AWS Step Functions
AWS Glue Studio
Amazon MWAA
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

Question: 7

A financial services company stores financial data in Amazon Redshift. A data engineer wants to run real-time queries on the financial data to support a web-based trading application. The data engineer wants to run the queries from within the trading application.
Which solution will meet these requirements with the LEAST operational overhead?

A. Establish WebSocket connections to Amazon Redshift.
B. Use the Amazon Redshift Data API.

C. Set up Java Database Connectivity (JDBC) connections to Amazon Redshift.

D. Store frequently accessed data in Amazon S3. Use Amazon S3 Select to run the queries.

Answer: B

Explanation:

The Amazon Redshift Data API is a built-in feature that allows you to run SQL queries on Amazon Redshift data with web services-based applications, such as AWS Lambda, Amazon SageMaker notebooks, and AWS Cloud9. The Data API does not require a persistent connection to your database, and it provides a secure HTTP endpoint and integration with AWS SDKs. You can use the endpoint to run SQL statements without managing connections. The Data API also supports both Amazon Redshift provisioned clusters and Redshift Serverless workgroups. The Data API is the best solution for running real-time queries on the financial data from within the trading application, as it has the least operational overhead compared to the other options.

Option A is not the best solution, as establishing WebSocket connections to Amazon Redshift would require more configuration and maintenance than using the Data API. WebSocket connections are also not supported by Amazon Redshift clusters or serverless workgroups.

Option C is not the best solution, as setting up JDBC connections to Amazon Redshift would also require more configuration and maintenance than using the Data API. JDBC connections are also not supported by Redshift Serverless workgroups.

Option D is not the best solution, as storing frequently accessed data in Amazon S3 and using Amazon S3 Select to run the queries would introduce additional latency and complexity than using the Data API. Amazon S3 Select is also not optimized for real-time queries, as it scans the entire object before returning the results.

Reference:

Using the Amazon Redshift Data API

Calling the Data API

Amazon Redshift Data API Reference

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

Question: 8

A company uses Amazon Athena for one-time queries against data that is in Amazon S3. The company has several use cases. The company must implement permission controls to separate query processes and access to query history among users, teams, and applications that are in the same AWS account.

Which solution will meet these requirements?

A. Create an S3 bucket for each use case. Create an S3 bucket policy that grants permissions to appropriate individual IAM users. Apply the S3 bucket policy to the S3 bucket.

B. Create an Athena workgroup for each use case. Apply tags to the workgroup. Create an 1AM policy that uses the tags to apply appropriate permissions to the workgroup.

C. Create an JAM role for each use case. Assign appropriate permissions to the role for each use case. Associate the role with Athena.

D. Create an AWS Glue Data Catalog resource policy that grants permissions to appropriate individual IAM users for each use case. Apply the resource policy to the specific tables that Athena uses.

Answer: B

Explanation:

Athena workgroups are a way to isolate query execution and query history among users, teams, and applications that share the same AWS account. By creating a workgroup for each use case, the company can control the access and actions on the workgroup resource using resource-level IAM permissions or identity-based IAM policies. The company can also use tags to organize and identify the workgroups, and use them as conditions in the IAM policies to grant or deny permissions to the workgroup. This solution meets the requirements of separating query processes and access to query history among users, teams, and applications that are in the same AWS account.
Reference:

Athena Workgroups
IAM policies for accessing workgroups
Workgroup example policies

Question: 9

A data engineer needs to schedule a workflow that runs a set of AWS Glue jobs every day. The data engineer does not require the Glue jobs to run or finish at a specific time.
Which solution will run the Glue jobs in the MOST cost-effective way?

A. Choose the FLEX execution class in the Glue job properties.
B. Use the Spot Instance type in Glue job properties.
C. Choose the STANDARD execution class in the Glue job properties.
D. Choose the latest version in the GlueVersion field in the Glue job properties.

Answer: A

Explanation:
The FLEX execution class allows you to run AWS Glue jobs on spare compute capacity instead of dedicated hardware. This can reduce the cost of running non-urgent or non-time sensitive data integration workloads, such as testing and one-time data loads. The FLEX execution class is available for AWS Glue 3.0 Spark jobs. The other options are not as cost-effective as FLEX, because they either use dedicated resources (STANDARD) or do not affect the cost at all (Spot Instance type and GlueVersion). Reference:

Introducing AWS Glue Flex jobs: Cost savings on ETL workloads
Serverless Data Integration - AWS Glue Pricing
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 125)

Question: 10

A data engineer needs to create an AWS Lambda function that converts the format of data from .csv to Apache Parquet. The Lambda function must run only if a user uploads a .csv file to an Amazon S3 bucket.
Which solution will meet these requirements with the LEAST operational overhead?

A. Create an S3 event notification that has an event type of s3:ObjectCreated:*. Use a filter rule to generate notifications only when the suffix includes .csv. Set the Amazon Resource Name (ARN) of the Lambda function as the destination for the event notification.
B. Create an S3 event notification that has an event type of s3:ObjectTagging:* for objects that have a tag set to .csv. Set the Amazon Resource Name (ARN) of the Lambda function as the destination for the event notification.
C. Create an S3 event notification that has an event type of s3:*. Use a filter rule to generate

notifications only when the suffix includes .csv. Set the Amazon Resource Name (ARN) of the Lambda function as the destination for the event notification.

D. Create an S3 event notification that has an event type of s3:ObjectCreated:*. Use a filter rule to generate notifications only when the suffix includes .csv. Set an Amazon Simple Notification Service (Amazon SNS) topic as the destination for the event notification. Subscribe the Lambda function to the SNS topic.

Answer: A

Explanation:

Option A is the correct answer because it meets the requirements with the least operational overhead. Creating an S3 event notification that has an event type of s3:ObjectCreated:* will trigger the Lambda function whenever a new object is created in the S3 bucket. Using a filter rule to generate notifications only when the suffix includes .csv will ensure that the Lambda function only runs for .csv files. Setting the ARN of the Lambda function as the destination for the event notification will directly invoke the Lambda function without any additional steps.

Option B is incorrect because it requires the user to tag the objects with .csv, which adds an extra step and increases the operational overhead.

Option C is incorrect because it uses an event type of s3:*, which will trigger the Lambda function for any S3 event, not just object creation. This could result in unnecessary invocations and increased costs.

Option D is incorrect because it involves creating and subscribing to an SNS topic, which adds an extra layer of complexity and operational overhead.

Reference:

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 3: Data Ingestion and Transformation, Section 3.2: S3 Event Notifications and Lambda Functions, Pages 67-69 Building Batch Data Analytics Solutions on AWS, Module 4: Data Transformation, Lesson 4.2: AWS Lambda, Pages 4-8

AWS Documentation Overview, AWS Lambda Developer Guide, Working with AWS Lambda Functions, Configuring Function Triggers, Using AWS Lambda with Amazon S3, Pages 1-5

Question: 11

A data engineer needs Amazon Athena queries to finish faster. The data engineer notices that all the files the Athena queries use are currently stored in uncompressed .csv format. The data engineer also notices that users perform most queries by selecting a specific column.

Which solution will MOST speed up the Athena query performance?

A. Change the data format from .csvto JSON format. Apply Snappy compression.
B. Compress the .csv files by using Snappy compression.
C. Change the data format from .csvto Apache Parquet. Apply Snappy compression.
D. Compress the .csv files by using gzjg compression.

Answer: C

Explanation:

Amazon Athena is a serverless interactive query service that allows you to analyze data in Amazon S3 using standard SQL. Athena supports various data formats, such as CSV, JSON, ORC, Avro, and Parquet. However, not all data formats are equally efficient for querying. Some data formats, such as CSV and JSON, are row-oriented, meaning that they store data as a sequence of records, each with the same fields. Row-oriented formats are suitable for loading and exporting data, but they are not optimal for analytical queries that often access only a

subset of columns. Row-oriented formats also do not support compression or encoding techniques that can reduce the data size and improve the query performance.

On the other hand, some data formats, such as ORC and Parquet, are column-oriented, meaning that they store data as a collection of columns, each with a specific data type. Column-oriented formats are ideal for analytical queries that often filter, aggregate, or join data by columns. Column-oriented formats also support compression and encoding techniques that can reduce the data size and improve the query performance. For example, Parquet supports dictionary encoding, which replaces repeated values with numeric codes, and run-length encoding, which replaces consecutive identical values with a single value and a count. Parquet also supports various compression algorithms, such as Snappy, GZIP, and ZSTD, that can further reduce the data size and improve the query performance. Therefore, changing the data format from CSV to Parquet and applying Snappy compression will most speed up the Athena query performance. Parquet is a column-oriented format that allows Athena to scan only the relevant columns and skip the rest, reducing the amount of data read from S3. Snappy is a compression algorithm that reduces the data size without compromising the query speed, as it is splittable and does not require decompression before reading. This solution will also reduce the cost of Athena queries, as Athena charges based on the amount of data scanned from S3. The other options are not as effective as changing the data format to Parquet and applying Snappy compression. Changing the data format from CSV to JSON and applying Snappy compression will not improve the query performance significantly, as JSON is also a row-oriented format that does not support columnar access or encoding techniques. Compressing the CSV files by using Snappy compression will reduce the data size, but it will not improve the query performance significantly, as CSV is still a row-oriented format that does not support columnar access or encoding techniques. Compressing the CSV files by using gzjg compression will reduce the data size, but it will degrade the query performance, as gzjg is not a splittable compression algorithm and requires decompression before reading. Reference:

Amazon Athena

Choosing the Right Data Format

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 5: Data Analysis and Visualization, Section 5.1: Amazon Athena

Question: 12

A manufacturing company collects sensor data from its factory floor to monitor and enhance operational efficiency. The company uses Amazon Kinesis Data Streams to publish the data that the sensors collect to a data stream. Then Amazon Kinesis Data Firehose writes the data to an Amazon S3 bucket.

The company needs to display a real-time view of operational efficiency on a large screen in the manufacturing facility.

Which solution will meet these requirements with the LOWEST latency?

A. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to process the sensor data. Use a connector for Apache Flink to write data to an Amazon Timestream database. Use the Timestream database as a source to create a Grafana dashboard.

B. Configure the S3 bucket to send a notification to an AWS Lambda function when any new object is created. Use the Lambda function to publish the data to Amazon Aurora. Use Aurora as a source to create an Amazon QuickSight dashboard.

C. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to process the sensor data. Create a new Data Firehose delivery stream to publish data directly to an Amazon Timestream database. Use the Timestream database as a source to create an Amazon QuickSight dashboard.

D. Use AWS Glue bookmarks to read sensor data from the S3 bucket in real time. Publish the data to an Amazon

Timestream database. Use the Timestream database as a source to create a Grafana dashboard.

Answer: C

Explanation:

This solution will meet the requirements with the lowest latency because it uses Amazon Managed Service for Apache Flink to process the sensor data in real time and write it to Amazon Timestream, a fast, scalable, and serverless time series database. Amazon Timestream is optimized for storing and analyzing time series data, such as sensor data, and can handle trillions of events per day with millisecond latency. By using Amazon Timestream as a source, you can create an Amazon QuickSight dashboard that displays a real-time view of operational efficiency on a large screen in the manufacturing facility. Amazon QuickSight is a fully managed business intelligence service that can connect to various data sources, including Amazon Timestream, and provide interactive visualizations and insights123.

The other options are not optimal for the following reasons:

A . Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to process the sensor data. Use a connector for Apache Flink to write data to an Amazon Timestream database. Use the Timestream database as a source to create a Grafana dashboard. This option is similar to option C, but it uses Grafana instead of Amazon QuickSight to create the dashboard. Grafana is an open source visualization tool that can also connect to Amazon Timestream, but it requires additional steps to set up and configure, such as deploying a Grafana server on Amazon EC2, installing the Amazon Timestream plugin, and creating an IAM role for Grafana to access Timestream. These steps can increase the latency and complexity of the solution. B . Configure the S3 bucket to send a notification to an AWS Lambda function when any new object is created. Use the Lambda function to publish the data to Amazon Aurora. Use Aurora as a source to create an Amazon QuickSight dashboard. This option is not suitable for displaying a real-time view of operational efficiency, as it introduces unnecessary delays and costs in the data pipeline. First, the sensor data is written to an S3 bucket by Amazon Kinesis Data Firehose, which can have a buffering interval of up to 900 seconds. Then, the S3 bucket sends a notification to a Lambda function, which can incur additional invocation and execution time. Finally, the Lambda function publishes the data to Amazon Aurora, a relational database that is not optimized for time series data and can have higher storage and performance costs than Amazon Timestream .

D . Use AWS Glue bookmarks to read sensor data from the S3 bucket in real time. Publish the data to an Amazon Timestream database. Use the Timestream database as a source to create a Grafana dashboard. This option is also not suitable for displaying a real-time view of operational efficiency, as it uses AWS Glue bookmarks to read sensor data from the S3 bucket. AWS Glue bookmarks are a feature that helps AWS Glue jobs and crawlers keep track of the data that has already been processed, so that they can resume from where they left off. However, AWS Glue jobs and crawlers are not designed for real-time data processing, as they can have a minimum frequency of 5 minutes and a variable start-up time. Moreover, this option also uses Grafana instead of Amazon QuickSight to create the dashboard, which can increase the latency and complexity of the solution .

Reference:

1: Amazon Managed Streaming for Apache Flink 2: Amazon Timestream

3: Amazon QuickSight

: Analyze data in Amazon Timestream using Grafana

: Amazon Kinesis Data Firehose

: Amazon Aurora

: AWS Glue Bookmarks

: AWS Glue Job and Crawler Scheduling

Question: 13

A company stores daily records of the financial performance of investment portfolios in .csv format in an

Amazon S3 bucket. A data engineer uses AWS Glue crawlers to crawl the S3 data.

The data engineer must make the S3 data accessible daily in the AWS Glue Data Catalog. Which solution will meet these requirements?

A. Create an IAM role that includes the AmazonS3FullAccess policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler's data store. Create a daily schedule to run the crawler. Configure the output destination to a new path in the existing S3 bucket.

B. Create an IAM role that includes the AWSGlueServiceRole policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler's data store. Create a daily schedule to run the crawler. Specify a database name for the output.

C. Create an IAM role that includes the AmazonS3FullAccess policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler's data store. Allocate data processing units (DPUs) to run the crawler every day. Specify a database name for the output.

D. Create an IAM role that includes the AWSGlueServiceRole policy. Associate the role with the crawler. Specify the S3 bucket path of the source data as the crawler's data store. Allocate data processing units (DPUs) to run the crawler every day. Configure the output destination to a new path in the existing S3 bucket.

Answer: B

Explanation:

To make the S3 data accessible daily in the AWS Glue Data Catalog, the data engineer needs to create a crawler that can crawl the S3 data and write the metadata to the Data Catalog. The crawler also needs to run on a daily schedule to keep the Data Catalog updated with the latest data. Therefore, the solution must include the following steps:

Create an IAM role that has the necessary permissions to access the S3 data and the Data Catalog. The AWSGlueServiceRole policy is a managed policy that grants these permissions1. Associate the role with the crawler.

Specify the S3 bucket path of the source data as the crawler's data store. The crawler will scan the data and infer the schema and format2.

Create a daily schedule to run the crawler. The crawler will run at the specified time every day and update the Data Catalog with any changes in the data3.

Specify a database name for the output. The crawler will create or update a table in the Data Catalog under the specified database. The table will contain the metadata about the data in the S3 bucket, such as the location, schema, and classification.

Option B is the only solution that includes all these steps. Therefore, option B is the correct answer. Option A is incorrect because it configures the output destination to a new path in the existing S3 bucket. This is unnecessary and may cause confusion, as the crawler does not write any data to the S3 bucket, only metadata to the Data Catalog.

Option C is incorrect because it allocates data processing units (DPUs) to run the crawler every day. This is also unnecessary, as DPUs are only used for AWS Glue ETL jobs, not crawlers.

Option D is incorrect because it combines the errors of option A and C. It configures the output destination to a new path in the existing S3 bucket and allocates DPUs to run the crawler every day, both of which are irrelevant for the crawler.

Reference:

1: AWS managed (predefined) policies for AWS Glue - AWS Glue
2: Data Catalog and crawlers in AWS Glue - AWS Glue
3: Scheduling an AWS Glue crawler - AWS Glue
[4] : Parameters set on Data Catalog tables by crawler - AWS Glue

[5] : AWS Glue pricing - Amazon Web Services (AWS)

Question: 14

A company loads transaction data for each day into Amazon Redshift tables at the end of each day. The company wants to have the ability to track which tables have been loaded and which tables still need to be loaded. A data engineer wants to store the load statuses of Redshift tables in an Amazon DynamoDB table. The data engineer creates an AWS Lambda function to publish the details of the load statuses to DynamoDB. How should the data engineer invoke the Lambda function to write load statuses to the DynamoDB table?

A. Use a second Lambda function to invoke the first Lambda function based on Amazon CloudWatch events.
B. Use the Amazon Redshift Data API to publish an event to Amazon EventBridqe. Configure an EventBridge rule to invoke the Lambda function.
C. Use the Amazon Redshift Data API to publish a message to an Amazon Simple Queue Service (Amazon SQS) queue. Configure the SQS queue to invoke the Lambda function.
D. Use a second Lambda function to invoke the first Lambda function based on AWS CloudTrail events.

Answer: C

Explanation:
The Amazon Redshift Data API enables you to interact with your Amazon Redshift data warehouse in an easy and secure way. You can use the Data API to run SQL commands, such as loading data into tables, without requiring a persistent connection to the cluster. The Data API also integrates with Amazon EventBridge, which allows you to monitor the execution status of your SQL commands and trigger actions based on events. By using the Data API to publish an event to EventBridge, the data engineer can invoke the Lambda function that writes the load statuses to the DynamoDB table. This solution is scalable, reliable, and cost-effective. The other options are either not possible or not optimal. You cannot use a second Lambda function to invoke the first Lambda function based on CloudWatch or CloudTrail events, as these services do not capture the load status of Redshift tables. You can use the Data API to publish a message to an SQS queue, but this would require additional configuration and polling logic to invoke the Lambda function from the queue. This would also introduce additional latency and cost.
Reference:
Using the Amazon Redshift Data API
Using Amazon EventBridge with Amazon Redshift
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 2: Data Store Management, Section 2.2: Amazon Redshift

Question: 15

A data engineer needs to securely transfer 5 TB of data from an on-premises data center to an Amazon S3 bucket. Approximately 5% of the data changes every day. Updates to the data need to be regularly proliferated to the S3 bucket. The data includes files that are in multiple formats. The data engineer needs to automate the transfer process and must schedule the process to run periodically. Which AWS service should the data engineer use to transfer the data in the MOST operationally efficient way?

A. AWS DataSync
B. AWS Glue
C. AWS Direct Connect
D. Amazon S3 Transfer Acceleration

Answer: A

Explanation:

AWS DataSync is an online data movement and discovery service that simplifies and accelerates data migrations to AWS as well as moving data to and from on-premises storage, edge locations, other cloud providers, and AWS Storage services! AWS DataSync can copy data to and from various sources and targets, including Amazon S3, and handle files in multiple formats. AWS DataSync also supports incremental transfers, meaning it can detect and copy only the changes to the data, reducing the amount of data transferred and improving the performance. AWS DataSync can automate and schedule the transfer process using triggers, and monitor the progress and status of the transfers using CloudWatch metrics and eventsl.

AWS DataSync is the most operationally efficient way to transfer the data in this scenario, as it meets all the requirements and offers a serverless and scalable solution. AWS Glue, AWS Direct Connect, and Amazon S3 Transfer Acceleration are not the best options for this scenario, as they have some limitations or drawbacks compared to AWS DataSync. AWS Glue is a serverless ETL service that can extract, transform, and load data from various sources to various targets, including Amazon S32. However, AWS Glue is not designed for large-scale data transfers, as it has some quotas and limits on the number and size of files it can process3. AWS Glue also does not support incremental transfers, meaning it would have to copy the entire data set every time, which would be inefficient and costly.

AWS Direct Connect is a service that establishes a dedicated network connection between your onpremises data center and AWS, bypassing the public internet and improving the bandwidth and performance of the data transfer. However, AWS Direct Connect is not a data transfer service by itself, as it requires additional services or tools to copy the data, such as AWS DataSync, AWS Storage Gateway, or AWS CLI. AWS Direct Connect also has some hardware and location requirements, and charges you for the port hours and data transfer out of AWS.

Amazon S3 Transfer Acceleration is a feature that enables faster data transfers to Amazon S3 over long distances, using the AWS edge locations and optimized network paths. However, Amazon S3 Transfer Acceleration is not a data transfer service by itself, as it requires additional services or tools to copy the data, such as AWS CLI, AWS SDK, or third-party software. Amazon S3 Transfer Acceleration also charges you for the data transferred over the accelerated endpoints, and does not guarantee a performance improvement for every transfer, as it depends on various factors such as the network conditions, the distance, and the object size. Reference:

AWS DataSync

AWS Glue

AWS Glue quotas and limits

[AWS Direct Connect]

[Data transfer options for AWS Direct Connect]

[Amazon S3 Transfer Acceleration]

[Using Amazon S3 Transfer Acceleration]

Question: 16

A company uses an on-premises Microsoft SQL Server database to store financial transaction dat a. The company migrates the transaction data from the on-premises database to AWS at the end of each month. The company has noticed that the cost to migrate data from the on-premises database to an Amazon RDS for SQL Server database has increased recently.

The company requires a cost-effective solution to migrate the data to AWS. The solution must cause minimal downtown for the applications that access the database.

Which AWS service should the company use to meet these requirements?

A. AWS Lambda

B. AWS Database Migration Service (AWS DMS)
C. AWS Direct Connect
D. AWS DataSync

Answer: B

Explanation:
AWS Database Migration Service (AWS DMS) is a cloud service that makes it possible to migrate relational databases, data warehouses, NoSQL databases, and other types of data stores to AWS quickly, securely, and with minimal downtime and zero data loss1. AWS DMS supports migration between 20-plus database and analytics engines, such as Microsoft SQL Server to Amazon RDS for SQL Server2. AWS DMS takes over many of the difficult or tedious tasks involved in a migration project, such as capacity analysis, hardware and software procurement, installation and administration, testing and debugging, and ongoing replication and monitoring1. AWS DMS is a costeffective solution, as you only pay for the compute resources and additional log storage used during the migration process2. AWS DMS is the best solution for the company to migrate the financial transaction data from the on-premises Microsoft SQL Server database to AWS, as it meets the requirements of minimal downtime, zero data loss, and low cost.
Option A is not the best solution, as AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers, but it does not provide any built-in features for database migration. You would have to write your own code to extract, transform, and load the data from the source to the target, which would increase the operational overhead and complexity. Option C is not the best solution, as AWS Direct Connect is a service that establishes a dedicated network connection from your premises to AWS, but it does not provide any built-in features for database migration. You would still need to use another service or tool to perform the actual data transfer, which would increase the cost and complexity.
Option D is not the best solution, as AWS DataSync is a service that makes it easy to transfer data between on-premises storage systems and AWS storage services, such as Amazon S3, Amazon EFS, and Amazon FSx for Windows File Server, but it does not support Amazon RDS for SQL Server as a target. You would have to use another service or tool to migrate the data from Amazon S3 to Amazon RDS for SQL Server, which would increase the latency and complexity. Reference: Database Migration - AWS Database Migration Service - AWS
What is AWS Database Migration Service?
AWS Database Migration Service Documentation
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

Question: 17

A data engineer is building a data pipeline on AWS by using AWS Glue extract, transform, and load (ETL) jobs. The data engineer needs to process data from Amazon RDS and MongoDB, perform transformations, and load the transformed data into Amazon Redshift for analytics. The data updates must occur every hour.
Which combination of tasks will meet these requirements with the LEAST operational overhead? (Choose two.)

A. Configure AWS Glue triggers to run the ETL jobs even/ hour.
B. Use AWS Glue DataBrewto clean and prepare the data for analytics.
C. Use AWS Lambda functions to schedule and run the ETL jobs even/ hour.
D. Use AWS Glue connections to establish connectivity between the data sources and Amazon Redshift.
E. Use the Redshift Data API to load transformed data into Amazon Redshift.

Answer: A D

Explanation:
The correct answer is to configure AWS Glue triggers to run the ETL jobs every hour and use AWS Glue connections to establish connectivity between the data sources and Amazon Redshift. AWS Glue triggers are a way to schedule and orchestrate ETL jobs with the least operational overhead. AWS Glue connections are a way to securely connect to data sources and targets using JDBC or MongoDB drivers. AWS Glue DataBrew is a visual data preparation tool that does not support MongoDB as a data source. AWS Lambda functions are a serverless option to schedule and run ETL jobs, but they have a limit of 15 minutes for execution time, which may not be enough for complex transformations. The Redshift Data API is a way to run SQL commands on Amazon Redshift clusters without needing a persistent connection, but it does not support loading data from AWS Glue ETL jobs.

Reference:

AWS Glue triggers
AWS Glue connections
AWS Glue DataBrew
[AWS Lambda functions] [Redshift Data API]

Question: 18

A company uses an Amazon Redshift cluster that runs on RA3 nodes. The company wants to scale read and write capacity to meet demand. A data engineer needs to identify a solution that will turn on concurrency scaling. Which solution will meet this requirement?

A. Turn on concurrency scaling in workload management (WLM) for Redshift Serverless workgroups.
B. Turn on concurrency scaling at the workload management (WLM) queue level in the Redshift cluster.
C. Turn on concurrency scaling in the settings during the creation of and new Redshift cluster.
D. Turn on concurrency scaling for the daily usage quota for the Redshift cluster.

Answer: B

Explanation:
Concurrency scaling is a feature that allows you to support thousands of concurrent users and queries, with consistently fast query performance. When you turn on concurrency scaling, Amazon Redshift automatically adds query processing power in seconds to process queries without any delays. You can manage which queries are sent to the concurrency-scaling cluster by configuring WLM queues. To turn on concurrency scaling for a queue, set the Concurrency Scaling mode value to auto. The other options are either incorrect or irrelevant, as they do not enable concurrency scaling for the existing Redshift cluster on RA3 nodes. Reference:

Working with concurrency scaling - Amazon Redshift
Amazon Redshift Concurrency Scaling - Amazon Web Services
Configuring concurrency scaling queues - Amazon Redshift
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 6, page 163)

Question: 19

A data engineer must orchestrate a series of Amazon Athena queries that will run every day. Each query can run for more than 15 minutes.
Which combination of steps will meet these requirements MOST cost-effectively? (Choose two.)

A. Use an AWS Lambda function and the Athena Boto3 client start_query_execution API call to invoke the Athena queries programmatically.

B. Create an AWS Step Functions workflow and add two states. Add the first state before the Lambda function. Configure the second state as a Wait state to periodically check whether the Athena query has finished using the Athena Boto3 get_query_execution API call. Configure the workflow to invoke the next query when the current query has finished running.

C. Use an AWS Glue Python shell job and the Athena Boto3 client start_query_execution API call to invoke the Athena queries programmatically.

D. Use an AWS Glue Python shell script to run a sleep timer that checks every 5 minutes to determine whether the current Athena query has finished running successfully. Configure the Python shell script to invoke the next query when the current query has finished running.

E. Use Amazon Managed Workflows for Apache Airflow (Amazon MWAA) to orchestrate the Athena queries in AWS Batch.

Answer: A B

Explanation:

Option A and B are the correct answers because they meet the requirements most cost-effectively. Using an AWS Lambda function and the Athena Boto3 client start_query_execution API call to invoke the Athena queries programmatically is a simple and scalable way to orchestrate the queries. Creating an AWS Step Functions workflow and adding two states to check the query status and invoke the next query is a reliable and efficient way to handle the long-running queries.

Option C is incorrect because using an AWS Glue Python shell job to invoke the Athena queries programmatically is more expensive than using a Lambda function, as it requires provisioning and running a Glue job for each query.

Option D is incorrect because using an AWS Glue Python shell script to run a sleep timer that checks every 5 minutes to determine whether the current Athena query has finished running successfully is not a cost-effective or reliable way to orchestrate the queries, as it wastes resources and time. Option E is incorrect because using Amazon Managed Workflows for Apache Airflow (Amazon MWAA) to orchestrate the Athena queries in AWS Batch is an overkill solution that introduces unnecessary complexity and cost, as it requires setting up and managing an Airflow environment and an AWS Batch compute environment.

Reference:

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 5: Data Orchestration, Section 5.2: AWS Lambda, Section 5.3: AWS Step Functions, Pages 125-135 Building Batch Data Analytics Solutions on AWS, Module 5: Data Orchestration, Lesson 5.1: AWS Lambda, Lesson 5.2: AWS Step Functions, Pages 1-15

AWS Documentation Overview, AWS Lambda Developer Guide, Working with AWS Lambda Functions, Configuring Function Triggers, Using AWS Lambda with Amazon Athena, Pages 1-4 AWS Documentation Overview, AWS Step Functions Developer Guide, Getting Started, Tutorial: Create a Hello World Workflow, Pages 1-8

Question: 20

A company is migrating on-premises workloads to AWS. The company wants to reduce overall operational overhead. The company also wants to explore serverless options.

The company's current workloads use Apache Pig, Apache Oozie, Apache Spark, Apache Hbase, and Apache Flink. The on-premises workloads process petabytes of data in seconds. The company must maintain similar or better performance after the migration to AWS.

Which extract, transform, and load (ETL) service will meet these requirements?

A. AWS Glue
B. Amazon EMR
C. AWS Lambda
D. Amazon Redshift

Answer: B

Explanation:
AWS Glue is a fully managed serverless ETL service that can handle petabytes of data in seconds. AWS Glue can run Apache Spark and Apache Flink jobs without requiring any infrastructure provisioning or management. AWS Glue can also integrate with Apache Pig, Apache Oozie, and Apache Hbase using AWS Glue Data Catalog and AWS Glue workflows. AWS Glue can reduce the overall operational overhead by automating the data discovery, data preparation, and data loading processes. AWS Glue can also optimize the cost and performance of ETL jobs by using AWS Glue Job Bookmarking, AWS Glue Crawlers, and AWS Glue Schema Registry. Reference:
AWS Glue
AWS Glue Data Catalog
AWS Glue Workflows
[AWS Glue Job Bookmarking]
[AWS Glue Crawlers]
[AWS Glue Schema Registry]
[AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

Question: 21

A data engineer must use AWS services to ingest a dataset into an Amazon S3 data lake. The data engineer profiles the dataset and discovers that the dataset contains personally identifiable information (PII). The data engineer must implement a solution to profile the dataset and obfuscate the PII.
Which solution will meet this requirement with the LEAST operational effort?

A. Use an Amazon Kinesis Data Firehose delivery stream to process the dataset. Create an AWS Lambda transform function to identify the PII. Use an AWS SDK to obfuscate the PII. Set the S3 data lake as the target for the delivery stream.
B. Use the Detect PII transform in AWS Glue Studio to identify the PII. Obfuscate the PII. Use an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake.
C. Use the Detect PII transform in AWS Glue Studio to identify the PII. Create a rule in AWS Glue Data Quality to obfuscate the PII. Use an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake.
D. Ingest the dataset into Amazon DynamoDB. Create an AWS Lambda function to identify and obfuscate the PII in the DynamoDB table and to transform the data. Use the same Lambda function to ingest the data into the S3 data lake.

Answer: C

Explanation:
AWS Glue is a fully managed service that provides a serverless data integration platform for data preparation, data cataloging, and data loading. AWS Glue Studio is a graphical interface that allows you to easily author, run, and

monitor AWS Glue ETL jobs. AWS Glue Data Quality is a feature that enables you to validate, cleanse, and enrich your data using predefined or custom rules. AWS Step Functions is a service that allows you to coordinate multiple AWS services into serverless workflows. Using the Detect PII transform in AWS Glue Studio, you can automatically identify and label the PII in your dataset, such as names, addresses, phone numbers, email addresses, etc. You can then create a rule in AWS Glue Data Quality to obfuscate the PII, such as masking, hashing, or replacing the values with dummy data. You can also use other rules to validate and cleanse your data, such as checking for null values, duplicates, outliers, etc. You can then use an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake. You can use AWS Glue DataBrew to visually explore and transform the data, AWS Glue crawlers to discover and catalog the data, and AWS Glue jobs to load the data into the S3 data lake.

This solution will meet the requirement with the least operational effort, as it leverages the serverless and managed capabilities of AWS Glue, AWS Glue Studio, AWS Glue Data Quality, and AWS Step Functions. You do not need to write any code to identify or obfuscate the PII, as you can use the built-in transforms and rules in AWS Glue Studio and AWS Glue Data Quality. You also do not need to provision or manage any servers or clusters, as AWS Glue and AWS Step Functions scale automatically based on the demand.

The other options are not as efficient as using the Detect PII transform in AWS Glue Studio, creating a rule in AWS Glue Data Quality, and using an AWS Step Functions state machine. Using an Amazon Kinesis Data Firehose delivery stream to process the dataset, creating an AWS Lambda transform function to identify the PII, using an AWS SDK to obfuscate the PII, and setting the S3 data lake as the target for the delivery stream will require more operational effort, as you will need to write and maintain code to identify and obfuscate the PII, as well as manage the Lambda function and its resources. Using the Detect PII transform in AWS Glue Studio to identify the PII, obfuscating the PII, and using an AWS Step Functions state machine to orchestrate a data pipeline to ingest the data into the S3 data lake will not be as effective as creating a rule in AWS Glue Data Quality to obfuscate the PII, as you will need to manually obfuscate the PII after identifying it, which can be error-prone and time-consuming. Ingesting the dataset into Amazon DynamoDB, creating an AWS Lambda function to identify and obfuscate the PII in the DynamoDB table and to transform the data, and using the same Lambda function to ingest the data into the S3 data lake will require more operational effort, as you will need to write and maintain code to identify and obfuscate the PII, as well as manage the Lambda function and its resources. You will also incur additional costs and complexity by using DynamoDB as an intermediate data store, which may not be necessary for your use case.

Reference:

AWS Glue

AWS Glue Studio

AWS Glue Data Quality

[AWS Step Functions]

[AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide], Chapter 6: Data Integration and Transformation, Section 6.1: AWS Glue

---

Question: 22

A company maintains multiple extract, transform, and load (ETL) workflows that ingest data from the company's operational databases into an Amazon S3 based data lake. The ETL workflows use AWS Glue and Amazon EMR to process data.

The company wants to improve the existing architecture to provide automated orchestration and to require minimal manual effort.

Which solution will meet these requirements with the LEAST operational overhead?

A. AWS Glue workflows

B. AWS Step Functions tasks

C. AWS Lambda functions

D. Amazon Managed Workflows for Apache Airflow (Amazon MWAA) workflows

Answer: A

Explanation:

AWS Glue workflows are a feature of AWS Glue that enable you to create and visualize complex ETL pipelines using AWS Glue components, such as crawlers, jobs, triggers, and development endpoints. AWS Glue workflows provide automated orchestration and require minimal manual effort, as they handle dependency resolution, error handling, state management, and resource allocation for your ETL workflows. You can use AWS Glue workflows to ingest data from your operational databases into your Amazon S3 based data lake, and then use AWS Glue and Amazon EMR to process the data in the data lake. This solution will meet the requirements with the least operational overhead, as it leverages the serverless and fully managed nature of AWS Glue, and the scalability and flexibility of Amazon EMR12.

The other options are not optimal for the following reasons:

B . AWS Step Functions tasks. AWS Step Functions is a service that lets you coordinate multiple AWS services into serverless workflows. You can use AWS Step Functions tasks to invoke AWS Glue and Amazon EMR jobs as part of your ETL workflows, and use AWS Step Functions state machines to define the logic and flow of your workflows. However, this option would require more manual effort than AWS Glue workflows, as you would need to write JSON code to define your state machines, handle errors and retries, and monitor the execution history and status of your workflows3.

C . AWS Lambda functions. AWS Lambda is a service that lets you run code without provisioning or managing servers. You can use AWS Lambda functions to trigger AWS Glue and Amazon EMR jobs as part of your ETL workflows, and use AWS Lambda event sources and destinations to orchestrate the flow of your workflows. However, this option would also require more manual effort than AWS Glue workflows, as you would need to write code to implement your business logic, handle errors and retries, and monitor the invocation and execution of your Lambda functions. Moreover, AWS Lambda functions have limitations on the execution time, memory, and concurrency, which may affect the performance and scalability of your ETL workflows.

D . Amazon Managed Workflows for Apache Airflow (Amazon MWAA) workflows. Amazon MWAA is a managed service that makes it easy to run open source Apache Airflow on AWS. Apache Airflow is a popular tool for creating and managing complex ETL pipelines using directed acyclic graphs (DAGs). You can use Amazon MWAA workflows to orchestrate AWS Glue and Amazon EMR jobs as part of your ETL workflows, and use the Airflow web interface to visualize and monitor your workflows. However, this option would have more operational overhead than AWS Glue workflows, as you would need to set up and configure your Amazon MWAA environment, write Python code to define your DAGs, and manage the dependencies and versions of your Airflow plugins and operators.

Reference:

1: AWS Glue Workflows

2: AWS Glue and Amazon EMR

3: AWS Step Functions

: AWS Lambda

: Amazon Managed Workflows for Apache Airflow

Question: 23

A company currently stores all of its data in Amazon S3 by using the S3 Standard storage class.

A data engineer examined data access patterns to identify trends. During the first 6 months, most data files are accessed several times each day. Between 6 months and 2 years, most data files are accessed once or twice each month. After 2 years, data files are accessed only once or twice each year.

The data engineer needs to use an S3 Lifecycle policy to develop new data storage rules. The new storage solution must continue to provide high availability.

Which solution will meet these requirements in the MOST cost-effective way?

A. Transition objects to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 6 months. Transfer objects to S3 Glacier Flexible Retrieval after 2 years.

B. Transition objects to S3 Standard-Infrequent Access (S3 Standard-IA) after 6 months. Transfer objects to S3 Glacier Flexible Retrieval after 2 years.

C. Transition objects to S3 Standard-Infrequent Access (S3 Standard-IA) after 6 months. Transfer objects to S3 Glacier Deep Archive after 2 years.

D. Transition objects to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 6 months. Transfer objects to S3 Glacier Deep Archive after 2 years.

Answer: C

Explanation:

To achieve the most cost-effective storage solution, the data engineer needs to use an S3 Lifecycle policy that transitions objects to lower-cost storage classes based on their access patterns, and deletes them when they are no longer needed. The storage classes should also provide high availability, which means they should be resilient to the loss of data in a single Availability Zone1. Therefore, the solution must include the following steps:

Transition objects to S3 Standard-Infrequent Access (S3 Standard-IA) after 6 months. S3 Standard-IA is designed for data that is accessed less frequently, but requires rapid access when needed. It offers the same high durability, throughput, and low latency as S3 Standard, but with a lower storage cost and a retrieval fee2. Therefore, it is suitable for data files that are accessed once or twice each month. S3 Standard-IA also provides high availability, as it stores data redundantly across multiple Availability Zones1.

Transfer objects to S3 Glacier Deep Archive after 2 years. S3 Glacier Deep Archive is the lowest-cost storage class that offers secure and durable storage for data that is rarely accessed and can tolerate a 12-hour retrieval time. It is ideal for long-term archiving and digital preservation3. Therefore, it is suitable for data files that are accessed only once or twice each year. S3 Glacier Deep Archive also provides high availability, as it stores data across at least three geographically dispersed Availability Zones1.

Delete objects when they are no longer needed. The data engineer can specify an expiration action in the S3 Lifecycle policy to delete objects after a certain period of time. This will reduce the storage cost and comply with any data retention policies.

Option C is the only solution that includes all these steps. Therefore, option C is the correct answer. Option A is incorrect because it transitions objects to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 6 months. S3 One Zone-IA is similar to S3 Standard-IA, but it stores data in a single Availability Zone. This means it has a lower availability and durability than S3 Standard-IA, and it is not resilient to the loss of data in a single Availability Zone1. Therefore, it does not provide high availability as required.

Option B is incorrect because it transfers objects to S3 Glacier Flexible Retrieval after 2 years. S3 Glacier Flexible Retrieval is a storage class that offers secure and durable storage for data that is accessed infrequently and can tolerate a retrieval time of minutes to hours. It is more expensive than S3 Glacier Deep Archive, and it is not suitable for data that is accessed only once or twice each year3. Therefore, it is not the most cost-effective option.

Option D is incorrect because it combines the errors of option A and B. It transitions objects to S3 One Zone-IA after 6 months, which does not provide high availability, and it transfers objects to S3 Glacier Flexible Retrieval after 2 years, which is not the most cost-effective option.

Reference:

1: Amazon S3 storage classes - Amazon Simple Storage Service

2: Amazon S3 Standard-Infrequent Access (S3 Standard-IA) - Amazon Simple Storage Service

3: Amazon S3 Glacier and S3 Glacier Deep Archive - Amazon Simple Storage Service

[4] : Expiring objects - Amazon Simple Storage Service

[5] : Managing your storage lifecycle - Amazon Simple Storage Service

[6] : Examples of S3 Lifecycle configuration - Amazon Simple Storage Service

[7] : Amazon S3 Lifecycle further optimizes storage cost savings with new features - What's New with AWS

Question: 24

A company maintains an Amazon Redshift provisioned cluster that the company uses for extract, transform, and load (ETL) operations to support critical analysis tasks. A sales team within the company maintains a Redshift cluster that the sales team uses for business intelligence (BI) tasks. The sales team recently requested access to the data that is in the ETL Redshift cluster so the team can perform weekly summary analysis tasks. The sales team needs to join data from the ETL cluster with data that is in the sales team's BI cluster.

The company needs a solution that will share the ETL cluster data with the sales team without interrupting the critical analysis tasks. The solution must minimize usage of the computing resources of the ETL cluster.

Which solution will meet these requirements?

A. Set up the sales team Bl cluster as a consumer of the ETL cluster by using Redshift data sharing.

B. Create materialized views based on the sales team's requirements. Grant the sales team direct access to the ETL cluster.

C. Create database views based on the sales team's requirements. Grant the sales team direct access to the ETL cluster.

D. Unload a copy of the data from the ETL cluster to an Amazon S3 bucket every week. Create an Amazon Redshift Spectrum table based on the content of the ETL cluster.

Answer: A

Explanation:

Redshift data sharing is a feature that enables you to share live data across different Redshift clusters without the need to copy or move data. Data sharing provides secure and governed access to data, while preserving the performance and concurrency benefits of Redshift. By setting up the sales team BI cluster as a consumer of the ETL cluster, the company can share the ETL cluster data with the sales team without interrupting the critical analysis tasks. The solution also minimizes the usage of the computing resources of the ETL cluster, as the data sharing does not consume any storage space or compute resources from the producer cluster. The other options are either not feasible or not efficient. Creating materialized views or database views would require the sales team to have direct access to the ETL cluster, which could interfere with the critical analysis tasks. Unloading a copy of the data from the ETL cluster to an Amazon S3 bucket every week would introduce additional latency and cost, as well as create data inconsistency issues. Reference:

Sharing data across Amazon Redshift clusters

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 2: Data Store Management, Section 2.2: Amazon Redshift

Question: 25

A data engineer needs to join data from multiple sources to perform a one-time analysis job. The data is stored in

Amazon DynamoDB, Amazon RDS, Amazon Redshift, and Amazon S3.
Which solution will meet this requirement MOST cost-effectively?

A. Use an Amazon EMR provisioned cluster to read from all sources. Use Apache Spark to join the data and perform the analysis.
B. Copy the data from DynamoDB, Amazon RDS, and Amazon Redshift into Amazon S3. Run Amazon Athena queries directly on the S3 files.
C. Use Amazon Athena Federated Query to join the data from all data sources.
D. Use Redshift Spectrum to query data from DynamoDB, Amazon RDS, and Amazon S3 directly from Redshift.

Answer: C

Explanation:
Amazon Athena Federated Query is a feature that allows you to query data from multiple sources using standard SQL. You can use Athena Federated Query to join data from Amazon DynamoDB, Amazon RDS, Amazon Redshift, and Amazon S3, as well as other data sources such as MongoDB, Apache HBase, and Apache Kafka1. Athena Federated Query is a serverless and interactive service, meaning you do not need to provision or manage any infrastructure, and you only pay for the amount of data scanned by your queries. Athena Federated Query is the most cost-effective solution for performing a one-time analysis job on data from multiple sources, as it eliminates the need to copy or move data, and allows you to query data directly from the source.
The other options are not as cost-effective as Athena Federated Query, as they involve additional steps or costs. Option A requires you to provision and pay for an Amazon EMR cluster, which can be expensive and time-consuming for a one-time job. Option B requires you to copy or move data from DynamoDB, RDS, and Redshift to S3, which can incur additional costs for data transfer and storage, and also introduce latency and complexity. Option D requires you to have an existing Redshift cluster, which can be costly and may not be necessary for a one-time job. Option D also does not support querying data from RDS directly, so you would need to use Redshift Federated Query to access RDS data, which adds another layer of complexity2. Reference:

Amazon Athena Federated Query
Redshift Spectrum vs Federated Query

Question: 26

A company is planning to use a provisioned Amazon EMR cluster that runs Apache Spark jobs to perform big data analysis. The company requires high reliability. A big data team must follow best practices for running cost-optimized and long-running workloads on Amazon EMR. The team must find a solution that will maintain the company's current level of performance.
Which combination of resources will meet these requirements MOST cost-effectively? (Choose two.)

A. Use Hadoop Distributed File System (HDFS) as a persistent data store.
B. Use Amazon S3 as a persistent data store.
C. Use x86-based instances for core nodes and task nodes.
D. Use Graviton instances for core nodes and task nodes.
E. Use Spot Instances for all primary nodes.

Answer: B D

Explanation:
The best combination of resources to meet the requirements of high reliability, cost-optimization, and

performance for running Apache Spark jobs on Amazon EMR is to use Amazon S3 as a persistent data store and Graviton instances for core nodes and task nodes.

Amazon S3 is a highly durable, scalable, and secure object storage service that can store any amount of data for a variety of use cases, including big data analytics! Amazon S3 is a better choice than HDFS as a persistent data store for Amazon EMR, as it decouples the storage from the compute layer, allowing for more flexibility and cost-efficiency. Amazon S3 also supports data encryption, versioning, lifecycle management, and cross-region replication! Amazon EMR integrates seamlessly with Amazon S3, using EMR File System (EMRFS) to access data stored in Amazon S3 buckets2. EMRFS also supports consistent view, which enables Amazon EMR to provide read-after-write consistency for Amazon S3 objects that are accessed through EMRFS2.

Graviton instances are powered by Arm-based AWS Graviton2 processors that deliver up to 40% better price performance over comparable current generation x86-based instances3. Graviton instances are ideal for running workloads that are CPU-bound, memory-bound, or network-bound, such as big data analytics, web servers, and open-source databases3. Graviton instances are compatible with Amazon EMR, and can be used for both core nodes and task nodes. Core nodes are responsible for running the data processing frameworks, such as Apache Spark, and storing data in HDFS or the local file system. Task nodes are optional nodes that can be added to a cluster to increase the processing power and throughput. By using Graviton instances for both core nodes and task nodes, you can achieve higher performance and lower cost than using x86-based instances. Using Spot Instances for all primary nodes is not a good option, as it can compromise the reliability and availability of the cluster. Spot Instances are spare EC2 instances that are available at up to 90% discount compared to On-Demand prices, but they can be interrupted by EC2 with a two-minute notice when EC2 needs the capacity back. Primary nodes are the nodes that run the cluster software, such as Hadoop, Spark, Hive, and Hue, and are essential for the cluster operation. If a primary node is interrupted by EC2, the cluster will fail or become unstable. Therefore, it is recommended to use On-Demand Instances or Reserved Instances for primary nodes, and use Spot Instances only for task nodes that can tolerate interruptions. Reference:

Amazon S3 - Cloud Object Storage

EMR File System (EMRFS)

AWS Graviton2 Processor-Powered Amazon EC2 Instances

[Plan and Configure EC2 Instances] [Amazon EC2 Spot Instances] [Best Practices for Amazon EMR]

Question: 27

A company wants to implement real-time analytics capabilities. The company wants to use Amazon Kinesis Data Streams and Amazon Redshift to ingest and process streaming data at the rate of several gigabytes per second. The company wants to derive near real-time insights by using existing business intelligence (BI) and analytics tools.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use Kinesis Data Streams to stage data in Amazon S3. Use the COPY command to load data from Amazon S3 directly into Amazon Redshift to make the data immediately available for real-time analysis.

B. Access the data from Kinesis Data Streams by using SQL queries. Create materialized views directly on top of the stream. Refresh the materialized views regularly to query the most recent stream data.

C. Create an external schema in Amazon Redshift to map the data from Kinesis Data Streams to an Amazon Redshift object. Create a materialized view to read data from the stream. Set the materialized view to auto refresh.

D. Connect Kinesis Data Streams to Amazon Kinesis Data Firehose. Use Kinesis Data Firehose to stage the data in Amazon S3. Use the COPY command to load the data from Amazon S3 to a table in Amazon Redshift.

Answer: C

Explanation:
This solution meets the requirements of implementing real-time analytics capabilities with the least operational overhead. By creating an external schema in Amazon Redshift, you can access the data from Kinesis Data Streams using SQL queries without having to load the data into the cluster. By creating a materialized view on top of the stream, you can store the results of the query in the cluster and make them available for analysis. By setting the materialized view to auto refresh, you can ensure that the view is updated with the latest data from the stream at regular intervals. This way, you can derive near real-time insights by using existing BI and analytics tools.
Reference: Amazon Redshift streaming ingestion
Creating an external schema for Amazon Kinesis Data Streams
Creating a materialized view for Amazon Kinesis Data Streams

Question: 28

A company uses an Amazon QuickSight dashboard to monitor usage of one of the company's applications. The company uses AWS Glue jobs to process data for the dashboard. The company stores the data in a single Amazon S3 bucket. The company adds new data every day.
A data engineer discovers that dashboard queries are becoming slower over time. The data engineer determines that the root cause of the slowing queries is long-running AWS Glue jobs.
Which actions should the data engineer take to improve the performance of the AWS Glue jobs? (Choose two.)

A. Partition the data that is in the S3 bucket. Organize the data by year, month, and day.
B. Increase the AWS Glue instance size by scaling up the worker type.
C. Convert the AWS Glue schema to the DynamicFrame schema class.
D. Adjust AWS Glue job scheduling frequency so the jobs run half as many times each day.
E. Modify the 1AM role that grants access to AWS glue to grant access to all S3 features.

Answer: A B

Explanation:
Partitioning the data in the S3 bucket can improve the performance of AWS Glue jobs by reducing the amount of data that needs to be scanned and processed. By organizing the data by year, month, and day, the AWS Glue job can use partition pruning to filter out irrelevant data and only read the data that matches the query criteria. This can speed up the data processing and reduce the cost of running the AWS Glue job. Increasing the AWS Glue instance size by scaling up the worker type can also improve the performance of AWS Glue jobs by providing more memory and CPU resources for the Spark execution engine. This can help the AWS Glue job handle larger data sets and complex transformations more efficiently. The other options are either incorrect or irrelevant, as they do not affect the performance of the AWS Glue jobs. Converting the AWS Glue schema to the DynamicFrame schema class does not improve the performance, but rather provides additional functionality and flexibility for data manipulation. Adjusting the AWS Glue job scheduling frequency does not improve the performance, but rather reduces the frequency of data updates. Modifying the IAM role that grants access to AWS Glue does not improve the performance, but rather affects the security and permissions of the AWS Glue service. Reference:
Optimising Glue Scripts for Efficient Data Processing: Part 1 (Section: Partitioning Data in S3) Best practices to optimize cost and performance for AWS Glue streaming ETL jobs (Section: Development tools)
Monitoring with AWS Glue job run insights (Section: Requirements) AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 133)

Question: 29

A data engineer needs to use AWS Step Functions to design an orchestration workflow. The workflow must parallel process a large collection of data files and apply a specific transformation to each file. Which Step Functions state should the data engineer use to meet these requirements?

A. Parallel state
B. Choice state
C. Map state
D. Wait state

Answer: C

Explanation:
Option C is the correct answer because the Map state is designed to process a collection of data in parallel by applying the same transformation to each element. The Map state can invoke a nested workflow for each element, which can be another state machine or a Lambda function. The Map state will wait until all the parallel executions are completed before moving to the next state.
Option A is incorrect because the Parallel state is used to execute multiple branches of logic concurrently, not to process a collection of data. The Parallel state can have different branches with different logic and states, whereas the Map state has only one branch that is applied to each element of the collection.
Option B is incorrect because the Choice state is used to make decisions based on a comparison of a value to a set of rules. The Choice state does not process any data or invoke any nested workflows. Option D is incorrect because the Wait state is used to delay the state machine from continuing for a specified time. The Wait state does not process any data or invoke any nested workflows.

Reference:
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 5: Data Orchestration, Section 5.3: AWS Step Functions, Pages 131-132
Building Batch Data Analytics Solutions on AWS, Module 5: Data Orchestration, Lesson 5.2: AWS Step Functions, Pages 9-10
AWS Documentation Overview, AWS Step Functions Developer Guide, Step Functions Concepts, State Types, Map State, Pages 1-3

Question: 30

A company is migrating a legacy application to an Amazon S3 based data lake. A data engineer reviewed data that is associated with the legacy application. The data engineer found that the legacy data contained some duplicate information.

The data engineer must identify and remove duplicate information from the legacy application data. Which solution will meet these requirements with the LEAST operational overhead?

A. Write a custom extract, transform, and load (ETL) job in Python. Use the DataFramedrop duplicatesf) function by importing the Pandas library to perform data deduplication.
B. Write an AWS Glue extract, transform, and load (ETL) job. Use the FindMatches machine learning (ML) transform to transform the data to perform data deduplication.
C. Write a custom extract, transform, and load (ETL) job in Python. Import the Python dedupe library. Use the dedupe library to perform data deduplication.
D. Write an AWS Glue extract, transform, and load (ETL) job. Import the Python dedupe library. Use the dedupe library to perform data deduplication.

Answer: B

Explanation:
AWS Glue is a fully managed serverless ETL service that can handle data deduplication with minimal operational overhead. AWS Glue provides a built-in ML transform called FindMatches, which can automatically identify and group similar records in a dataset. FindMatches can also generate a primary key for each group of records and remove duplicates. FindMatches does not require any coding or prior ML experience, as it can learn from a sample of labeled data provided by the user. FindMatches can also scale to handle large datasets and optimize the cost and performance of the ETL job. Reference:

AWS Glue

FindMatches ML Transform

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

Question: 31

A company is building an analytics solution. The solution uses Amazon S3 for data lake storage and Amazon Redshift for a data warehouse. The company wants to use Amazon Redshift Spectrum to query the data that is in Amazon S3.

Which actions will provide the FASTEST queries? (Choose two.)

A. Use gzip compression to compress individual files to sizes that are between 1 GB and 5 GB.
B. Use a columnar storage file format.
C. Partition the data based on the most common query predicates.
D. Split the data into files that are less than 10 KB.
E. Use file formats that are not

Answer: B C

Explanation: Amazon Redshift Spectrum is a feature that allows you to run SQL queries directly against data in Amazon S3, without loading or transforming the data. Redshift Spectrum can query various data formats, such as CSV, JSON, ORC, Avro, and Parquet. However, not all data formats are equally efficient for querying. Some data formats, such as CSV and JSON, are row-oriented, meaning that they store data as a sequence of records, each with the same fields. Row-oriented formats are suitable for loading and exporting data, but they are not optimal for analytical queries that often access only a subset of columns. Row-oriented formats also do not support compression or encoding techniques that can reduce the data size and improve the query performance.
On the other hand, some data formats, such as ORC and Parquet, are column-oriented, meaning that they store data as a collection of columns, each with a specific data type. Column-oriented formats are ideal for analytical queries that often filter, aggregate, or join data by columns. Column-oriented formats also support compression and encoding techniques that can reduce the data size and improve the query performance. For example, Parquet supports dictionary encoding, which replaces repeated values with numeric codes, and run-length encoding, which replaces consecutive identical values with a single value and a count. Parquet also supports various compression algorithms, such as Snappy, GZIP, and ZSTD, that can further reduce the data size and improve the query performance. Therefore, using a columnar storage file format, such as Parquet, will provide faster queries, as it allows Redshift Spectrum to scan only the relevant columns and skip the rest, reducing the amount of data read from S3. Additionally, partitioning the data based on the most common query predicates, such as date, time, region, etc., will provide faster queries, as it allows Redshift Spectrum to prune the partitions that do not match the query criteria, reducing the amount of data scanned from S3. Partitioning also improves the performance of

joins and aggregations, as it reduces data skew and shufling.

The other options are not as effective as using a columnar storage file format and partitioning the data. Using gzip compression to compress individual files to sizes that are between 1 GB and 5 GB will reduce the data size, but it will not improve the query performance significantly, as gzip is not a splittable compression algorithm and requires decompression before reading. Splitting the data into files that are less than 10 KB will increase the number of files and the metadata overhead, which will degrade the query performance. Using file formats that are not supported by Redshift Spectrum, such as XML, will not work, as Redshift Spectrum will not be able to read or parse the data. Reference:

Amazon Redshift Spectrum

Choosing the Right Data Format

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Lakes and Data Warehouses, Section 4.3: Amazon Redshift Spectrum

---

Question: 32

A company uses Amazon RDS to store transactional dat
a. The company runs an RDS DB instance in a private subnet. A developer wrote an AWS Lambda function with default settings to insert, update, or delete data in the DB instance.

The developer needs to give the Lambda function the ability to connect to the DB instance privately without using the public internet.

Which combination of steps will meet this requirement with the LEAST operational overhead? (Choose two.)

A. Turn on the public access setting for the DB instance.

B. Update the security group of the DB instance to allow only Lambda function invocations on the database port.

C. Configure the Lambda function to run in the same subnet that the DB instance uses.

D. Attach the same security group to the Lambda function and the DB instance. Include a selfreferencing rule that allows access through the database port.

E. Update the network ACL of the private subnet to include a self-referencing rule that allows access through the database port.

Answer: C D

Explanation:

To enable the Lambda function to connect to the RDS DB instance privately without using the public internet, the best combination of steps is to configure the Lambda function to run in the same subnet that the DB instance uses, and attach the same security group to the Lambda function and the DB instance. This way, the Lambda function and the DB instance can communicate within the same private network, and the security group can allow traffic between them on the database port. This solution has the least operational overhead, as it does not require any changes to the public access setting, the network ACL, or the security group of the DB instance.

The other options are not optimal for the following reasons:

A . Turn on the public access setting for the DB instance. This option is not recommended, as it would expose the DB instance to the public internet, which can compromise the security and privacy of the data. Moreover, this option would not enable the Lambda function to connect to the DB instance privately, as it would still require the Lambda function to use the public internet to access the DB instance.

B . Update the security group of the DB instance to allow only Lambda function invocations on the database port. This option is not sufficient, as it would only modify the inbound rules of the security group of the DB instance, but not the outbound rules of the security group of the Lambda function. Moreover, this option would not enable the

Lambda function to connect to the DB instance privately, as it would still require the Lambda function to use the public internet to access the DB instance.

E . Update the network ACL of the private subnet to include a self-referencing rule that allows access through the database port. This option is not necessary, as the network ACL of the private subnet already allows all traffic within the subnet by default. Moreover, this option would not enable the Lambda function to connect to the DB instance privately, as it would still require the Lambda function to use the public internet to access the DB instance.

Reference:

1: Connecting to an Amazon RDS DB instance

2: Configuring a Lambda function to access resources in a VPC

3: Working with security groups

: Network ACLs

Question: 33

A company has a frontend ReactJS website that uses Amazon API Gateway to invoke REST APIs. The APIs perform the functionality of the website. A data engineer needs to write a Python script that can be occasionally invoked through API Gateway. The code must return results to API Gateway. Which solution will meet these requirements with the LEAST operational overhead?

A. Deploy a custom Python script on an Amazon Elastic Container Service (Amazon ECS) cluster.

B. Create an AWS Lambda Python function with provisioned concurrency.

C. Deploy a custom Python script that can integrate with API Gateway on Amazon Elastic Kubernetes Service (Amazon EKS).

D. Create an AWS Lambda function. Ensure that the function is warm by scheduling an Amazon EventBridge rule to invoke the Lambda function every 5 minutes by using mock events.

Answer: B

Explanation:

AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers. You can use Lambda to create functions that perform custom logic and integrate with other AWS services, such as API Gateway. Lambda automatically scales your application by running code in response to each trigger. You pay only for the compute time you consume1. Amazon ECS is a fully managed container orchestration service that allows you to run and scale containerized applications on AWS. You can use ECS to deploy, manage, and scale Docker containers using either Amazon EC2 instances or AWS Fargate, a serverless compute engine for containers2. Amazon EKS is a fully managed Kubernetes service that allows you to run Kubernetes clusters on AWS without needing to install, operate, or maintain your own Kubernetes control plane. You can use EKS to deploy, manage, and scale containerized applications using Kubernetes on AWS3.

The solution that meets the requirements with the least operational overhead is to create an AWS Lambda Python function with provisioned concurrency. This solution has the following advantages: It does not require you to provision, manage, or scale any servers or clusters, as Lambda handles all the infrastructure for you. This reduces the operational complexity and cost of running your code. It allows you to write your Python script as a Lambda function and integrate it with API Gateway using a simple configuration. API Gateway can invoke your Lambda function synchronously or asynchronously, and return the results to the frontend website.

It ensures that your Lambda function is ready to respond to API requests without any cold start delays, by using provisioned concurrency. Provisioned concurrency is a feature that keeps your function initialized and hyper-ready to respond in double-digit milliseconds. You can specify the number of concurrent executions that you want to

provision for your function.

Option A is incorrect because it requires you to deploy a custom Python script on an Amazon ECS cluster. This solution has the following disadvantages:

It requires you to provision, manage, and scale your own ECS cluster, either using EC2 instances or Fargate. This increases the operational complexity and cost of running your code.

It requires you to package your Python script as a Docker container image and store it in a container registry, such as Amazon ECR or Docker Hub. This adds an extra step to your deployment process. It requires you to configure your ECS cluster to integrate with API Gateway, either using an Application Load Balancer or a Network Load Balancer. This adds another layer of complexity to your architecture.

Option C is incorrect because it requires you to deploy a custom Python script that can integrate with API Gateway on Amazon EKS. This solution has the following disadvantages:

It requires you to provision, manage, and scale your own EKS cluster, either using EC2 instances or Fargate. This increases the operational complexity and cost of running your code.

It requires you to package your Python script as a Docker container image and store it in a container registry, such as Amazon ECR or Docker Hub. This adds an extra step to your deployment process. It requires you to configure your EKS cluster to integrate with API Gateway, either using an Application Load Balancer, a Network Load Balancer, or a service of type LoadBalancer. This adds another layer of complexity to your architecture.

Option D is incorrect because it requires you to create an AWS Lambda function and ensure that the function is warm by scheduling an Amazon EventBridge rule to invoke the Lambda function every 5 minutes by using mock events. This solution has the following disadvantages:

It does not guarantee that your Lambda function will always be warm, as Lambda may scale down your function if it does not receive any requests for a long period of time. This may cause cold start delays when your function is invoked by API Gateway.

It incurs unnecessary costs, as you pay for the compute time of your Lambda function every time it is invoked by the EventBridge rule, even if it does not perform any useful work1.

Reference:

1: AWS Lambda - Features

2: Amazon Elastic Container Service - Features

3: Amazon Elastic Kubernetes Service - Features

[4] : Building API Gateway REST API with Lambda integration - Amazon API Gateway

[5] : Improving latency with Provisioned Concurrency - AWS Lambda

[6] : Integrating Amazon ECS with Amazon API Gateway - Amazon Elastic Container Service

[7] : Integrating Amazon EKS with Amazon API Gateway - Amazon Elastic Kubernetes Service

[8] : Managing concurrency for a Lambda function - AWS Lambda

---

Question: 34

A company has a production AWS account that runs company workloads. The company's security team created a security AWS account to store and analyze security logs from the production AWS account. The security logs in the production AWS account are stored in Amazon CloudWatch Logs. The company needs to use Amazon Kinesis Data Streams to deliver the security logs to the security AWS account.

Which solution will meet these requirements?

A. Create a destination data stream in the production AWS account. In the security AWS account, create an IAM role that has cross-account permissions to Kinesis Data Streams in the production AWS account.

B. Create a destination data stream in the security AWS account. Create an IAM role and a trust policy to grant CloudWatch Logs the permission to put data into the stream. Create a subscription filter in the security AWS account.

C. Create a destination data stream in the production AWS account. In the production AWS account, create an IAM role that has cross-account permissions to Kinesis Data Streams in the security AWS account.

D. Create a destination data stream in the security AWS account. Create an IAM role and a trust policy to grant CloudWatch Logs the permission to put data into the stream. Create a subscription filter in the production AWS account.

Answer: D

Explanation:

Amazon Kinesis Data Streams is a service that enables you to collect, process, and analyze real-time streaming data. You can use Kinesis Data Streams to ingest data from various sources, such as Amazon CloudWatch Logs, and deliver it to different destinations, such as Amazon S3 or Amazon Redshift. To use Kinesis Data Streams to deliver the security logs from the production AWS account to the security AWS account, you need to create a destination data stream in the security AWS account. This data stream will receive the log data from the CloudWatch Logs service in the production AWS account. To enable this cross-account data delivery, you need to create an IAM role and a trust policy

in the security AWS account. The IAM role defines the permissions that the CloudWatch Logs service needs to put data into the destination data stream. The trust policy allows the production AWS account to assume the IAM role. Finally, you need to create a subscription filter in the production AWS account. A subscription filter defines the pattern to match log events and the destination to send the matching events. In this case, the destination is the destination data stream in the security AWS account. This solution meets the requirements of using Kinesis Data Streams to deliver the security logs to the security AWS account. The other options are either not possible or not optimal. You cannot create a destination data stream in the production AWS account, as this would not deliver the data to the security AWS account. You cannot create a subscription filter in the security AWS account, as this would not capture the log events from the production AWS account. Reference: Using Amazon Kinesis Data Streams with Amazon CloudWatch Logs

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 3: Data Ingestion and Transformation, Section 3.3: Amazon Kinesis Data Streams

---

Question: 35

A company uses Amazon S3 to store semi-structured data in a transactional data lake. Some of the data files are small, but other data files are tens of terabytes.

A data engineer must perform a change data capture (CDC) operation to identify changed data from the data source. The data source sends a full snapshot as a JSON file every day and ingests the changed data into the data lake.

Which solution will capture the changed data MOST cost-effectively?

A. Create an AWS Lambda function to identify the changes between the previous data and the current data. Configure the Lambda function to ingest the changes into the data lake.

B. Ingest the data into Amazon RDS for MySQL. Use AWS Database Migration Service (AWS DMS) to write the changed data to the data lake.

C. Use an open source data lake format to merge the data source with the S3 data lake to insert the new data and update the existing data.

D. Ingest the data into an Amazon Aurora MySQL DB instance that runs Aurora Serverless. Use AWS Database Migration Service (AWS DMS) to write the changed data to the data lake.

Answer: C

Explanation:

An open source data lake format, such as Apache Parquet, Apache ORC, or Delta Lake, is a costeffective way to perform a change data capture (CDC) operation on semi-structured data stored in Amazon S3. An open source data lake format allows you to query data directly from S3 using standard SQL, without the need to move or copy data to another service. An open source data lake format also supports schema evolution, meaning it can handle changes in the data structure over time. An open source data lake format also supports upserts, meaning it can insert new data and update existing data in the same operation, using a merge command. This way, you can efficiently capture the changes from the data source and apply them to the S3 data lake, without duplicating or losing any data.

The other options are not as cost-effective as using an open source data lake format, as they involve additional steps or costs. Option A requires you to create and maintain an AWS Lambda function, which can be complex and error-prone. AWS Lambda also has some limits on the execution time, memory, and concurrency, which can affect the performance and reliability of the CDC operation. Option B and D require you to ingest the data into a relational database service, such as Amazon RDS or Amazon Aurora, which can be expensive and unnecessary for semi-structured data. AWS Database Migration Service (AWS DMS) can write the changed data to the data lake, but it also charges you for the data replication and transfer. Additionally, AWS DMS does not support JSON as a source data type, so you would need to convert the data to a supported format before using AWS DMS. Reference:

What is a data lake?

Choosing a data format for your data lake

Using the MERGE INTO command in Delta Lake

[AWS Lambda quotas]

[AWS Database Migration Service quotas]

Question: 36

A data engineer runs Amazon Athena queries on data that is in an Amazon S3 bucket. The Athena queries use AWS Glue Data Catalog as a metadata table.

The data engineer notices that the Athena query plans are experiencing a performance bottleneck. The data engineer determines that the cause of the performance bottleneck is the large number of partitions that are in the S3 bucket. The data engineer must resolve the performance bottleneck and reduce Athena query planning time.

Which solutions will meet these requirements? (Choose two.)

A. Create an AWS Glue partition index. Enable partition filtering.

B. Bucket the data based on a column that the data have in common in a WHERE clause of the user query

C. Use Athena partition projection based on the S3 bucket prefix.

D. Transform the data that is in the S3 bucket to Apache Parquet format.

E. Use the Amazon EMR S3DistCP utility to combine smaller objects in the S3 bucket into larger objects.

Answer: A C

Explanation:

The best solutions to resolve the performance bottleneck and reduce Athena query planning time are to create an

AWS Glue partition index and enable partition filtering, and to use Athena partition projection based on the S3 bucket prefix.

AWS Glue partition indexes are a feature that allows you to speed up query processing of highly partitioned tables cataloged in AWS Glue Data Catalog. Partition indexes are available for queries in Amazon EMR, Amazon Redshift Spectrum, and AWS Glue ETL jobs. Partition indexes are sublists of partition keys defined in the table. When you create a partition index, you specify a list of partition keys that already exist on a given table. AWS Glue then creates an index for the specified keys and stores it in the Data Catalog. When you run a query that filters on the partition keys, AWS Glue uses the partition index to quickly identify the relevant partitions without scanning the entire table metadata. This reduces the query planning time and improves the query performance1.

Athena partition projection is a feature that allows you to speed up query processing of highly partitioned tables and automate partition management. In partition projection, Athena calculates partition values and locations using the table properties that you configure directly on your table in AWS Glue. The table properties allow Athena to 'project', or determine, the necessary partition information instead of having to do a more time-consuming metadata lookup in the AWS Glue Data Catalog. Because in-memory operations are often faster than remote operations, partition projection can reduce the runtime of queries against highly partitioned tables. Partition projection also automates partition management because it removes the need to manually create partitions in Athena, AWS Glue, or your external Hive metastore2.

Option B is not the best solution, as bucketing the data based on a column that the data have in common in a WHERE clause of the user query would not reduce the query planning time. Bucketing is a technique that divides data into buckets based on a hash function applied to a column. Bucketing can improve the performance of join queries by reducing the amount of data that needs to be shufled between nodes. However, bucketing does not affect the partition metadata retrieval, which is the main cause of the performance bottleneck in this scenario3.

Option D is not the best solution, as transforming the data that is in the S3 bucket to Apache Parquet format would not reduce the query planning time. Apache Parquet is a columnar storage format that can improve the performance of analytical queries by reducing the amount of data that needs to be scanned and providing efficient compression and encoding schemes. However, Parquet does not affect the partition metadata retrieval, which is the main cause of the performance bottleneck in this scenario4.

Option E is not the best solution, as using the Amazon EMR S3DistCP utility to combine smaller objects in the S3 bucket into larger objects would not reduce the query planning time. S3DistCP is a tool that can copy large amounts of data between Amazon S3 buckets or from HDFS to Amazon S3. S3DistCP can also aggregate smaller files into larger files to improve the performance of sequential access. However, S3DistCP does not affect the partition metadata retrieval, which is the main cause of the performance bottleneck in this scenario5.

Reference:

Improve query performance using AWS Glue partition indexes

Partition projection with Amazon Athena

Bucketing vs Partitioning

Columnar Storage Formats

S3DistCp

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

---

Question: 37

A data engineer must manage the ingestion of real-time streaming data into AWS. The data engineer wants to perform real-time analytics on the incoming streaming data by using time-based aggregations over a window of up to 30 minutes. The data engineer needs a solution that is highly fault tolerant.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use an AWS Lambda function that includes both the business and the analytics logic to perform time-based aggregations over a window of up to 30 minutes for the data in Amazon Kinesis Data Streams.

B. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to analyze the data that might occasionally contain duplicates by using multiple types of aggregations.

C. Use an AWS Lambda function that includes both the business and the analytics logic to perform aggregations for a tumbling window of up to 30 minutes, based on the event timestamp.

D. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to analyze the data by using multiple types of aggregations to perform time-based analytics over a window of up to 30 minutes.

Answer: A

Explanation:

This solution meets the requirements of managing the ingestion of real-time streaming data into AWS and performing real-time analytics on the incoming streaming data with the least operational overhead. Amazon Managed Service for Apache Flink is a fully managed service that allows you to run Apache Flink applications without having to manage any infrastructure or clusters. Apache Flink is a framework for stateful stream processing that supports various types of aggregations, such as tumbling, sliding, and session windows, over streaming data. By using Amazon Managed Service for Apache Flink, you can easily connect to Amazon Kinesis Data Streams as the source and sink of your streaming data, and perform time-based analytics over a window of up to 30 minutes. This solution is also highly fault tolerant, as Amazon Managed Service for Apache Flink automatically scales, monitors, and restarts your Flink applications in case of failures. Reference:

Amazon Managed Service for Apache Flink

Apache Flink

Window Aggregations in Flink

Question: 38

A company is planning to upgrade its Amazon Elastic Block Store (Amazon EBS) General Purpose SSD storage from gp2 to gp3. The company wants to prevent any interruptions in its Amazon EC2 instances that will cause data loss during the migration to the upgraded storage.

Which solution will meet these requirements with the LEAST operational overhead?

A. Create snapshots of the gp2 volumes. Create new gp3 volumes from the snapshots. Attach the new gp3 volumes to the EC2 instances.

B. Create new gp3 volumes. Gradually transfer the data to the new gp3 volumes. When the transfer is complete, mount the new gp3 volumes to the EC2 instances to replace the gp2 volumes.

C. Change the volume type of the existing gp2 volumes to gp3. Enter new values for volume size, IOPS, and throughput.

D. Use AWS DataSync to create new gp3 volumes. Transfer the data from the original gp2 volumes to the new gp3 volumes.

Answer: C

Explanation:

Changing the volume type of the existing gp2 volumes to gp3 is the easiest and fastest way to migrate to the new storage type without any downtime or data loss. You can use the AWS Management Console, the AWS CLI, or the

Amazon EC2 API to modify the volume type, size, IOPS, and throughput of your gp2 volumes. The modification takes effect immediately, and you can monitor the progress of the modification using CloudWatch. The other options are either more complex or require additional steps, such as creating snapshots, transferring data, or attaching new volumes, which can increase the operational overhead and the risk of errors. Reference: Migrating Amazon EBS volumes from gp2 to gp3 and save up to 20% on costs (Section: How to migrate from gp2 to gp3) Switching from gp2 Volumes to gp3 Volumes to Lower AWS EBS Costs (Section: How to Switch from GP2 Volumes to GP3 Volumes)
Modifying the volume type, IOPS, or size of an EBS volume - Amazon Elastic Compute Cloud (Section: Modifying the volume type)

Question: 39

A company is migrating its database servers from Amazon EC2 instances that run Microsoft SQL Server to Amazon RDS for Microsoft SQL Server DB instances. The company's analytics team must export large data elements every day until the migration is complete. The data elements are the result of SQL joins across multiple tables. The data must be in Apache Parquet format. The analytics team must store the data in Amazon S3.
Which solution will meet these requirements in the MOST operationally efficient way?

A. Create a view in the EC2 instance-based SQL Server databases that contains the required data elements. Create an AWS Glue job that selects the data directly from the view and transfers the data in Parquet format to an S3 bucket. Schedule the AWS Glue job to run every day.
B. Schedule SQL Server Agent to run a daily SQL query that selects the desired data elements from the EC2 instance-based SQL Server databases. Configure the query to direct the output .csv objects to an S3 bucket. Create an S3 event that invokes an AWS Lambda function to transform the output format from .csv to Parquet.
C. Use a SQL query to create a view in the EC2 instance-based SQL Server databases that contains the required data elements. Create and run an AWS Glue crawler to read the view. Create an AWS Glue job that retrieves the data and transfers the data in Parquet format to an S3 bucket. Schedule the AWS Glue job to run every day.
D. Create an AWS Lambda function that queries the EC2 instance-based databases by using Java Database Connectivity (JDBC). Configure the Lambda function to retrieve the required data, transform the data into Parquet format, and transfer the data into an S3 bucket. Use Amazon EventBridge to schedule the Lambda function to run every day.

Answer: A

Explanation:
Option A is the most operationally efficient way to meet the requirements because it minimizes the number of steps and services involved in the data export process. AWS Glue is a fully managed service that can extract, transform, and load (ETL) data from various sources to various destinations, including Amazon S3. AWS Glue can also convert data to different formats, such as Parquet, which is a columnar storage format that is optimized for analytics. By creating a view in the SQL Server databases that contains the required data elements, the AWS Glue job can select the data directly from the view without having to perform any joins or transformations on the source data. The AWS Glue job can then transfer the data in Parquet format to an S3 bucket and run on a daily schedule. Option B is not operationally efficient because it involves multiple steps and services to export the data. SQL Server Agent is a tool that can run scheduled tasks on SQL Server databases, such as executing SQL queries. However, SQL Server Agent cannot directly export data to S3, so the query output must be saved as .csv objects on the EC2 instance. Then, an S3 event must be configured to trigger an AWS Lambda function that can transform the .csv objects to Parquet format and upload them to S3. This option adds complexity and latency to the data export

process and requires additional resources and configuration.

Option C is not operationally efficient because it introduces an unnecessary step of running an AWS Glue crawler to read the view. An AWS Glue crawler is a service that can scan data sources and create metadata tables in the AWS Glue Data Catalog. The Data Catalog is a central repository that stores information about the data sources, such as schema, format, and location. However, in this scenario, the schema and format of the data elements are already known and fixed, so there is no need to run a crawler to discover them. The AWS Glue job can directly select the data from the view without using the Data Catalog. Running a crawler adds extra time and cost to the data export process. Option D is not operationally efficient because it requires custom code and configuration to query the databases and transform the data. An AWS Lambda function is a service that can run code in response to events or triggers, such as Amazon EventBridge. Amazon EventBridge is a service that can connect applications and services with event sources, such as schedules, and route them to targets, such as Lambda functions. However, in this scenario, using a Lambda function to query the databases and transform the data is not the best option because it requires writing and maintaining code that uses JDBC to connect to the SQL Server databases, retrieve the required data, convert the data to Parquet format, and transfer the data to S3. This option also has limitations on the execution time, memory, and concurrency of the Lambda function, which may affect the performance and reliability of the data export process.

Reference:

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide AWS Glue Documentation

Working with Views in AWS Glue Converting to Columnar Formats

---

Question: 40

A data engineering team is using an Amazon Redshift data warehouse for operational reporting. The team wants to prevent performance issues that might result from long- running queries. A data engineer must choose a system table in Amazon Redshift to record anomalies when a query optimizer identifies conditions that might indicate performance issues.

Which table views should the data engineer use to meet this requirement?

A. STL USAGE CONTROL
B. STL ALERT EVENT LOG
C. STL QUERY METRICS
D. STL PLAN INFO

Answer: B

Explanation:

The STL ALERT EVENT LOG table view records anomalies when the query optimizer identifies conditions that might indicate performance issues. These conditions include skewed data distribution, missing statistics, nested loop joins, and broadcasted data. The STL ALERT EVENT LOG table view can help the data engineer to identify and troubleshoot the root causes of performance issues and optimize the query execution plan. The other table views are not relevant for this requirement. STL USAGE CONTROL records the usage limits and quotas for Amazon Redshift resources. STL QUERY METRICS records the execution time and resource consumption of queries. STL PLAN INFO records the query execution plan and the steps involved in each query. Reference:

STL ALERT EVENT LOG

System Tables and Views

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

www.atmicnetworks.com

Question: 41

A data engineer must ingest a source of structured data that is in .csv format into an Amazon S3 data lake. The .csv files contain 15 columns. Data analysts need to run Amazon Athena queries on one or two columns of the dataset. The data analysts rarely query the entire file.

Which solution will meet these requirements MOST cost-effectively?

A. Use an AWS Glue PySpark job to ingest the source data into the data lake in .csv format.

B. Create an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data source. Configure the job to ingest the data into the data lake in JSON format.

C. Use an AWS Glue PySpark job to ingest the source data into the data lake in Apache Avro format.

D. Create an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data source. Configure the job to write the data into the data lake in Apache Parquet format.

Answer:D

Amazon Athena is a serverless interactive query service that allows you to analyze data in Amazon S3 using standard SQL. Athena supports various data formats, such as CSV, JSON, ORC, Avro, and Parquet. However, not all data formats are equally efficient for querying. Some data formats, such as CSV and JSON, are row-oriented, meaning that they store data as a sequence of records, each with the same fields. Row-oriented formats are suitable for loading and exporting data, but they are not optimal for analytical queries that often access only a subset of columns. Row-oriented formats also do not support compression or encoding techniques that can reduce the data size and improve the query performance.

On the other hand, some data formats, such as ORC and Parquet, are column-oriented, meaning that they store data as a collection of columns, each with a specific data type. Column-oriented formats are ideal for analytical queries that often filter, aggregate, or join data by columns. Column-oriented formats also support compression and encoding techniques that can reduce the data size and improve the query performance. For example, Parquet supports dictionary encoding, which replaces repeated values with numeric codes, and run-length encoding, which replaces consecutive identical values with a single value and a count. Parquet also supports various compression algorithms, such as Snappy, GZIP, and ZSTD, that can further reduce the data size and improve the query performance. Therefore, creating an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data source and writing the data into the data lake in Apache Parquet format will meet the requirements most cost-effectively. AWS Glue is a fully managed service that provides a serverless data integration platform for data preparation, data cataloging, and data loading. AWS Glue ETL jobs allow you to transform and load data from various sources into various targets, using either a graphical interface (AWS Glue Studio) or a code-based interface (AWS Glue console or AWS Glue API). By using AWS Glue ETL jobs, you can easily convert the data from CSV to Parquet format, without having to write or manage any code. Parquet is a column-oriented format that allows Athena to scan only the relevant columns and skip the rest, reducing the amount of data read from S3. This solution will also reduce the cost of Athena queries, as Athena charges based on the amount of data scanned from S3.

The other options are not as cost-effective as creating an AWS Glue ETL job to write the data into the data lake in Parquet format. Using an AWS Glue PySpark job to ingest the source data into the data lake in .csv format will not improve the query performance or reduce the query cost, as .csv is a row- oriented format that does not support columnar access or compression. Creating an AWS Glue ETL job to ingest the data into the data lake in JSON format will not improve the query performance or reduce the query cost, as JSON is also a row-oriented format that does not support columnar access or compression. Using an AWS Glue PySpark job to ingest the source data into the data lake in Apache Avro format will improve the query performance, as Avro is a column-oriented format that supports compression and encoding, but it will require more operational effort, as you will need to write and

maintain PySpark code to convert the data from CSV to Avro format. Reference: Amazon Athena

Choosing the Right Data Format

AWS Glue

[AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide], Chapter 5: Data Analysis and Visualization, Section 5.1: Amazon Athena

Question: 42

A company has five offices in different AWS Regions. Each office has its own human resources (HR) department that uses a unique IAM role. The company stores employee records in a data lake that is based on Amazon S3 storage.

A data engineering team needs to limit access to the records. Each HR department should be able to access records for only employees who are within the HR department's Region.

Which combination of steps should the data engineering team take to meet this requirement with the LEAST operational overhead? (Choose two.)

A. Use data filters for each Region to register the S3 paths as data locations.

B. Register the S3 path as an AWS Lake Formation location.

C. Modify the IAM roles of the HR departments to add a data filter for each department's Region.

D. Enable fine-grained access control in AWS Lake Formation. Add a data filter for each Region.

E. Create a separate S3 bucket for each Region. Configure an IAM policy to allow S3 access. Restrict access based on Region.

Answer: B D

Explanation:

AWS Lake Formation is a service that helps you build, secure, and manage data lakes on Amazon S3. You can use AWS Lake Formation to register the S3 path as a data lake location, and enable finegrained access control to limit access to the records based on the HR department's Region. You can use data filters to specify which S3 prefixes or partitions each HR department can access, and grant permissions to the IAM roles of the HR departments accordingly. This solution will meet the requirement with the least operational overhead, as it simplifies the data lake management and security, and leverages the existing IAM roles of the HR departments12.

The other options are not optimal for the following reasons:

A . Use data filters for each Region to register the S3 paths as data locations. This option is not possible, as data filters are not used to register S3 paths as data locations, but to grant permissions to access specific S3 prefixes or partitions within a data location. Moreover, this option does not specify how to limit access to the records based on the HR department's Region.

C . Modify the IAM roles of the HR departments to add a data filter for each department's Region.

This option is not possible, as data filters are not added to IAM roles, but to permissions granted by AWS Lake Formation. Moreover, this option does not specify how to register the S3 path as a data lake location, or how to enable fine-grained access control in AWS Lake Formation.

E . Create a separate S3 bucket for each Region. Configure an IAM policy to allow S3 access. Restrict access based on Region. This option is not recommended, as it would require more operational overhead to create and manage multiple S3 buckets, and to configure and maintain IAM policies for each HR department. Moreover, this option does not leverage the benefits of AWS Lake Formation, such as data cataloging, data transformation, and data governance.

Reference:

1: AWS Lake Formation
2: AWS Lake Formation Permissions
: AWS Identity and Access Management
: Amazon S3

Question: 43

A company uses AWS Step Functions to orchestrate a data pipeline. The pipeline consists of Amazon EMR jobs that ingest data from data sources and store the data in an Amazon S3 bucket. The pipeline also includes EMR jobs that load the data to Amazon Redshift.

The company's cloud infrastructure team manually built a Step Functions state machine. The cloud infrastructure team launched an EMR cluster into a VPC to support the EMR jobs. However, the deployed Step Functions state machine is not able to run the EMR jobs.

Which combination of steps should the company take to identify the reason the Step Functions state machine is not able to run the EMR jobs? (Choose two.)

A. Use AWS CloudFormation to automate the Step Functions state machine deployment. Create a step to pause the state machine during the EMR jobs that fail. Configure the step to wait for a human user to send approval through an email message. Include details of the EMR task in the email message for further analysis.

B. Verify that the Step Functions state machine code has all IAM permissions that are necessary to create and run the EMR jobs. Verify that the Step Functions state machine code also includes IAM permissions to access the Amazon S3 buckets that the EMR jobs use. Use Access Analyzer for S3 to check the S3 access properties.

C. Check for entries in Amazon CloudWatch for the newly created EMR cluster. Change the AWS Step Functions state machine code to use Amazon EMR on EKS. Change the IAM access policies and the security group configuration for the Step Functions state machine code to reflect inclusion of Amazon Elastic Kubernetes Service (Amazon EKS).

D. Query the flow logs for the VPC. Determine whether the traffic that originates from the EMR cluster can successfully reach the data providers. Determine whether any security group that might be attached to the Amazon EMR cluster allows connections to the data source servers on the informed ports.

E. Check the retry scenarios that the company configured for the EMR jobs. Increase the number of seconds in the interval between each EMR task. Validate that each fallback state has the appropriate catch for each decision state. Configure an Amazon Simple Notification Service (Amazon SNS) topic to store the error messages.

Answer: B D

Explanation:

To identify the reason why the Step Functions state machine is not able to run the EMR jobs, the company should take the following steps:

Verify that the Step Functions state machine code has all IAM permissions that are necessary to create and run the EMR jobs. The state machine code should have an IAM role that allows it to invoke the EMR APIs, such as RunJobFlow, AddJobFlowSteps, and DescribeStep. The state machine code should also have IAM permissions to access the Amazon S3 buckets that the EMR jobs use as input and output locations. The company can use Access Analyzer for S3 to check the access policies and permissions of the S3 buckets12.

Therefore, option B is correct.

Query the flow logs for the VPC. The flow logs can provide information about the network traffic to and from the EMR cluster that is launched in the VPC. The company can use the flow logs to determine whether the traffic that originates from the EMR cluster can successfully reach the data providers, such as Amazon

RDS, Amazon Redshift, or other external sources. The company can also determine whether any security group that might be attached to the EMR cluster allows connections to the data source servers on the informed ports. The company can use Amazon VPC Flow Logs or Amazon CloudWatch Logs Insights to query the flow logs3 . Therefore, option D is correct.

Option A is incorrect because it suggests using AWS CloudFormation to automate the Step Functions state machine deployment. While this is a good practice to ensure consistency and repeatability of the deployment, it does not help to identify the reason why the state machine is not able to run the EMR jobs. Moreover, creating a step to pause the state machine during the EMR jobs that fail and wait for a human user to send approval through an email message is not a reliable way to troubleshoot the issue. The company should use the Step Functions console or API to monitor the execution history and status of the state machine, and use Amazon CloudWatch to view the logs and metrics of the EMR jobs .

Option C is incorrect because it suggests changing the AWS Step Functions state machine code to use Amazon EMR on EKS. Amazon EMR on EKS is a service that allows you to run EMR jobs on Amazon Elastic Kubernetes Service (Amazon EKS) clusters. While this service has some benefits, such as lower cost and faster execution time, it does not support all the features and integrations that EMR on EC2 does, such as EMR Notebooks, EMR Studio, and EMRFS. Therefore, changing the state machine code to use EMR on EKS may not be compatible with the existing data pipeline and may introduce new issues.

Option E is incorrect because it suggests checking the retry scenarios that the company configured for the EMR jobs. While this is a good practice to handle transient failures and errors, it does not help to identify the root cause of why the state machine is not able to run the EMR jobs. Moreover, increasing the number of seconds in the interval between each EMR task may not improve the success rate of the jobs, and may increase the execution time and cost of the state machine. Configuring an Amazon SNS topic to store the error messages may help to notify the company of any failures, but it does not provide enough information to troubleshoot the issue.

Reference:

1: Manage an Amazon EMR Job - AWS Step Functions
2: Access Analyzer for S3 - Amazon Simple Storage Service
3: Working with Amazon EMR and VPC Flow Logs - Amazon EMR
[4] : Analyzing VPC Flow Logs with Amazon CloudWatch Logs Insights - Amazon Virtual Private Cloud
[5] : Monitor AWS Step Functions - AWS Step Functions
[6] : Monitor Amazon EMR clusters - Amazon EMR
[7] : Amazon EMR on Amazon EKS - Amazon EMR

Question: 44 A company is developing an application that runs on Amazon EC2 instances. Currently, the data that the application generates is temporary. However, the company needs to persist the data, even if the EC2 instances are terminated.

A data engineer must launch new EC2 instances from an Amazon Machine Image (AMI) and configure the instances to preserve the data.

Which solution will meet this requirement?

A. Launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume that contains the application data. Apply the default settings to the EC2 instances.

B. Launch new EC2 instances by using an AMI that is backed by a root Amazon Elastic Block Store (Amazon EBS) volume that contains the application data. Apply the default settings to the EC2 instances.

C. Launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume. Attach an Amazon Elastic Block Store (Amazon EBS) volume to contain the application data. Apply the default settings to the EC2 instances.

D. Launch new EC2 instances by using an AMI that is backed by an Amazon Elastic Block Store

(Amazon EBS) volume. Attach an additional EC2 instance store volume to contain the application data. Apply the default settings to the EC2 instances.

Answer: C

Explanation:

Amazon EC2 instances can use two types of storage volumes: instance store volumes and Amazon EBS volumes. Instance store volumes are ephemeral, meaning they are only attached to the instance for the duration of its life cycle. If the instance is stopped, terminated, or fails, the data on the instance store volume is lost. Amazon EBS volumes are persistent, meaning they can be detached from the instance and attached to another instance, and the data on the volume is preserved. To meet the requirement of persisting the data even if the EC2 instances are terminated, the data engineer must use Amazon EBS volumes to store the application data. The solution is to launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume, which is the default option for most AMIs. Then, the data engineer must attach an Amazon EBS volume to each instance and configure the application to write the data to the EBS volume. This way, the data will be saved on the EBS volume and can be accessed by another instance if needed. The data engineer can apply the default settings to the EC2 instances, as there is no need to modify the instance type, security group, or IAM role for this solution. The other options are either not feasible or not optimal. Launching new EC2 instances by using an AMI that is backed by an EC2 instance store volume that contains the application data (option A) or by using an AMI that is backed by a root Amazon EBS volume that contains the application data (option B) would not work, as the data on the AMI would be outdated and overwritten by the new instances. Attaching an additional EC2 instance store volume to contain the application data (option D) would not work, as the data on the instance store volume would be lost if the instance is terminated. Reference:

Amazon EC2 Instance Store

Amazon EBS Volumes

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 2: Data Store Management, Section 2.1: Amazon EC2

Question: 45 A company uses Amazon Athena to run SQL queries for extract, transform, and load (ETL) tasks by using Create Table As Select (CTAS). The company must use Apache Spark instead of SQL to generate analytics.

Which solution will give the company the ability to use Spark to access Athena?

A. Athena query settings
B. Athena workgroup
C. Athena data source
D. Athena query editor

Answer: C

Explanation:

Athena data source is a solution that allows you to use Spark to access Athena by using the Athena JDBC driver and the Spark SQL interface. You can use the Athena data source to create Spark DataFrames from Athena tables, run SQL queries on the DataFrames, and write the results back to Athena. The Athena data source supports various data formats, such as CSV, JSON, ORC, and Parquet, and also supports partitioned and bucketed tables. The Athena data source is a cost-effective and scalable way to use Spark to access Athena, as it does not require any additional infrastructure or services, and you only pay for the data scanned by Athena.

The other options are not solutions that give the company the ability to use Spark to access Athena. Option A,

Athena query settings, is a feature that allows you to configure various parameters for your Athena queries, such as the output location, the encryption settings, the query timeout, and the workgroup. Option B, Athena workgroup, is a feature that allows you to isolate and manage your Athena queries and resources, such as the query history, the query notifications, the query concurrency, and the query cost. Option D, Athena query editor, is a feature that allows you to write and run SQL queries on Athena using the web console or the API. None of these options enable you to use Spark instead of SQL to generate analytics on Athena. Reference:

Using Apache Spark in Amazon Athena

Athena JDBC Driver

Spark SQL

Athena query settings [Athena workgroups] [Athena query editor]

Question: 46

A company needs to partition the Amazon S3 storage that the company uses for a data lake. The partitioning will use a path of the S3 object keys in the following format: s3://bucket/prefix/year=2023/month=01/day=01.

A data engineer must ensure that the AWS Glue Data Catalog synchronizes with the S3 storage when the company adds new partitions to the bucket.

Which solution will meet these requirements with the LEAST latency?

A. Schedule an AWS Glue crawler to run every morning.
B. Manually run the AWS Glue CreatePartition API twice each day.
C. Use code that writes data to Amazon S3 to invoke the Boto3 AWS Glue create partition API call.
D. Run the MSCK REPAIR TABLE command from the AWS Glue console.

Answer: A

Explanation:

The best solution to ensure that the AWS Glue Data Catalog synchronizes with the S3 storage when the company adds new partitions to the bucket with the least latency is to use code that writes data to Amazon S3 to invoke the Boto3 AWS Glue create partition API call. This way, the Data Catalog is updated as soon as new data is written to S3, and the partition information is immediately available for querying by other services. The Boto3 AWS Glue create partition API call allows you to create a new partition in the Data Catalog by specifying the table name, the database name, and the partition values1. You can use this API call in your code that writes data to S3, such as a Python script or an AWS Glue ETL job, to create a partition for each new S3 object key that matches the partitioning scheme.

Option A is not the best solution, as scheduling an AWS Glue crawler to run every morning would introduce a significant latency between the time new data is written to S3 and the time the Data Catalog is updated. AWS Glue crawlers are processes that connect to a data store, progress through a prioritized list of classifiers to determine the schema for your data, and then create metadata tables in the Data Catalog2. Crawlers can be scheduled to run periodically, such as daily or hourly, but they cannot run continuously or in real-time. Therefore, using a crawler to synchronize the Data Catalog with the S3 storage would not meet the requirement of the least latency.

Option B is not the best solution, as manually running the AWS Glue CreatePartition API twice each day would also introduce a significant latency between the time new data is written to S3 and the time the Data Catalog is updated. Moreover, manually running the API would require more operational overhead and human intervention than using code that writes data to S3 to invoke the API automatically.

Option D is not the best solution, as running the MSCK REPAIR TABLE command from the AWS Glue console would also introduce a significant latency between the time new data is written to S3 and the time the Data Catalog is updated. The MSCK REPAIR TABLE command is a SQL command that you can run in the AWS Glue console to add partitions to the Data Catalog based on the S3 object keys that match the partitioning scheme3. However, this command is not meant to be run frequently or in real-time, as it can take a long time to scan the entire S3 bucket and add the partitions. Therefore, using this command to synchronize the Data Catalog with the S3 storage would not meet the requirement of the least latency. Reference:

AWS Glue CreatePartition API
Populating the AWS Glue Data Catalog
MSCK REPAIR TABLE Command
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

Question: 47

A media company uses software as a service (SaaS) applications to gather data by using third-party tools. The company needs to store the data in an Amazon S3 bucket. The company will use Amazon Redshift to perform analytics based on the data.
Which AWS service or feature will meet these requirements with the LEAST operational overhead?

A. Amazon Managed Streaming for Apache Kafka (Amazon MSK)
B. Amazon AppFlow
C. AWS Glue Data Catalog
D. Amazon Kinesis

Answer: B

Explanation:
Amazon AppFlow is a fully managed integration service that enables you to securely transfer data between SaaS applications and AWS services like Amazon S3 and Amazon Redshift. Amazon AppFlow supports many SaaS applications as data sources and targets, and allows you to configure data flows with a few clicks. Amazon AppFlow also provides features such as data transformation, filtering, validation, and encryption to prepare and protect your data. Amazon AppFlow meets the requirements of the media company with the least operational overhead, as it eliminates the need to write code, manage infrastructure, or monitor data pipelines. Reference:

Amazon AppFlow
Amazon AppFlow | SaaS Integrations List
Get started with data integration from Amazon S3 to Amazon Redshift using AWS Glue interactive sessions

Question: 48

A data engineer is using Amazon Athena to analyze sales data that is in Amazon S3. The data engineer writes a query to retrieve sales amounts for 2023 for several products from a table named sales_data. However, the query does not return results for all of the products that are in the sales_data table. The data engineer needs to troubleshoot the query to resolve the issue.
The data engineer's original query is as follows: SELECT product_name, sum(sales_amount)
FROM sales_data
WHERE year = 2023

GROUP BY product_name

How should the data engineer modify the Athena query to meet these requirements?

A. Replace sum(sales amount) with count(*J for the aggregation.

B. Change WHERE year = 2023 to WHERE extractlyear FROM sales data) = 2023.

C. Add HAVING sumfsales amount) > 0 after the GROUP BY clause.

D. Remove the GROUP BY clause

Answer: B

Explanation:

The original query does not return results for all of the products because the year column in the sales_data table is not an integer, but a timestamp. Therefore, the WHERE clause does not filter the data correctly, and only returns the products that have a null value for the year column. To fix this, the data engineer should use the extract function to extract the year from the timestamp and compare it with 2023. This way, the query will return the correct results for all of the products in the sales_data table. The other options are either incorrect or irrelevant, as they do not address the root cause of the issue. Replacing sum with count does not change the filtering condition, adding HAVING clause does not affect the grouping logic, and removing the GROUP BY clause does not solve the problem of missing products. Reference:

Troubleshooting JSON queries - Amazon Athena (Section: JSON related errors)

When I query a table in Amazon Athena, the TIMESTAMP result is empty (Section: Resolution) AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 7, page 197)

Question: 49

A data engineer has a one-time task to read data from objects that are in Apache Parquet format in an Amazon S3 bucket. The data engineer needs to query only one column of the data.

Which solution will meet these requirements with the LEAST operational overhead?

A. Confiqure an AWS Lambda function to load data from the S3 bucket into a pandas dataframe- Write a SQL SELECT statement on the dataframe to query the required column.

B. Use S3 Select to write a SQL SELECT statement to retrieve the required column from the S3 objects.

C. Prepare an AWS Glue DataBrew project to consume the S3 objects and to query the required column.

D. Run an AWS Glue crawler on the S3 objects. Use a SQL SELECT statement in Amazon Athena to query the required column.

Answer: B

Explanation:

Option B is the best solution to meet the requirements with the least operational overhead because S3 Select is a feature that allows you to retrieve only a subset of data from an S3 object by using simple SQL expressions. S3 Select works on objects stored in CSV, JSON, or Parquet format. By using S3 Select, you can avoid the need to download and process the entire S3 object, which reduces the amount of data transferred and the computation time. S3 Select is also easy to use and does not require any additional services or resources. Option A is not a good solution because it involves writing custom code and configuring an AWS Lambda function to load data from the S3 bucket into a pandas dataframe and query the required column. This option adds complexity and latency to the data retrieval process and requires additional resources and configuration. Moreover, AWS Lambda has limitations on the execution time, memory, and concurrency, which may affect the

performance and reliability of the data retrieval process. Option C is not a good solution because it involves creating and running an AWS Glue DataBrew project to consume the S3 objects and query the required column. AWS Glue DataBrew is a visual data preparation tool that allows you to clean, normalize, and transform data without writing code. However, in this scenario, the data is already in Parquet format, which is a columnar storage format that is optimized for analytics. Therefore, there is no need to use AWS Glue DataBrew to prepare the data. Moreover, AWS Glue DataBrew adds extra time and cost to the data retrieval process and requires additional resources and configuration.

Option D is not a good solution because it involves running an AWS Glue crawler on the S3 objects and using a SQL SELECT statement in Amazon Athena to query the required column. An AWS Glue crawler is a service that can scan data sources and create metadata tables in the AWS Glue Data Catalog. The Data Catalog is a central repository that stores information about the data sources, such as schema, format, and location. Amazon Athena is a serverless interactive query service that allows you to analyze data in S3 using standard SQL. However, in this scenario, the schema and format of the data are already known and fixed, so there is no need to run a crawler to discover them.

Moreover, running a crawler and using Amazon Athena adds extra time and cost to the data retrieval process and requires additional services and configuration.

Reference:

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

S3 Select and Glacier Select - Amazon Simple Storage Service

AWS Lambda - FAQs

What Is AWS Glue DataBrew? - AWS Glue DataBrew

Populating the AWS Glue Data Catalog - AWS Glue

What is Amazon Athena? - Amazon Athena

Question: 50

A company uses Amazon Redshift for its data warehouse. The company must automate refresh schedules for Amazon Redshift materialized views.

Which solution will meet this requirement with the LEAST effort?

A. Use Apache Airflow to refresh the materialized views.

B. Use an AWS Lambda user-defined function (UDF) within Amazon Redshift to refresh the materialized views.

C. Use the query editor v2 in Amazon Redshift to refresh the materialized views.

D. Use an AWS Glue workflow to refresh the materialized views.

Answer: B

Explanation:

The query editor v2 in Amazon Redshift is a web-based tool that allows users to run SQL queries and scripts on Amazon Redshift clusters. The query editor v2 supports creating and managing materialized views, which are precomputed results of a query that can improve the performance of subsequent queries. The query editor v2 also supports scheduling queries to run at specified intervals, which can be used to refresh materialized views automatically. This solution requires the least effort, as it does not involve any additional services, coding, or configuration. The other solutions are more complex and require more operational overhead. Apache Airflow is an opensource platform for orchestrating workflows, which can be used to refresh materialized views, but it requires setting up and managing an Airflow environment, creating DAGs (directed acyclic graphs) to define the workflows, and integrating with Amazon Redshift. AWS Lambda is a serverless compute service that can run code in response

to events, which can be used to refresh materialized views, but it requires creating and deploying Lambda functions, defining UDFs within Amazon Redshift, and triggering the functions using events or schedules. AWS Glue is a fully managed ETL service that can run jobs to transform and load data, which can be used to refresh materialized views, but it requires creating and configuring Glue jobs, defining Glue workflows to orchestrate the jobs, and scheduling the workflows using triggers. Reference:

Query editor V2

Working with materialized views

Scheduling queries

[AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

Question: 51 A data engineer must orchestrate a data pipeline that consists of one AWS Lambda function and one AWS Glue job. The solution must integrate with AWS services.

Which solution will meet these requirements with the LEAST management overhead?

A. Use an AWS Step Functions workflow that includes a state machine. Configure the state machine to run the Lambda function and then the AWS Glue job.

B. Use an Apache Airflow workflow that is deployed on an Amazon EC2 instance. Define a directed acyclic graph (DAG) in which the first task is to call the Lambda function and the second task is to call the AWS Glue job.

C. Use an AWS Glue workflow to run the Lambda function and then the AWS Glue job.

D. Use an Apache Airflow workflow that is deployed on Amazon Elastic Kubernetes Service (Amazon EKS). Define a directed acyclic graph (DAG) in which the first task is to call the Lambda function and the second task is to call the AWS Glue job.

Answer: A

Explanation:

AWS Step Functions is a service that allows you to coordinate multiple AWS services into serverless workflows. You can use Step Functions to create state machines that define the sequence and logic of the tasks in your workflow. Step Functions supports various types of tasks, such as Lambda functions, AWS Glue jobs, Amazon EMR clusters, Amazon ECS tasks, etc. You can use Step Functions to monitor and troubleshoot your workflows, as well as to handle errors and retries.

Using an AWS Step Functions workflow that includes a state machine to run the Lambda function and then the AWS Glue job will meet the requirements with the least management overhead, as it leverages the serverless and managed capabilities of Step Functions. You do not need to write any code to orchestrate the tasks in your workflow, as you can use the Step Functions console or the AWS Serverless Application Model (AWS SAM) to define and deploy your state machine. You also do not need to provision or manage any servers or clusters, as Step Functions scales automatically based on the demand.

The other options are not as efficient as using an AWS Step Functions workflow. Using an Apache Airflow workflow that is deployed on an Amazon EC2 instance or on Amazon Elastic Kubernetes Service (Amazon EKS) will require more management overhead, as you will need to provision, configure, and maintain the EC2 instance or the EKS cluster, as well as the Airflow components. You will also need to write and maintain the Airflow DAGs to orchestrate the tasks in your workflow. Using an AWS Glue workflow to run the Lambda function and then the AWS Glue job will not work, as AWS Glue workflows only support AWS Glue jobs and crawlers as tasks, not Lambda functions. Reference: AWS Step Functions

---

AWS Glue

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 6: Data Integration and Transformation, Section 6.3: AWS Step Functions

www.atmicnetworks.com

Question: 52

A company needs to set up a data catalog and metadata management for data sources that run in the AWS Cloud. The company will use the data catalog to maintain the metadata of all the objects that are in a set of data stores. The data stores include structured sources such as Amazon RDS and

Amazon Redshift. The data stores also include semistructured sources such as JSON files and .xml files that are stored in Amazon S3.

The company needs a solution that will update the data catalog on a regular basis. The solution also must detect changes to the source metadata.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use Amazon Aurora as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the Aurora data catalog. Schedule the Lambda functions to run periodically.

B. Use the AWS Glue Data Catalog as the central metadata repository. Use AWS Glue crawlers to connect to multiple data stores and to update the Data Catalog with metadata changes. Schedule the crawlers to run periodically to update the metadata catalog.

C. Use Amazon DynamoDB as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the DynamoDB data catalog. Schedule the Lambda functions to run periodically.

D. Use the AWS Glue Data Catalog as the central metadata repository. Extract the schema for Amazon RDS and Amazon Redshift sources, and build the Data Catalog. Use AWS Glue crawlers for data that is in Amazon S3 to infer the schema and to automatically update the Data Catalog.

Answer: B

Explanation:

This solution will meet the requirements with the least operational overhead because it uses the AWS Glue Data Catalog as the central metadata repository for data sources that run in the AWS Cloud. The AWS Glue Data Catalog is a fully managed service that provides a unified view of your data assets across AWS and on-premises data sources. It stores the metadata of your data in tables, partitions, and columns, and enables you to access and query your data using various AWS services, such as Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum. You can use AWS Glue crawlers to connect to multiple data stores, such as Amazon RDS, Amazon Redshift, and Amazon S3, and to update the Data Catalog with metadata changes. AWS Glue crawlers can automatically discover the schema and partition structure of your data, and create or update the corresponding tables in the Data Catalog. You can schedule the crawlers to run periodically to update the metadata catalog, and configure them to detect changes to the source metadata, such as new columns, tables, or partitions12.

The other options are not optimal for the following reasons:

A . Use Amazon Aurora as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the Aurora data catalog. Schedule the Lambda functions to run periodically. This option is not recommended, as it would require more operational overhead to create and manage an Amazon Aurora database as the data catalog, and to write and maintain AWS Lambda functions to gather and update the metadata information from multiple sources. Moreover, this option would not leverage the benefits of the AWS Glue Data Catalog, such as data cataloging, data transformation, and data governance.

C . Use Amazon DynamoDB as the data catalog. Create AWS Lambda functions that will connect to the data catalog. Configure the Lambda functions to gather the metadata information from multiple sources and to update the DynamoDB data catalog. Schedule the Lambda functions to run periodically. This option is also not

recommended, as it would require more operational overhead to create and manage an Amazon DynamoDB table as the data catalog, and to write and maintain AWS Lambda functions to gather and update the metadata information from multiple sources. Moreover, this option would not leverage the benefits of the AWS Glue Data Catalog, such as data cataloging, data transformation, and data governance.

D . Use the AWS Glue Data Catalog as the central metadata repository. Extract the schema for Amazon RDS and Amazon Redshift sources, and build the Data Catalog. Use AWS Glue crawlers for data that is in Amazon S3 to infer the schema and to automatically update the Data Catalog. This option is not optimal, as it would require more manual effort to extract the schema for Amazon RDS and Amazon Redshift sources, and to build the Data Catalog. This option would not take advantage of the AWS Glue crawlers' ability to automatically discover the schema and partition structure of your data from various data sources, and to create or update the corresponding tables in the Data Catalog. Reference:

1: AWS Glue Data Catalog

2: AWS Glue Crawlers

: Amazon Aurora

: AWS Lambda

: Amazon DynamoDB

Question: 53

A company stores data from an application in an Amazon DynamoDB table that operates in provisioned capacity mode. The workloads of the application have predictable throughput load on a regular schedule. Every Monday, there is an immediate increase in activity early in the morning. The application has very low usage during weekends.

The company must ensure that the application performs consistently during peak usage times. Which solution will meet these requirements in the MOST cost-effective way?

A. Increase the provisioned capacity to the maximum capacity that is currently present during peak load times.

B. Divide the table into two tables. Provision each table with half of the provisioned capacity of the original table. Spread queries evenly across both tables.

C. Use AWS Application Auto Scaling to schedule higher provisioned capacity for peak usage times. Schedule lower capacity during off-peak times.

D. Change the capacity mode from provisioned to on-demand. Configure the table to scale up and scale down based on the load on the table.

Answer: C

Explanation:

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB offers two capacity modes for throughput capacity: provisioned and on-demand. In provisioned capacity mode, you specify the number of read and write capacity units per second that you expect your application to require. DynamoDB reserves the resources to meet your throughput needs with consistent performance. In on-demand capacity mode, you pay per request and DynamoDB scales the resources up and down automatically based on the actual workload. On-demand capacity mode is suitable for unpredictable workloads that can vary significantly over time1.

The solution that meets the requirements in the most cost-effective way is to use AWS Application Auto Scaling to schedule higher provisioned capacity for peak usage times and lower capacity during off-peak times. This solution has the following advantages:

It allows you to optimize the cost and performance of your DynamoDB table by adjusting the provisioned capacity according to your predictable workload patterns. You can use scheduled scaling to specify the date and time for the scaling actions, and the new minimum and maximum capacity limits. For example, you can schedule higher capacity for every Monday morning and lower capacity for weekends2.

It enables you to take advantage of the lower cost per unit of provisioned capacity mode compared to on-demand capacity mode. Provisioned capacity mode charges a flat hourly rate for the capacity you reserve, regardless of how much you use. On-demand capacity mode charges for each read and write request you consume, with no minimum capacity required. For predictable workloads, provisioned capacity mode can be more cost-effective than on-demand capacity mode1.

It ensures that your application performs consistently during peak usage times by having enough capacity to handle the increased load. You can also use auto scaling to automatically adjust the provisioned capacity based on the actual utilization of your table, and set a target utilization percentage for your table or global secondary index. This way, you can avoid under-provisioning or over-provisioning your table2.

Option A is incorrect because it suggests increasing the provisioned capacity to the maximum capacity that is currently present during peak load times. This solution has the following disadvantages:

It wastes money by paying for unused capacity during off-peak times. If you provision the same high capacity for all times, regardless of the actual workload, you are over-provisioning your table and paying for resources that you don't needl.

It does not account for possible changes in the workload patterns over time. If your peak load times increase or decrease in the future, you may need to manually adjust the provisioned capacity to match the new demand. This adds operational overhead and complexity to your application2. Option B is incorrect because it suggests dividing the table into two tables and provisioning each table with half of the provisioned capacity of the original table. This solution has the following disadvantages:

It complicates the data model and the application logic by splitting the data into two separate tables. You need to ensure that the queries are evenly distributed across both tables, and that the data is consistent and synchronized between them. This adds extra development and maintenance effort to your application3.

It does not solve the problem of adjusting the provisioned capacity according to the workload patterns. You still need to manually or automatically scale the capacity of each table based on the actual utilization and demand. This may result in under-provisioning or over-provisioning your tables2.

Option D is incorrect because it suggests changing the capacity mode from provisioned to on- demand. This solution has the following disadvantages:

It may incur higher costs than provisioned capacity mode for predictable workloads. On-demand capacity mode charges for each read and write request you consume, with no minimum capacity required. For predictable workloads, provisioned capacity mode can be more cost-effective than on- demand capacity mode, as you can reserve the capacity you need at a lower rate1.

It may not provide consistent performance during peak usage times, as on-demand capacity mode may take some time to scale up the resources to meet the sudden increase in demand. On-demand capacity mode uses adaptive capacity to handle bursts of traffic, but it may not be able to handle very large spikes or sustained high throughput. In such cases, you may experience throttling or increased latency.

Reference:

1: Choosing the right DynamoDB capacity mode - Amazon DynamoDB
2: Managing throughput capacity automatically with DynamoDB auto scaling - Amazon DynamoDB
3: Best practices for designing and using partition keys effectively - Amazon DynamoDB
[4] : On-demand mode guidelines - Amazon DynamoDB
[5] : How to optimize Amazon DynamoDB costs - AWS Database Blog
[6] : DynamoDB adaptive capacity: How it works and how it helps - AWS Database Blog
[7] : Amazon DynamoDB pricing - Amazon Web Services (AWS)

www.atmicnetworks.com

Question: 54

A company is planning to migrate on-premises Apache Hadoop clusters to Amazon EMR. The company also needs to migrate a data catalog into a persistent storage solution.

The company currently stores the data catalog in an on-premises Apache Hive metastore on the Hadoop clusters. The company requires a serverless solution to migrate the data catalog. Which solution will meet these requirements MOST cost-effectively?

A. Use AWS Database Migration Service (AWS DMS) to migrate the Hive metastore into Amazon S3. Configure AWS Glue Data Catalog to scan Amazon S3 to produce the data catalog.

B. Configure a Hive metastore in Amazon EMR. Migrate the existing on-premises Hive metastore into Amazon EMR. Use AWS Glue Data Catalog to store the company's data catalog as an external data catalog.

C. Configure an external Hive metastore in Amazon EMR. Migrate the existing on-premises Hive metastore into Amazon EMR. Use Amazon Aurora MySQL to store the company's data catalog.

D. Configure a new Hive metastore in Amazon EMR. Migrate the existing on-premises Hive metastore into Amazon EMR. Use the new metastore as the company's data catalog.

Answer: A

Explanation:

AWS Database Migration Service (AWS DMS) is a service that helps you migrate databases to AWS quickly and securely. You can use AWS DMS to migrate the Hive metastore from the on-premises Hadoop clusters into Amazon S3, which is a highly scalable, durable, and cost-effective object storage service. AWS Glue Data Catalog is a serverless, managed service that acts as a central metadata repository for your data assets. You can use AWS Glue Data Catalog to scan the Amazon S3 bucket that contains the migrated Hive metastore and create a data catalog that is compatible with Apache Hive and other AWS services. This solution meets the requirements of migrating the data catalog into a persistent storage solution and using a serverless solution. This solution is also the most costeffective, as it does not incur any additional charges for running Amazon EMR or Amazon Aurora MySQL clusters. The other options are either not feasible or not optimal. Configuring a Hive metastore in Amazon EMR (option B) or an external Hive metastore in Amazon EMR (option C) would require running and maintaining Amazon EMR clusters, which would incur additional costs and complexity. Using Amazon Aurora MySQL to store the company's data catalog (option C) would also incur additional costs and complexity, as well as introduce compatibility issues with Apache Hive. Configuring a new Hive metastore in Amazon EMR (option D) would not migrate the existing data catalog, but create a new one, which would result in data loss and inconsistency.

Reference: Using AWS Database Migration Service Populating the AWS Glue Data Catalog

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Analysis and Visualization, Section 4.2: AWS Glue Data Catalog

Question: 55

A company uses an Amazon Redshift provisioned cluster as its database. The Redshift cluster has five reserved ra3.4xlarge nodes and uses key distribution.

A data engineer notices that one of the nodes frequently has a CPU load over 90%. SQL Queries that run on the node are queued. The other four nodes usually have a CPU load under 15% during daily operations.

The data engineer wants to maintain the current number of compute nodes. The data engineer also wants to balance the load more evenly across all five compute nodes.

Which solution will meet these requirements?

A. Change the sort key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement.

B. Change the distribution key to the table column that has the largest dimension.

C. Upgrade the reserved node from ra3.4xlarqe to ra3.16xlarqe.

D. Change the primary key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement.

Answer: B

Explanation:

Changing the distribution key to the table column that has the largest dimension will help to balance the load more evenly across all five compute nodes. The distribution key determines how the rows of a table are distributed among the slices of the cluster. If the distribution key is not chosen wisely, it can cause data skew, meaning some slices will have more data than others, resulting in uneven CPU load and query performance. By choosing the table column that has the largest dimension, meaning the column that has the most distinct values, as the distribution key, the data engineer can ensure that the rows are distributed more uniformly across the slices, reducing data skew and improving query performance.

The other options are not solutions that will meet the requirements. Option A, changing the sort key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement, will not affect the data distribution or the CPU load. The sort key determines the order in which the rows of a table are stored on disk, which can improve the performance of range-restricted queries, but not the load balancing. Option C, upgrading the reserved node from ra3.4xlarge to ra3.16xlarge, will not maintain the current number of compute nodes, as it will increase the cost and the capacity of the cluster. Option D, changing the primary key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement, will not affect the data distribution or the CPU load either. The primary key is a constraint that enforces the uniqueness of the rows in a table, but it does not influence the data layout or the query optimization. Reference:

Choosing a data distribution style Choosing a data sort key Working with primary keys

---

Question: 56

A security company stores IoT data that is in JSON format in an Amazon S3 bucket. The data structure can change when the company upgrades the IoT devices. The company wants to create a data catalog that includes the IoT dat

a. The company's analytics department will use the data catalog to index the data.

Which solution will meet these requirements MOST cost-effectively?

A. Create an AWS Glue Data Catalog. Configure an AWS Glue Schema Registry. Create a new AWS Glue workload to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.

B. Create an Amazon Redshift provisioned cluster. Create an Amazon Redshift Spectrum database for the analytics department to explore the data that is in Amazon S3. Create Redshift stored procedures to load the data into Amazon Redshift.

C. Create an Amazon Athena workgroup. Explore the data that is in Amazon S3 by using Apache Spark through Athena. Provide the Athena workgroup schema and tables to the analytics department.

D. Create an AWS Glue Data Catalog. Configure an AWS Glue Schema Registry. Create AWS Lambda user defined functions (UDFs) by using the Amazon Redshift Data API. Create an AWS Step Functions job to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless.

Answer: C

Explanation:

The best solution to meet the requirements of creating a data catalog that includes the IoT data, and allowing the analytics department to index the data, most cost-effectively, is to create an Amazon Athena workgroup, explore the data that is in Amazon S3 by using Apache Spark through Athena, and provide the Athena workgroup schema and tables to the analytics department.

Amazon Athena is a serverless, interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL or Python1. Amazon Athena also supports Apache Spark, an opensource distributed processing framework that can run large-scale data analytics applications across clusters of servers2. You can use Athena to run Spark code on data in Amazon S3 without having to set up, manage, or scale any infrastructure. You can also use Athena to create and manage external tables that point to your data in Amazon S3, and store them in an external data catalog, such as AWS Glue Data Catalog, Amazon Athena Data Catalog, or your own Apache Hive metastore3. You can create Athena workgroups to separate query execution and resource allocation based on different criteria, such as users, teams, or applications4. You can share the schemas and tables in your Athena workgroup with other users or applications, such as Amazon QuickSight, for data visualization and analysis5. Using Athena and Spark to create a data catalog and explore the IoT data in Amazon S3 is the most cost-effective solution, as you pay only for the queries you run or the compute you use, and you pay nothing when the service is idle1. You also save on the operational overhead and complexity of managing data warehouse infrastructure, as Athena and Spark are serverless and scalable. You can also benefit from the flexibility and performance of Athena and Spark, as they support various data formats, including JSON, and can handle schema changes and complex queries efficiently.

Option A is not the best solution, as creating an AWS Glue Data Catalog, configuring an AWS Glue Schema Registry, creating a new AWS Glue workload to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless, would incur more costs and complexity than using Athena and Spark. AWS Glue Data Catalog is a persistent metadata store that contains table definitions, job definitions, and other control information to help you manage your AWS Glue components6. AWS Glue Schema Registry is a service that allows you to centrally store and manage the schemas of your streaming data in AWS Glue Data Catalog7. AWS Glue is a serverless data integration service that makes it easy to prepare, clean, enrich, and move data between data stores8. Amazon Redshift Serverless is a feature of Amazon Redshift, a fully managed data warehouse service, that allows you to run and scale analytics without having to manage data warehouse infrastructure9. While these services are powerful and useful for many data engineering scenarios, they are not necessary or cost-effective for creating a data catalog and indexing the IoT data in Amazon S3. AWS Glue Data Catalog and Schema Registry charge you based on the number of objects stored and the number of requests made67. AWS Glue charges you based on the compute time and the data processed by your ETL jobs8. Amazon Redshift Serverless charges you based on the amount of data scanned by your queries and the compute time used by your workloads9. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using AWS Glue and Amazon Redshift Serverless would introduce additional latency and complexity, as you would have to ingest the data from Amazon S3 to Amazon Redshift Serverless, and then query it from there, instead of querying it directly from Amazon S3 using Athena and Spark.

Option B is not the best solution, as creating an Amazon Redshift provisioned cluster, creating an Amazon Redshift Spectrum database for the analytics department to explore the data that is in Amazon S3, and creating Redshift stored procedures to load the data into Amazon Redshift, would incur more costs and complexity than using Athena and Spark. Amazon Redshift provisioned clusters are clusters that you create and manage by specifying the number and type of nodes, and the amount of storage and compute capacity10. Amazon Redshift Spectrum is a feature of Amazon Redshift that allows you to query and join data across your data warehouse and your data lake using standard SQL11. Redshift stored procedures are SQL statements that you can define and store in Amazon Redshift, and then call them by using the CALL command12. While these features are powerful and useful for many data warehousing scenarios, they are not necessary or cost-effective for creating a data catalog and indexing

the IoT data in Amazon S3. Amazon Redshift provisioned clusters charge you based on the node type, the number of nodes, and the duration of the cluster10. Amazon Redshift Spectrum charges you based on the amount of data scanned by your queries11. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using Amazon Redshift provisioned clusters and Spectrum would introduce additional latency and complexity, as you would have to provision and manage the cluster, create an external schema and database for the data in Amazon S3, and load the data into the cluster using stored procedures, instead of querying it directly from Amazon S3 using Athena and Spark.

Option D is not the best solution, as creating an AWS Glue Data Catalog, configuring an AWS Glue Schema Registry, creating AWS Lambda user defined functions (UDFs) by using the Amazon Redshift Data API, and creating an AWS Step Functions job to orchestrate the ingestion of the data that the analytics department will use into Amazon Redshift Serverless, would incur more costs and complexity than using Athena and Spark. AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers13. AWS Lambda UDFs are Lambda functions that you can invoke from within an Amazon Redshift query. Amazon Redshift Data API is a service that allows you to run SQL statements on Amazon Redshift clusters using HTTP requests, without needing a persistent connection. AWS Step Functions is a service that lets you coordinate multiple AWS services into serverless workflows. While these services are powerful and useful for many data engineering scenarios, they are not necessary or cost-effective for creating a data catalog and indexing the IoT data in Amazon S3. AWS Glue Data Catalog and Schema Registry charge you based on the number of objects stored and the number of requests made67. AWS Lambda charges you based on the number of requests and the duration of your functions13. Amazon Redshift Serverless charges you based on the amount of data scanned by your queries and the compute time used by your workloads9. AWS Step Functions charges you based on the number of state transitions in your workflows. These costs can add up quickly, especially if you have large volumes of IoT data and frequent schema changes. Moreover, using AWS Glue, AWS Lambda, Amazon Redshift Data API, and AWS Step Functions would introduce additional latency and complexity, as you would have to create and invoke Lambda functions to ingest the data from Amazon S3 to Amazon Redshift Serverless using the Data API, and coordinate the ingestion process using Step Functions, instead of querying it directly from Amazon S3 using Athena and Spark.

Reference:

What is Amazon Athena?

Apache Spark on Amazon Athena

Creating tables, updating the schema, and adding new partitions in the Data Catalog from AWS Glue ETL jobs

Managing Athena workgroups

Using Amazon QuickSight to visualize data in Amazon Athena

AWS Glue Data Catalog AWS Glue Schema Registry What is AWS Glue?

Amazon Redshift Serverless

Amazon Redshift provisioned clusters

Querying external data using Amazon Redshift Spectrum Using stored procedures in Amazon Redshift

What is AWS Lambda?

[Creating and using AWS Lambda UDFs] [Using the Amazon Redshift Data API] [What is AWS Step Functions?]

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

Question: 57

A company stores details about transactions in an Amazon S3 bucket. The company wants to log all writes to the S3 bucket into another S3 bucket that is in the same AWS Region.

Which solution will meet this requirement with the LEAST operational effort?

A. Configure an S3 Event Notifications rule for all activities on the transactions S3 bucket to invoke an AWS Lambda function. Program the Lambda function to write the event to Amazon Kinesis Data Firehose. Configure Kinesis Data Firehose to write the event to the logs S3 bucket.

B. Create a trail of management events in AWS CloudTraiL. Configure the trail to receive data from the transactions S3 bucket. Specify an empty prefix and write-only events. Specify the logs S3 bucket as the destination bucket.

C. Configure an S3 Event Notifications rule for all activities on the transactions S3 bucket to invoke an AWS Lambda function. Program the Lambda function to write the events to the logs S3 bucket.

D. Create a trail of data events in AWS CloudTraiL. Configure the trail to receive data from the transactions S3 bucket. Specify an empty prefix and write-only events. Specify the logs S3 bucket as the destination bucket.

Answer: D

Explanation:

This solution meets the requirement of logging all writes to the S3 bucket into another S3 bucket with the least operational effort. AWS CloudTrail is a service that records the API calls made to AWS services, including Amazon S3. By creating a trail of data events, you can capture the details of the requests that are made to the transactions S3 bucket, such as the requester, the time, the IP address, and the response elements. By specifying an empty prefix and write-only events, you can filter the data events to only include the ones that write to the bucket. By specifying the logs S3 bucket as the destination bucket, you can store the CloudTrail logs in another S3 bucket that is in the same AWS Region. This solution does not require any additional coding or configuration, and it is more scalable and reliable than using S3 Event Notifications and Lambda functions. Reference:

Logging Amazon S3 API calls using AWS CloudTrail

Creating a trail for data events

Enabling Amazon S3 server access logging

Question: 58

A data engineer needs to maintain a central metadata repository that users access through Amazon EMR and Amazon Athena queries. The repository needs to provide the schema and properties of many tables. Some of the metadata is stored in Apache Hive. The data engineer needs to import the metadata from Hive into the central metadata repository.

Which solution will meet these requirements with the LEAST development effort?

A. Use Amazon EMR and Apache Ranger.

B. Use a Hive metastore on an EMR cluster.

C. Use the AWS Glue Data Catalog.

D. Use a metastore on an Amazon RDS for MySQL DB instance.

Answer: C

Explanation:

The AWS Glue Data Catalog is an Apache Hive metastore-compatible catalog that provides a central metadata repository for various data sources and formats. You can use the AWS Glue Data Catalog as an external Hive metastore for Amazon EMR and Amazon Athena queries, and import metadata from existing Hive metastores into the Data Catalog. This solution requires the least development effort, as you can use AWS Glue crawlers to automatically discover and catalog the metadata from Hive, and use the AWS Glue console, AWS CLI, or Amazon EMR API to configure the Data Catalog as the Hive metastore. The other options are either more complex or require additional steps, such as setting up Apache Ranger for security, managing a Hive metastore on an EMR

cluster or an RDS instance, or migrating the metadata manually. Reference:

Using the AWS Glue Data Catalog as the metastore for Hive (Section: Specifying AWS Glue Data Catalog as the metastore)

Metadata Management: Hive Metastore vs AWS Glue (Section: AWS Glue Data Catalog)

AWS Glue Data Catalog support for Spark SQL jobs (Section: Importing metadata from an existing Hive metastore)

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 131)

Question: 59

A company needs to build a data lake in AWS. The company must provide row-level data access and column-level data access to specific teams. The teams will access the data by using Amazon Athena, Amazon Redshift Spectrum, and Apache Hive from Amazon EMR.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use Amazon S3 for data lake storage. Use S3 access policies to restrict data access by rows and columns. Provide data access through Amazon S3.

B. Use Amazon S3 for data lake storage. Use Apache Ranger through Amazon EMR to restrict data access by rows and columns. Provide data access by using Apache Pig.

C. Use Amazon Redshift for data lake storage. Use Redshift security policies to restrict data access by rows and columns. Provide data access by using Apache Spark and Amazon Athena federated queries.

D. Use Amazon S3 for data lake storage. Use AWS Lake Formation to restrict data access by rows and columns. Provide data access through AWS Lake Formation.

Answer: D

Explanation:

Option D is the best solution to meet the requirements with the least operational overhead because AWS Lake Formation is a fully managed service that simplifies the process of building, securing, and managing data lakes. AWS Lake Formation allows you to define granular data access policies at the row and column level for different users and groups. AWS Lake Formation also integrates with Amazon Athena, Amazon Redshift Spectrum, and Apache Hive on Amazon EMR, enabling these services to access the data in the data lake through AWS Lake Formation.

Option A is not a good solution because S3 access policies cannot restrict data access by rows and columns. S3 access policies are based on the identity and permissions of the requester, the bucket and object ownership, and the object prefix and tags. S3 access policies cannot enforce fine-grained data access control at the row and column level.

Option B is not a good solution because it involves using Apache Ranger and Apache Pig, which are not fully managed services and require additional configuration and maintenance. Apache Ranger is a framework that provides centralized security administration for data stored in Hadoop clusters, such as Amazon EMR. Apache Ranger can enforce row-level and column-level access policies for Apache Hive tables. However, Apache Ranger is not a native AWS service and requires manual installation and configuration on Amazon EMR clusters. Apache Pig is a platform that allows you to analyze large data sets using a high-level scripting language called Pig Latin. Apache Pig can access data stored in Amazon S3 and process it using Apache Hive. However, Apache Pig is not a native AWS service and requires manual installation and configuration on Amazon EMR clusters.

Option C is not a good solution because Amazon Redshift is not a suitable service for data lake storage. Amazon Redshift is a fully managed data warehouse service that allows you to run complex analytical queries using standard SQL. Amazon Redshift can enforce row-level and column-level access policies for different users and

groups. However, Amazon Redshift is not designed to store and process large volumes of unstructured or semi-structured data, which are typical characteristics of data lakes. Amazon Redshift is also more expensive and less scalable than Amazon S3 for data lake storage.

Reference:

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

What Is AWS Lake Formation? - AWS Lake Formation

Using AWS Lake Formation with Amazon Athena - AWS Lake Formation

Using AWS Lake Formation with Amazon Redshift Spectrum - AWS Lake Formation

Using AWS Lake Formation with Apache Hive on Amazon EMR - AWS Lake Formation Using Bucket Policies and User Policies - Amazon Simple Storage Service Apache Ranger

Apache Pig

What Is Amazon Redshift? - Amazon Redshift

Question: 60

An airline company is collecting metrics about flight activities for analytics. The company is conducting a proof of concept (POC) test to show how analytics can provide insights that the company can use to increase on-time departures.

The POC test uses objects in Amazon S3 that contain the metrics in .csv format. The POC test uses Amazon Athena to query the dat

a. The data is partitioned in the S3 bucket by date.

As the amount of data increases, the company wants to optimize the storage solution to improve query performance.

Which combination of solutions will meet these requirements? (Choose two.)

A. Add a randomized string to the beginning of the keys in Amazon S3 to get more throughput across partitions.

B. Use an S3 bucket that is in the same account that uses Athena to query the data.

C. Use an S3 bucket that is in the same AWS Region where the company runs Athena queries.

D. Preprocess the .csv data to JSON format by fetching only the document keys that the query requires.

E. Preprocess the .csv data to Apache Parquet format by fetching only the data blocks that are needed for predicates.

Answer: C E

Explanation:

Using an S3 bucket that is in the same AWS Region where the company runs Athena queries can improve query performance by reducing data transfer latency and costs. Preprocessing the .csv data to Apache Parquet format can also improve query performance by enabling columnar storage, compression, and partitioning, which can reduce the amount of data scanned and fetched by the query. These solutions can optimize the storage solution for the POC test without requiring much effort or changes to the existing data pipeline. The other solutions are not optimal or relevant for this requirement. Adding a randomized string to the beginning of the keys in Amazon S3 can improve the throughput across partitions, but it can also make the data harder to query and manage. Using an S3 bucket that is in the same account that uses Athena to query the data does not have any significant impact on query performance, as long as the proper permissions are granted. Preprocessing the .csv data to JSON format does not offer any benefits over the .csv format, as both are row-based and verbose formats that require more data scanning and fetching than columnar formats like Parquet. Reference:

Best Practices When Using Athena with AWS Glue

Question: 61

A company uses Amazon RDS for MySQL as the database for a critical application. The database workload is mostly writes, with a small number of reads.

A data engineer notices that the CPU utilization of the DB instance is very high. The high CPU utilization is slowing down the application. The data engineer must reduce the CPU utilization of the DB Instance.

Which actions should the data engineer take to meet this requirement? (Choose two.)

A. Use the Performance Insights feature of Amazon RDS to identify queries that have high CPU utilization. Optimize the problematic queries.
B. Modify the database schema to include additional tables and indexes.
C. Reboot the RDS DB instance once each week.
D. Upgrade to a larger instance size.
E. Implement caching to reduce the database query load.

Answer: A, E

Explanation:

Amazon RDS is a fully managed service that provides relational databases in the cloud. Amazon RDS for MySQL is one of the supported database engines that you can use to run your applications. Amazon RDS provides various features and tools to monitor and optimize the performance of your DB instances, such as Performance Insights, Enhanced Monitoring, CloudWatch metrics and alarms, etc.

Using the Performance Insights feature of Amazon RDS to identify queries that have high CPU utilization and optimizing the problematic queries will help reduce the CPU utilization of the DB instance. Performance Insights is a feature that allows you to analyze the load on your DB instance and determine what is causing performance issues. Performance Insights collects, analyzes, and displays database performance data using an interactive dashboard. You can use Performance Insights to identify the top SQL statements, hosts, users, or processes that are consuming the most CPU resources. You can also drill down into the details of each query and see the execution plan, wait events, locks, etc. By using Performance Insights, you can pinpoint the root cause of the high CPU utilization and optimize the queries accordingly. For example, you can rewrite the queries to make them more efficient, add or remove indexes, use prepared statements, etc.

Implementing caching to reduce the database query load will also help reduce the CPU utilization of the DB instance. Caching is a technique that allows you to store frequently accessed data in a fast and scalable storage layer, such as Amazon ElastiCache. By using caching, you can reduce the number of requests that hit your database, which in turn reduces the CPU load on your DB instance. Caching also improves the performance and availability of your application, as it reduces the latency and increases the throughput of your data access. You can use caching for various scenarios, such as storing session data, user preferences, application configuration, etc. You can also use caching for read-heavy workloads, such as displaying product details, recommendations, reviews, etc.

The other options are not as effective as using Performance Insights and caching. Modifying the database schema to include additional tables and indexes may or may not improve the CPU utilization, depending on the nature of the workload and the queries. Adding more tables and indexes may increase the complexity and overhead of the database, which may negatively affect the performance. Rebooting the RDS DB instance once each week will not reduce the CPU utilization, as it will not address the underlying cause of the high CPU load. Rebooting may also cause downtime and disruption to your application. Upgrading to a larger instance size may reduce the CPU

utilization, but it will also increase the cost and complexity of your solution. Upgrading may also not be necessary if you can optimize the queries and reduce the database load by using caching.

Reference:

Amazon RDS

Performance Insights

Amazon ElastiCache

[AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide], Chapter 3: Data Storage and Management, Section 3.1: Amazon RDS

Question: 62

A company has used an Amazon Redshift table that is named Orders for 6 months. The company performs weekly updates and deletes on the table. The table has an interleaved sort key on a column that contains AWS Regions.

The company wants to reclaim disk space so that the company will not run out of storage space. The company also wants to analyze the sort key column.

Which Amazon Redshift command will meet these requirements?

A. VACUUM FULL Orders
B. VACUUM DELETE ONLY Orders
C. VACUUM REINDEX Orders
D. VACUUM SORT ONLY Orders

Answer: C

Explanation:

Amazon Redshift is a fully managed, petabyte-scale data warehouse service that enables fast and cost-effective analysis of large volumes of data. Amazon Redshift uses columnar storage, compression, and zone maps to optimize the storage and performance of data. However, over time, as data is inserted, updated, or deleted, the physical storage of data can become fragmented, resulting in wasted disk space and degraded query performance. To address this issue, Amazon Redshift provides the VACUUM command, which reclaims disk space and resorts rows in either a specified table or all tables in the current schema1.

The VACUUM command has four options: FULL, DELETE ONLY, SORT ONLY, and REINDEX. The option that best meets the requirements of the question is VACUUM REINDEX, which re-sorts the rows in a table that has an interleaved sort key and rewrites the table to a new location on disk. An interleaved sort key is a type of sort key that gives equal weight to each column in the sort key, and stores the rows in a way that optimizes the performance of queries that filter by multiple columns in the sort key. However, as data is added or changed, the interleaved sort order can become skewed, resulting in suboptimal query performance. The VACUUM REINDEX option restores the optimal interleaved sort order and reclaims disk space by removing deleted rows. This option also analyzes the sort key column and updates the table statistics, which are used by the query optimizer to generate the most efficient query execution plan23.

The other options are not optimal for the following reasons:

A . VACUUM FULL Orders. This option reclaims disk space by removing deleted rows and resorts the entire table. However, this option is not suitable for tables that have an interleaved sort key, as it does not restore the optimal interleaved sort order. Moreover, this option is the most resourceintensive and time-consuming, as it rewrites the entire table to a new location on disk.

B . VACUUM DELETE ONLY Orders. This option reclaims disk space by removing deleted rows, but does not resort the table. This option is not suitable for tables that have any sort key, as it does not improve the query performance by restoring the sort order. Moreover, this option does not analyze the sort key column and update the table statistics.

D . VACUUM SORT ONLY Orders. This option resorts the entire table, but does not reclaim disk space by removing deleted rows. This option is not suitable for tables that have an interleaved sort key, as it does not restore the optimal interleaved sort order. Moreover, this option does not analyze the sort key column and update the table statistics.

Reference:

1: Amazon Redshift VACUUM

2: Amazon Redshift Interleaved Sorting

3: Amazon Redshift ANALYZE

Question: 63

A manufacturing company wants to collect data from sensors. A data engineer needs to implement a solution that ingests sensor data in near real time.

The solution must store the data to a persistent data store. The solution must store the data in nested JSON format. The company must have the ability to query from the data store with a latency of less than 10 milliseconds.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use a self-hosted Apache Kafka cluster to capture the sensor data. Store the data in Amazon S3 for querying.

B. Use AWS Lambda to process the sensor data. Store the data in Amazon S3 for querying.

C. Use Amazon Kinesis Data Streams to capture the sensor data. Store the data in Amazon DynamoDB for querying.

D. Use Amazon Simple Queue Service (Amazon SQS) to buffer incoming sensor data. Use AWS Glue to store the data in Amazon RDS for querying.

Answer: C

Explanation:

Amazon Kinesis Data Streams is a service that enables you to collect, process, and analyze streaming data in real time. You can use Kinesis Data Streams to capture sensor data from various sources, such as IoT devices, web applications, or mobile apps. You can create data streams that can scale up to handle any amount of data from thousands of producers. You can also use the Kinesis Client Library (KCL) or the Kinesis Data Streams API to write applications that process and analyze the data in the streams1.

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. You can use DynamoDB to store the sensor data in nested JSON format, as DynamoDB supports document data types, such as lists and maps. You can also use DynamoDB to query the data with a latency of less than 10 milliseconds, as DynamoDB offers singledigit millisecond performance for any scale of data. You can use the DynamoDB API or the AWS SDKs to perform queries on the data, such as using key-value lookups, scans, or queries2.

The solution that meets the requirements with the least operational overhead is to use Amazon Kinesis Data Streams to capture the sensor data and store the data in Amazon DynamoDB for querying. This solution has the following advantages:

It does not require you to provision, manage, or scale any servers, clusters, or queues, as Kinesis Data Streams and

DynamoDB are fully managed services that handle all the infrastructure for you. This reduces the operational complexity and cost of running your solution.

It allows you to ingest sensor data in near real time, as Kinesis Data Streams can capture data records as they are produced and deliver them to your applications within seconds. You can also use Kinesis Data Firehose to load the data from the streams to DynamoDB automatically and continuously3.

It allows you to store the data in nested JSON format, as DynamoDB supports document data types, such as lists and maps. You can also use DynamoDB Streams to capture changes in the data and trigger actions, such as sending notifications or updating other databases.

It allows you to query the data with a latency of less than 10 milliseconds, as DynamoDB offers single-digit millisecond performance for any scale of data. You can also use DynamoDB Accelerator (DAX) to improve the read performance by caching frequently accessed data.

Option A is incorrect because it suggests using a self-hosted Apache Kafka cluster to capture the sensor data and store the data in Amazon S3 for querying. This solution has the following disadvantages:

It requires you to provision, manage, and scale your own Kafka cluster, either on EC2 instances or onpremises servers. This increases the operational complexity and cost of running your solution.

It does not allow you to query the data with a latency of less than 10 milliseconds, as Amazon S3 is an object storage service that is not optimized for low-latency queries. You need to use another service, such as Amazon Athena or Amazon Redshift Spectrum, to query the data in S3, which may incur additional costs and latency.

Option B is incorrect because it suggests using AWS Lambda to process the sensor data and store the data in Amazon S3 for querying. This solution has the following disadvantages:

It does not allow you to ingest sensor data in near real time, as Lambda is a serverless compute service that runs code in response to events. You need to use another service, such as API Gateway or Kinesis Data Streams, to trigger Lambda functions with sensor data, which may add extra latency and complexity to your solution.

It does not allow you to query the data with a latency of less than 10 milliseconds, as Amazon S3 is an object storage service that is not optimized for low-latency queries. You need to use another service, such as Amazon Athena or Amazon Redshift Spectrum, to query the data in S3, which may incur additional costs and latency.

Option D is incorrect because it suggests using Amazon Simple Queue Service (Amazon SQS) to buffer incoming sensor data and use AWS Glue to store the data in Amazon RDS for querying. This solution has the following disadvantages:

It does not allow you to ingest sensor data in near real time, as Amazon SQS is a message queue service that delivers messages in a best-effort manner. You need to use another service, such as Lambda or EC2, to poll the messages from the queue and process them, which may add extra latency and complexity to your solution.

It does not allow you to store the data in nested JSON format, as Amazon RDS is a relational database service that supports structured data types, such as tables and columns. You need to use another service, such as AWS Glue, to transform the data from JSON to relational format, which may add extra cost and overhead to your solution.

Reference:

1: Amazon Kinesis Data Streams - Features

2: Amazon DynamoDB - Features

3: Loading Streaming Data into Amazon DynamoDB - Amazon Kinesis Data Firehose

[4] : Capturing Table Activity with DynamoDB Streams - Amazon DynamoDB

[5] : Amazon DynamoDB Accelerator (DAX) - Features

[6] : Amazon S3 - Features

[7] : AWS Lambda - Features

[8] : Amazon Simple Queue Service - Features

[9] : Amazon Relational Database Service - Features

[10] : Working with JSON in Amazon RDS - Amazon Relational Database Service
[11] : AWS Glue - Features

Question: 64

A company stores data in a data lake that is in Amazon S3. Some data that the company stores in the data lake contains personally identifiable information (PII). Multiple user groups need to access the raw dat
a. The company must ensure that user groups can access only the PII that they require.
Which solution will meet these requirements with the LEAST effort?

A. Use Amazon Athena to query the data. Set up AWS Lake Formation and create data filters to establish levels of access for the company's IAM roles. Assign each user to the IAM role that matches the user's PII access requirements.

B. Use Amazon QuickSight to access the data. Use column-level security features in QuickSight to limit the PII that users can retrieve from Amazon S3 by using Amazon Athena. Define QuickSight access levels based on the PII access requirements of the users.

C. Build a custom query builder UI that will run Athena queries in the background to access the data. Create user groups in Amazon Cognito. Assign access levels to the user groups based on the PII access requirements of the users.

D. Create IAM roles that have different levels of granular access. Assign the IAM roles to IAM user groups. Use an identity-based policy to assign access levels to user groups at the column level.

Answer: A

Explanation:

Amazon Athena is a serverless, interactive query service that enables you to analyze data in Amazon S3 using standard SQL. AWS Lake Formation is a service that helps you build, secure, and manage data lakes on AWS. You can use AWS Lake Formation to create data filters that define the level of access for different IAM roles based on the columns, rows, or tags of the data. By using Amazon Athena to query the data and AWS Lake Formation to create data filters, the company can meet the requirements of ensuring that user groups can access only the PII that they require with the least effort. The solution is to use Amazon Athena to query the data in the data lake that is in Amazon S3. Then, set up AWS Lake Formation and create data filters to establish levels of access for the company's IAM roles. For example, a data filter can allow a user group to access only the columns that contain the PII that they need, such as name and email address, and deny access to the columns that contain the PII that they do not need, such as phone number and social security number. Finally, assign each user to the IAM role that matches the user's PII access requirements. This way, the user groups can access the data in the data lake securely and efficiently. The other options are either not feasible or not optimal. Using Amazon QuickSight to access the data (option B) would require the company to pay for the QuickSight service and to configure the column-level security features for each user. Building a custom query builder UI that will run Athena queries in the background to access the data (option C) would require the company to develop and maintain the UI and to integrate it with Amazon Cognito. Creating IAM roles that have different levels of granular access (option D) would require the company to manage multiple IAM roles and policies and to ensure that they are aligned with the data schema. Reference:

Amazon Athena
AWS Lake Formation
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Analysis and Visualization, Section 4.3: Amazon Athena

Question: 65

A data engineer must build an extract, transform, and load (ETL) pipeline to process and load data from 10 source systems into 10 tables that are in an Amazon Redshift database. All the source systems generate .csv, JSON, or Apache Parquet files every 15 minutes. The source systems all deliver files into one Amazon S3 bucket. The file sizes range from 10 MB to 20 GB. The ETL pipeline must function correctly despite changes to the data schema.

Which data pipeline solutions will meet these requirements? (Choose two.)

A. Use an Amazon EventBridge rule to run an AWS Glue job every 15 minutes. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.

B. Use an Amazon EventBridge rule to invoke an AWS Glue workflow job every 15 minutes. Configure the AWS Glue workflow to have an on-demand trigger that runs an AWS Glue crawler and then runs an AWS Glue job when the crawler finishes running successfully. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.

C. Configure an AWS Lambda function to invoke an AWS Glue crawler when a file is loaded into the S3 bucket. Configure an AWS Glue job to process and load the data into the Amazon Redshift tables. Create a second Lambda function to run the AWS Glue job. Create an Amazon EventBridge rule to invoke the second Lambda function when the AWS Glue crawler finishes running successfully.

D. Configure an AWS Lambda function to invoke an AWS Glue workflow when a file is loaded into the S3 bucket. Configure the AWS Glue workflow to have an on-demand trigger that runs an AWS Glue crawler and then runs an AWS Glue job when the crawler finishes running successfully. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.

E. Configure an AWS Lambda function to invoke an AWS Glue job when a file is loaded into the S3 bucket. Configure the AWS Glue job to read the files from the S3 bucket into an Apache Spark DataFrame. Configure the AWS Glue job to also put smaller partitions of the DataFrame into an Amazon Kinesis Data Firehose delivery stream. Configure the delivery stream to load data into the Amazon Redshift tables.

Answer: A, B

Explanation:

Using an Amazon EventBridge rule to run an AWS Glue job or invoke an AWS Glue workflow job every 15 minutes are two possible solutions that will meet the requirements. AWS Glue is a serverless ETL service that can process and load data from various sources to various targets, including Amazon Redshift. AWS Glue can handle different data formats, such as CSV, JSON, and Parquet, and also support schema evolution, meaning it can adapt to changes in the data schema over time. AWS Glue can also leverage Apache Spark to perform distributed processing and transformation of large datasets. AWS Glue integrates with Amazon EventBridge, which is a serverless event bus service that can trigger actions based on rules and schedules. By using an Amazon EventBridge rule, you can invoke an AWS Glue job or workflow every 15 minutes, and configure the job or workflow to run an AWS Glue crawler and then load the data into the Amazon Redshift tables. This way, you can build a cost-effective and scalable ETL pipeline that can handle data from 10 source systems and function correctly despite changes to the data schema.

The other options are not solutions that will meet the requirements. Option C, configuring an AWS Lambda function to invoke an AWS Glue crawler when a file is loaded into the S3 bucket, and creating a second Lambda function to run the AWS Glue job, is not a feasible solution, as it would require a lot of Lambda invocations and coordination. AWS Lambda has some limits on the execution time, memory, and concurrency, which can affect the performance and reliability of the ETL pipeline. Option D, configuring an AWS Lambda function to invoke an AWS Glue workflow when a file is loaded into the S3 bucket, is not a necessary solution, as you can use an Amazon

EventBridge rule to invoke the AWS Glue workflow directly, without the need for a Lambda function. Option E, configuring an AWS Lambda function to invoke an AWS Glue job when a file is loaded into the S3 bucket, and configuring the AWS Glue job to put smaller partitions of the DataFrame into an Amazon Kinesis Data Firehose delivery stream, is not a cost-effective solution, as it would incur additional costs for Lambda invocations and data delivery. Moreover, using Amazon Kinesis Data Firehose to load data into Amazon Redshift is not suitable for frequent and small batches of data, as it can cause performance issues and data fragmentation. Reference:

AWS Glue

Amazon EventBridge

Using AWS Glue to run ETL jobs against non-native JDBC data sources

[AWS Lambda quotas]

[Amazon Kinesis Data Firehose quotas]

Question: 66

A financial company wants to use Amazon Athena to run on-demand SQL queries on a petabyte-scale dataset to support a business intelligence (BI) application. An AWS Glue job that runs during nonbusiness hours updates the dataset once every day. The BI application has a standard data refresh frequency of 1 hour to comply with company policies.

A data engineer wants to cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs.

Which solution will meet these requirements with the LEAST operational overhead?

A. Configure an Amazon S3 Lifecycle policy to move data to the S3 Glacier Deep Archive storage class after 1 day

B. Use the query result reuse feature of Amazon Athena for the SQL queries.

C. Add an Amazon ElastiCache cluster between the Bl application and Athena.

D. Change the format of the files that are in the dataset to Apache Parquet.

Answer: B

Explanation:

The best solution to cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs is to use the query result reuse feature of Amazon Athena for the SQL queries. This feature allows you to run the same query multiple times without incurring additional charges, as long as the underlying data has not changed and the query results are still in the query result location in Amazon S31. This feature is useful for scenarios where you have a petabyte-scale dataset that is updated infrequently, such as once a day, and you have a BI application that runs the same queries repeatedly, such as every hour. By using the query result reuse feature, you can reduce the amount of data scanned by your queries and save on the cost of running Athena. You can enable or disable this feature at the workgroup level or at the individual query level1.

Option A is not the best solution, as configuring an Amazon S3 Lifecycle policy to move data to the S3 Glacier Deep Archive storage class after 1 day would not cost optimize the company's use of Amazon Athena, but rather increase the cost and complexity. Amazon S3 Lifecycle policies are rules that you can define to automatically transition objects between different storage classes based on specified criteria, such as the age of the object2. S3 Glacier Deep Archive is the lowest-cost storage class in Amazon S3, designed for long-term data archiving that is accessed once or twice in a year3. While moving data to S3 Glacier Deep Archive can reduce the storage cost, it

would also increase the retrieval cost and latency, as it takes up to 12 hours to restore the data from S3 Glacier Deep Archive3. Moreover, Athena does not support querying data that is in S3 Glacier or S3 Glacier Deep Archive storage classes4. Therefore, using this option would not meet the requirements of running on-demand SQL queries on the dataset.

Option C is not the best solution, as adding an Amazon ElastiCache cluster between the BI application and Athena would not cost optimize the company's use of Amazon Athena, but rather increase the cost and complexity. Amazon ElastiCache is a service that offers fully managed inmemory data stores, such as Redis and Memcached, that can improve the performance and scalability of web applications by caching frequently accessed data. While using ElastiCache can reduce the latency and load on the BI application, it would not reduce the amount of data scanned by Athena, which is the main factor that determines the cost of running Athena. Moreover, using ElastiCache would introduce additional infrastructure costs and operational overhead, as you would have to provision, manage, and scale the ElastiCache cluster, and integrate it with the BI application and Athena.

Option D is not the best solution, as changing the format of the files that are in the dataset to Apache Parquet would not cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs, but rather increase the complexity. Apache Parquet is a columnar storage format that can improve the performance of analytical queries by reducing the amount of data that needs to be scanned and providing efficient compression and encoding schemes. However, changing the format of the files that are in the dataset to Apache Parquet would require additional processing and transformation steps, such as using AWS Glue or Amazon EMR to convert the files from their original format to Parquet, and storing the converted files in a separate location in Amazon S3. This would increase the complexity and the operational overhead of the data pipeline, and also incur additional costs for using AWS Glue or Amazon EMR. Reference:

Query result reuse

Amazon S3 Lifecycle

S3 Glacier Deep Archive

Storage classes supported by Athena [What is Amazon ElastiCache?] [Amazon Athena pricing] [Columnar Storage Formats] AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

---

Question: 67 A company's data engineer needs to optimize the performance of table SQL queries. The company stores data in an Amazon Redshift cluster. The data engineer cannot increase the size of the cluster because of budget constraints.

The company stores the data in multiple tables and loads the data by using the EVEN distribution style. Some tables are hundreds of gigabytes in size. Other tables are less than 10 MB in size. Which solution will meet these requirements?

A. Keep using the EVEN distribution style for all tables. Specify primary and foreign keys for all tables.
B. Use the ALL distribution style for large tables. Specify primary and foreign keys for all tables.
C. Use the ALL distribution style for rarely updated small tables. Specify primary and foreign keys for all tables.
D. Specify a combination of distribution, sort, and partition keys for all tables.

Answer: D

Explanation:

This solution meets the requirements of optimizing the performance of table SQL queries without increasing the size of the cluster. By using the ALL distribution style for rarely updated small tables, you can ensure that the entire table is copied to every node in the cluster, which eliminates the need for data redistribution during joins. This can improve query performance significantly, especially for frequently joined dimension tables. However, using the ALL distribution style also increases the storage space and the load time, so it is only suitable for small tables that are not updated frequently or extensively. By specifying primary and foreign keys for all tables, you can help the query optimizer to generate better query plans and avoid unnecessary scans or joins. You can also use the

AUTO distribution style to let Amazon Redshift choose the optimal distribution style based on the table size and the query patterns. Reference:

Choose the best distribution style

Distribution styles

Working with data distribution styles

Question: 68

A company receives .csv files that contain physical address dat
a. The data is in columns that have the following names: Door_No, Street_Name, City, and Zip_Code. The company wants to create a single column to store these values in the following format:

```
"Door_No": "24",
"Street_Name": "AAA street"
"City": "BBB",
"Zip_Code": "111111"
```

Which solution will meet this requirement with the LEAST coding effort?

A. Use AWS Glue DataBrew to read the files. Use the NEST TO ARRAY transformation to create the new column.

B. Use AWS Glue DataBrew to read the files. Use the NEST TO MAP transformation to create the new column.

C. Use AWS Glue DataBrew to read the files. Use the PIVOT transformation to create the new column.

D. Write a Lambda function in Python to read the files. Use the Python data dictionary type to create the new column.

Answer: B

Explanation:

The NEST TO MAP transformation allows you to combine multiple columns into a single column that contains a JSON object with key-value pairs. This is the easiest way to achieve the desired format for the physical address data, as you can simply select the columns to nest and specify the keys for each column. The NEST TO ARRAY transformation creates a single column that contains an array of values, which is not the same as the JSON object format. The PIVOT transformation reshapes the data by creating new columns from unique values in a selected column, which is not applicable for this use case. Writing a Lambda function in Python requires more coding effort than using AWS Glue DataBrew, which provides a visual and interactive interface for data transformations.

Reference: 7 most common data preparation transformations in AWS Glue DataBrew (Section: Nesting and unnesting columns)

NEST TO MAP - AWS Glue DataBrew (Section: Syntax)

Question: 69

A company receives call logs as Amazon S3 objects that contain sensitive customer information. The company must protect the S3 objects by using encryption. The company must also use encryption keys that only specific employees can access.

Which solution will meet these requirements with the LEAST effort?

A. Use an AWS CloudHSM cluster to store the encryption keys. Configure the process that writes to Amazon S3 to make calls to CloudHSM to encrypt and decrypt the objects. Deploy an IAM policy that restricts access to the CloudHSM cluster.

B. Use server-side encryption with customer-provided keys (SSE-C) to encrypt the objects that contain customer information. Restrict access to the keys that encrypt the objects.

C. Use server-side encryption with AWS KMS keys (SSE-KMS) to encrypt the objects that contain customer information. Configure an IAM policy that restricts access to the KMS keys that encrypt the objects.

D. Use server-side encryption with Amazon S3 managed keys (SSE-S3) to encrypt the objects that contain customer information. Configure an IAM policy that restricts access to the Amazon S3 managed keys that encrypt the objects.

Answer: C

Explanation: Option C is the best solution to meet the requirements with the least effort because server-side encryption with AWS KMS keys (SSE-KMS) is a feature that allows you to encrypt data at rest in Amazon S3 using keys managed by AWS Key Management Service (AWS KMS). AWS KMS is a fully managed service that enables you to create and manage encryption keys for your AWS services and applications. AWS KMS also allows you to define granular access policies for your keys, such as who can use them to encrypt and decrypt data, and under what conditions. By using SSE-KMS, you can protect your S3 objects by using encryption keys that only specific employees can access, without having to manage the encryption and decryption process yourself.

Option A is not a good solution because it involves using AWS CloudHSM, which is a service that provides hardware security modules (HSMs) in the AWS Cloud. AWS CloudHSM allows you to generate and use your own encryption keys on dedicated hardware that is compliant with various standards and regulations. However, AWS CloudHSM is not a fully managed service and requires more effort to set up and maintain than AWS KMS. Moreover, AWS CloudHSM does not integrate with Amazon S3, so you have to configure the process that writes to S3 to make calls to CloudHSM to encrypt and decrypt the objects, which adds complexity and latency to the data protection process. Option B is not a good solution because it involves using server-side encryption with customer-provided keys (SSE-C), which is a feature that allows you to encrypt data at rest in Amazon S3 using keys that you provide and manage yourself. SSE-C requires you to send your encryption key along with each request to upload or retrieve an object. However, SSE-C does not provide any mechanism to restrict access to the keys that encrypt the objects, so you have to implement your own key management and access control system, which adds more effort and risk to the data protection process.

Option D is not a good solution because it involves using server-side encryption with Amazon S3 managed keys (SSE-S3), which is a feature that allows you to encrypt data at rest in Amazon S3 using keys that are managed by Amazon S3. SSE-S3 automatically encrypts and decrypts your objects as they are uploaded and downloaded from S3. However, SSE-S3 does not allow you to control who can access the encryption keys or under what conditions. SSE-S3 uses a single encryption key for each S3 bucket, which is shared by all users who have access to the bucket. This means that you cannot restrict access to the keys that encrypt the objects by specific employees, which does not meet the requirements.

Reference:

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

Protecting Data Using Server-Side Encryption with AWS KMS-Managed Encryption Keys (SSE-KMS) - Amazon Simple Storage Service

What is AWS Key Management Service? - AWS Key Management Service

What is AWS CloudHSM? - AWS CloudHSM

Protecting Data Using Server-Side Encryption with Customer-Provided Encryption Keys (SSE-C) - Amazon Simple Storage Service

Protecting Data Using Server-Side Encryption with Amazon S3-Managed Encryption Keys (SSE-S3) - Amazon Simple Storage Service

Question: 70

A company stores petabytes of data in thousands of Amazon S3 buckets in the S3 Standard storage class. The data supports analytics workloads that have unpredictable and variable data access patterns.

The company does not access some data for months. However, the company must be able to retrieve all data within milliseconds. The company needs to optimize S3 storage costs.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use S3 Storage Lens standard metrics to determine when to move objects to more cost-optimized storage classes. Create S3 Lifecycle policies for the S3 buckets to move objects to cost-optimized storage classes. Continue to refine the S3 Lifecycle policies in the future to optimize storage costs.

B. Use S3 Storage Lens activity metrics to identify S3 buckets that the company accesses infrequently. Configure S3 Lifecycle rules to move objects from S3 Standard to the S3 Standard-Infrequent Access (S3 Standard-IA) and S3 Glacier storage classes based on the age of the data.

C. Use S3 Intelligent-Tiering. Activate the Deep Archive Access tier.

D. Use S3 Intelligent-Tiering. Use the default access tier.

Answer: D

Explanation:

S3 Intelligent-Tiering is a storage class that automatically moves objects between four access tiers based on the changing access patterns. The default access tier consists of two tiers: Frequent Access and Infrequent Access. Objects in the Frequent Access tier have the same performance and availability as S3 Standard, while objects in the Infrequent Access tier have the same performance and availability as S3 Standard-IA. S3 Intelligent-Tiering monitors the access patterns of each object and moves them between the tiers accordingly, without any operational overhead or retrieval fees. This solution can optimize S3 storage costs for data with unpredictable and variable access patterns, while ensuring millisecond latency for data retrieval. The other solutions are not optimal or relevant for this requirement. Using S3 Storage Lens standard metrics and activity metrics can provide insights into the storage usage and access patterns, but they do not automate the data movement between storage classes. Creating S3 Lifecycle policies for the S3 buckets can move objects to more cost- optimized storage classes, but they require manual configuration and maintenance, and they may incur retrieval fees for data that is accessed unexpectedly. Activating the Deep Archive Access tier for S3 Intelligent-Tiering can further reduce the storage costs for data that is rarely accessed, but it also increases the retrieval time to 12 hours, which does not meet the requirement of millisecond latency. Reference:

S3 Intelligent-Tiering

S3 Storage Lens

S3 Lifecycle policies

[AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

Question: 71

During a security review, a company identified a vulnerability in an AWS Glue job. The company discovered that credentials to access an Amazon Redshift cluster were hard coded in the job script. A data engineer must remediate the security vulnerability in the AWS Glue job. The solution must securely store the credentials.

Which combination of steps should the data engineer take to meet these requirements? (Choose two.)

A. Store the credentials in the AWS Glue job parameters.
B. Store the credentials in a configuration file that is in an Amazon S3 bucket.
C. Access the credentials from a configuration file that is in an Amazon S3 bucket by using the AWS Glue job.
D. Store the credentials in AWS Secrets Manager.
E. Grant the AWS Glue job 1AM role access to the stored credentials.

Answer: D E

Explanation:
AWS Secrets Manager is a service that allows you to securely store and manage secrets, such as database credentials, API keys, passwords, etc. You can use Secrets Manager to encrypt, rotate, and audit your secrets, as well as to control access to them using fine-grained policies. AWS Glue is a fully managed service that provides a serverless data integration platform for data preparation, data cataloging, and data loading. AWS Glue jobs allow you to transform and load data from various sources into various targets, using either a graphical interface (AWS Glue Studio) or a code-based interface (AWS Glue console or AWS Glue API).
Storing the credentials in AWS Secrets Manager and granting the AWS Glue job 1AM role access to the stored credentials will meet the requirements, as it will remediate the security vulnerability in the AWS Glue job and securely store the credentials. By using AWS Secrets Manager, you can avoid hard coding the credentials in the job script, which is a bad practice that exposes the credentials to unauthorized access or leakage. Instead, you can store the credentials as a secret in Secrets Manager and reference the secret name or ARN in the job script. You can also use Secrets Manager to encrypt the credentials using AWS Key Management Service (AWS KMS), rotate the credentials automatically or on demand, and monitor the access to the credentials using AWS CloudTrail. By granting the AWS Glue job 1AM role access to the stored credentials, you can use the principle of least privilege to ensure that only the AWS Glue job can retrieve the credentials from Secrets Manager. You can also use resource-based or tag-based policies to further restrict the access to the credentials.
The other options are not as secure as storing the credentials in AWS Secrets Manager and granting the AWS Glue job 1AM role access to the stored credentials. Storing the credentials in the AWS Glue job parameters will not remediate the security vulnerability, as the job parameters are still visible in the AWS Glue console and API. Storing the credentials in a configuration file that is in an Amazon S3 bucket and accessing the credentials from the configuration file by using the AWS Glue job will not be as secure as using Secrets Manager, as the configuration file may not be encrypted or rotated, and the access to the file may not be audited or controlled.
Reference:
AWS Secrets Manager
AWS Glue
AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 6: Data Integration and Transformation, Section 6.1: AWS Glue

Question: 72

A data engineer uses Amazon Redshift to run resource-intensive analytics processes once every month. Every month, the data engineer creates a new Redshift provisioned cluster. The data engineer deletes the Redshift provisioned cluster after the analytics processes are complete every month. Before the data engineer deletes the cluster each month, the data engineer unloads backup data from the cluster to an Amazon S3 bucket.
The data engineer needs a solution to run the monthly analytics processes that does not require the data engineer to manage the infrastructure manually.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use Amazon Step Functions to pause the Redshift cluster when the analytics processes are complete and to resume the cluster to run new processes every month.
B. Use Amazon Redshift Serverless to automatically process the analytics workload.
C. Use the AWS CLI to automatically process the analytics workload.
D. Use AWS CloudFormation templates to automatically process the analytics workload.

Answer: B

Explanation:
Amazon Redshift Serverless is a new feature of Amazon Redshift that enables you to run SQL queries on data in Amazon S3 without provisioning or managing any clusters. You can use Amazon Redshift Serverless to automatically process the analytics workload, as it scales up and down the compute resources based on the query demand, and charges you only for the resources consumed. This solution will meet the requirements with the least operational overhead, as it does not require the data engineer to create, delete, pause, or resume any Redshift clusters, or to manage any infrastructure manually. You can use the Amazon Redshift Data API to run queries from the AWS CLI, AWS SDK, or AWS Lambda functions12.
The other options are not optimal for the following reasons:
A . Use Amazon Step Functions to pause the Redshift cluster when the analytics processes are complete and to resume the cluster to run new processes every month. This option is not recommended, as it would still require the data engineer to create and delete a new Redshift provisioned cluster every month, which can incur additional costs and time. Moreover, this option would require the data engineer to use Amazon Step Functions to orchestrate the workflow of pausing and resuming the cluster, which can add complexity and overhead.
C . Use the AWS CLI to automatically process the analytics workload. This option is vague and does not specify how the AWS CLI is used to process the analytics workload. The AWS CLI can be used to run queries on data in Amazon S3 using Amazon Redshift Serverless, Amazon Athena, or Amazon EMR, but each of these services has different features and benefits. Moreover, this option does not address the requirement of not managing the infrastructure manually, as the data engineer may still need to provision and configure some resources, such as Amazon EMR clusters or Amazon Athena workgroups.
D . Use AWS CloudFormation templates to automatically process the analytics workload. This option is also vague and does not specify how AWS CloudFormation templates are used to process the analytics workload. AWS CloudFormation is a service that lets you model and provision AWS resources using templates. You can use AWS CloudFormation templates to create and delete a Redshift provisioned cluster every month, or to create and configure other AWS resources, such as Amazon EMR, Amazon Athena, or Amazon Redshift Serverless. However, this option does not address the requirement of not managing the infrastructure manually, as the data engineer may still need to write and maintain the AWS CloudFormation templates, and to monitor the status and performance of the resources.
Reference:
1: Amazon Redshift Serverless 2: Amazon Redshift Data API : Amazon Step Functions : AWS CLI : AWS CloudFormation

Question: 73 A company receives a daily file that contains customer data in .xls format. The company stores the file in Amazon S3. The daily file is approximately 2 GB in size.
A data engineer concatenates the column in the file that contains customer first names and the column that contains customer last names. The data engineer needs to determine the number of distinct customers in the file.
Which solution will meet this requirement with the LEAST operational effort?

A. Create and run an Apache Spark job in an AWS Glue notebook. Configure the job to read the S3 file and calculate the number of distinct customers.

B. Create an AWS Glue crawler to create an AWS Glue Data Catalog of the S3 file. Run SQL queries from Amazon Athena to calculate the number of distinct customers.

C. Create and run an Apache Spark job in Amazon EMR Serverless to calculate the number of distinct customers.

D. Use AWS Glue DataBrew to create a recipe that uses the COUNT_DISTINCT aggregate function to calculate the number of distinct customers.

Answer: D

Explanation:

AWS Glue DataBrew is a visual data preparation tool that allows you to clean, normalize, and transform data without writing code. You can use DataBrew to create recipes that define the steps to apply to your data, such as filtering, renaming, splitting, or aggregating columns. You can also use DataBrew to run jobs that execute the recipes on your data sources, such as Amazon S3, Amazon Redshift, or Amazon Aurora. DataBrew integrates with AWS Glue Data Catalog, which is a centralized metadata repository for your data assets1.

The solution that meets the requirement with the least operational effort is to use AWS Glue DataBrew to create a recipe that uses the COUNT_DISTINCT aggregate function to calculate the number of distinct customers. This solution has the following advantages:

It does not require you to write any code, as DataBrew provides a graphical user interface that lets you explore, transform, and visualize your data. You can use DataBrew to concatenate the columns that contain customer first names and last names, and then use the COUNT DISTINCT aggregate function to count the number of unique values in the resulting column2.

It does not require you to provision, manage, or scale any servers, clusters, or notebooks, as DataBrew is a fully managed service that handles all the infrastructure for you. DataBrew can automatically scale up or down the compute resources based on the size and complexity of your data and recipes1.

It does not require you to create or update any AWS Glue Data Catalog entries, as DataBrew can automatically create and register the data sources and targets in the Data Catalog. DataBrew can also use the existing Data Catalog entries to access the data in S3 or other sources3.

Option A is incorrect because it suggests creating and running an Apache Spark job in an AWS Glue notebook. This solution has the following disadvantages:

It requires you to write code, as AWS Glue notebooks are interactive development environments that allow you to write, test, and debug Apache Spark code using Python or Scala. You need to use the Spark SQL or the Spark DataFrame API to read the S3 file and calculate the number of distinct customers.

It requires you to provision and manage a development endpoint, which is a serverless Apache Spark environment that you can connect to your notebook. You need to specify the type and number of workers for your development endpoint, and monitor its status and metrics.

It requires you to create or update the AWS Glue Data Catalog entries for the S3 file, either manually or using a crawler. You need to use the Data Catalog as a metadata store for your Spark job, and specify the database and table names in your code.

Option B is incorrect because it suggests creating an AWS Glue crawler to create an AWS Glue Data Catalog of the S3 file, and running SQL queries from Amazon Athena to calculate the number of distinct customers. This solution has the following disadvantages:

It requires you to create and run a crawler, which is a program that connects to your data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in the Data Catalog. You need to specify the data store, the IAM role, the schedule, and the output database for your crawler.

It requires you to write SQL queries, as Amazon Athena is a serverless interactive query service that allows you to analyze data in S3 using standard SQL. You need to use Athena to concatenate the columns that contain customer first names and last names, and then use the COUNT(DISTINCT) aggregate function to count the number of unique values in the resulting column.

Option C is incorrect because it suggests creating and running an Apache Spark job in Amazon EMR Serverless to calculate the number of distinct customers. This solution has the following disadvantages:

It requires you to write code, as Amazon EMR Serverless is a service that allows you to run Apache Spark jobs on AWS without provisioning or managing any infrastructure. You need to use the Spark SQL or the Spark DataFrame API to read the S3 file and calculate the number of distinct customers.

It requires you to create and manage an Amazon EMR Serverless cluster, which is a fully managed and scalable Spark environment that runs on AWS Fargate. You need to specify the cluster name, the IAM role, the VPC, and the subnet for your cluster, and monitor its status and metrics.

It requires you to create or update the AWS Glue Data Catalog entries for the S3 file, either manually or using a crawler. You need to use the Data Catalog as a metadata store for your Spark job, and specify the database and table names in your code.

Reference:

1: AWS Glue DataBrew - Features
2: Working with recipes - AWS Glue DataBrew
3: Working with data sources and data targets - AWS Glue DataBrew
[4] : AWS Glue notebooks - AWS Glue
[5] : Development endpoints - AWS Glue
[6] : Populating the AWS Glue Data Catalog - AWS Glue
[7] : Crawlers - AWS Glue
[8] : Amazon Athena - Features
[9] : Amazon EMR Serverless - Features
[10] : Creating an Amazon EMR Serverless cluster - Amazon EMR
[11] : Using the AWS Glue Data Catalog with Amazon EMR Serverless - Amazon EMR

Question: 74

A healthcare company uses Amazon Kinesis Data Streams to stream real-time health data from wearable devices, hospital equipment, and patient records.

A data engineer needs to find a solution to process the streaming dat

a. The data engineer needs to store the data in an Amazon Redshift Serverless warehouse. The solution must support near real-time analytics of the streaming data and the previous day's data. Which solution will meet these requirements with the LEAST operational overhead?

A. Load data into Amazon Kinesis Data Firehose. Load the data into Amazon Redshift.
B. Use the streaming ingestion feature of Amazon Redshift.
C. Load the data into Amazon S3. Use the COPY command to load the data into Amazon Redshift.
D. Use the Amazon Aurora zero-ETL integration with Amazon Redshift.

Answer: B

Explanation:

The streaming ingestion feature of Amazon Redshift enables you to ingest data from streaming sources, such as Amazon Kinesis Data Streams, into Amazon Redshift tables in near real-time. You can use the streaming ingestion feature to process the streaming data from the wearable devices, hospital equipment, and patient records. The streaming ingestion feature also supports incremental updates, which means you can append new data or update

existing data in the Amazon Redshift tables. This way, you can store the data in an Amazon Redshift Serverless warehouse and support near real-time analytics of the streaming data and the previous day's data. This solution meets the requirements with the least operational overhead, as it does not require any additional services or components to ingest and process the streaming data. The other options are either not feasible or not optimal. Loading data into Amazon Kinesis Data Firehose and then into Amazon Redshift (option A) would introduce additional latency and cost, as well as require additional configuration and management. Loading data into Amazon S3 and then using the COPY command to load the data into Amazon Redshift (option C) would also introduce additional latency and cost, as well as require additional storage space and ETL logic. Using the Amazon Aurora zero-ETL integration with Amazon Redshift (option D) would not work, as it requires the data to be stored in Amazon Aurora first, which is not the case for the streaming data from the healthcare company.

Reference:

Using streaming ingestion with Amazon Redshift

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 3: Data Ingestion and Transformation, Section 3.5: Amazon Redshift Streaming Ingestion

Question: 75

A data engineer needs to use an Amazon QuickSight dashboard that is based on Amazon Athena queries on data that is stored in an Amazon S3 bucket. When the data engineer connects to the QuickSight dashboard, the data engineer receives an error message that indicates insufficient permissions.

Which factors could cause to the permissions-related errors? (Choose two.)

A. There is no connection between QuickSgqht and Athena.
B. The Athena tables are not cataloged.
C. QuickSiqht does not have access to the S3 bucket.
D. QuickSight does not have access to decrypt S3 data.
E. There is no 1AM role assigned to QuickSiqht.

Answer: C D

Explanation:

QuickSight does not have access to the S3 bucket and QuickSight does not have access to decrypt S3 data are two possible factors that could cause the permissions-related errors. Amazon QuickSight is a business intelligence service that allows you to create and share interactive dashboards based on various data sources, including Amazon Athena. Amazon Athena is a serverless query service that allows you to analyze data stored in Amazon S3 using standard SQL. To use an Amazon QuickSight dashboard that is based on Amazon Athena queries on data that is stored in an Amazon S3 bucket, you need to grant QuickSight access to both Athena and S3, as well as any encryption keys that are used to encrypt the S3 data. If QuickSight does not have access to the S3 bucket or the encryption keys, it will not be able to read the data from Athena and display it on the dashboard, resulting in an error message that indicates insufficient permissions.

The other options are not factors that could cause the permissions-related errors. Option A, there is no connection between QuickSight and Athena, is not a factor, as QuickSight supports Athena as a native data source, and you can easily create a connection between them using the QuickSight console or the API. Option B, the Athena tables are not cataloged, is not a factor, as QuickSight can automatically discover the Athena tables that are cataloged in the AWS Glue Data Catalog, and you can also manually specify the Athena tables that are not cataloged. Option E, there is no IAM role assigned to QuickSight, is not a factor, as QuickSight requires an IAM role to access any AWS data sources, including Athena and S3, and you can create and assign an IAM role to QuickSight using the QuickSight console or the API. Reference:

Using Amazon Athena as a Data Source
Granting Amazon QuickSight Access to AWS Resources
Encrypting Data at Rest in Amazon S3

Question: 76

A company stores datasets in JSON format and .csv format in an Amazon S3 bucket. The company has Amazon RDS for Microsoft SQL Server databases, Amazon DynamoDB tables that are in provisioned capacity mode, and an Amazon Redshift cluster. A data engineering team must develop a solution that will give data scientists the ability to query all data sources by using syntax similar to SQL. Which solution will meet these requirements with the LEAST operational overhead?

A. Use AWS Glue to crawl the data sources. Store metadata in the AWS Glue Data Catalog. Use Amazon Athena to query the data. Use SQL for structured data sources. Use PartiQL for data that is stored in JSON format.
B. Use AWS Glue to crawl the data sources. Store metadata in the AWS Glue Data Catalog. Use Redshift Spectrum to query the data. Use SQL for structured data sources. Use PartiQL for data that is stored in JSON format.
C. Use AWS Glue to crawl the data sources. Store metadata in the AWS Glue Data Catalog. Use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv format. Store the transformed data in an S3 bucket. Use Amazon Athena to query the original and transformed data from the S3 bucket.
D. Use AWS Lake Formation to create a data lake. Use Lake Formation jobs to transform the data from all data sources to Apache Parquet format. Store the transformed data in an S3 bucket. Use Amazon Athena or Redshift Spectrum to query the data.

Answer: A

Explanation:
The best solution to meet the requirements of giving data scientists the ability to query all data sources by using syntax similar to SQL with the least operational overhead is to use AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, use Amazon Athena to query the data, use SQL for structured data sources, and use PartiQL for data that is stored in JSON format. AWS Glue is a serverless data integration service that makes it easy to prepare, clean, enrich, and move data between data stores1. AWS Glue crawlers are processes that connect to a data store, progress through a prioritized list of classifiers to determine the schema for your data, and then create metadata tables in the Data Catalog2. The Data Catalog is a persistent metadata store that contains table definitions, job definitions, and other control information to help you manage your AWS Glue components3. You can use AWS Glue to crawl the data sources, such as Amazon S3, Amazon RDS for Microsoft SQL Server, and Amazon DynamoDB, and store the metadata in the Data Catalog.
Amazon Athena is a serverless, interactive query service that makes it easy to analyze data directly in Amazon S3 using standard SQL or Python4. Amazon Athena also supports PartiQL, a SQL-compatible query language that lets you query, insert, update, and delete data from semi-structured and nested data, such as JSON. You can use Amazon Athena to query the data from the Data Catalog using SQL for structured data sources, such as .csv files and relational databases, and PartiQL for data that is stored in JSON format. You can also use Athena to query data from other data sources, such as Amazon Redshift, using federated queries.
Using AWS Glue and Amazon Athena to query all data sources by using syntax similar to SQL is the least operational overhead solution, as you do not need to provision, manage, or scale any infrastructure, and you pay only for the resources you use. AWS Glue charges you based on the compute time and the data processed by your crawlers and ETL jobs1. Amazon Athena charges you based on the amount of data scanned by your queries. You can also reduce the cost and improve the performance of your queries by using compression, partitioning, and columnar formats for your data in Amazon S3.

Option B is not the best solution, as using AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, and use Redshift Spectrum to query the data, would incur more costs and complexity than using Amazon Athena. Redshift Spectrum is a feature of Amazon Redshift, a fully managed data warehouse service, that allows you to query and join data across your data warehouse and your data lake using standard SQL. While Redshift Spectrum is powerful and useful for many data warehousing scenarios, it is not necessary or cost-effective for querying all data sources by using syntax similar to SQL. Redshift Spectrum charges you based on the amount of data scanned by your queries, which is similar to Amazon Athena, but it also requires you to have an Amazon Redshift cluster, which charges you based on the node type, the number of nodes, and the duration of the cluster5. These costs can add up quickly, especially if you have large volumes of data and complex queries. Moreover, using Redshift Spectrum would introduce additional latency and complexity, as you would have to provision and manage the cluster, and create an external schema and database for the data in the Data Catalog, instead of querying it directly from Amazon Athena.

Option C is not the best solution, as using AWS Glue to crawl the data sources, store metadata in the AWS Glue Data Catalog, use AWS Glue jobs to transform data that is in JSON format to Apache Parquet or .csv format, store the transformed data in an S3 bucket, and use Amazon Athena to query the original and transformed data from the S3 bucket, would incur more costs and complexity than using Amazon Athena with PartiQL. AWS Glue jobs are ETL scripts that you can write in Python or Scala to transform your data and load it to your target data store. Apache Parquet is a columnar storage format that can improve the performance of analytical queries by reducing the amount of data that needs to be scanned and providing efficient compression and encoding schemes6. While using AWS Glue jobs and Parquet can improve the performance and reduce the cost of your queries, they would also increase the complexity and the operational overhead of the data pipeline, as you would have to write, run, and monitor the ETL jobs, and store the transformed data in a separate location in Amazon S3. Moreover, using AWS Glue jobs and Parquet would introduce additional latency, as you would have to wait for the ETL jobs to finish before querying the transformed data. Option D is not the best solution, as using AWS Lake Formation to create a data lake, use Lake Formation jobs to transform the data from all data sources to Apache Parquet format, store the transformed data in an S3 bucket, and use Amazon Athena or Redshift Spectrum to query the data, would incur more costs and complexity than using Amazon Athena with PartiQL. AWS Lake Formation is a service that helps you centrally govern, secure, and globally share data for analytics and machine learning7. Lake Formation jobs are ETL jobs that you can create and run using the Lake Formation console or API. While using Lake Formation and Parquet can improve the performance and reduce the cost of your queries, they would also increase the complexity and the operational overhead of the data pipeline, as you would have to create, run, and monitor the Lake Formation jobs, and store the transformed data in a separate location in Amazon S3. Moreover, using Lake Formation and Parquet would introduce additional latency, as you would have to wait for the Lake Formation jobs to finish before querying the transformed data. Furthermore, using Redshift Spectrum to query the data would also incur the same costs and complexity as mentioned in option

B. Reference: What is Amazon Athena?

Data Catalog and crawlers in AWS Glue

AWS Glue Data Catalog

Columnar Storage Formats

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

AWS Glue Schema Registry

What is AWS Glue?

Amazon Redshift Serverless

Amazon Redshift provisioned clusters

[Querying external data using Amazon Redshift Spectrum] [Using stored procedures in Amazon Redshift] [What is AWS Lambda?] [PartiQL for Amazon Athena] [Federated queries in Amazon Athena] [Amazon Athena pricing]

[Top 10 performance tuning tips for Amazon Athena]

[AWS Glue ETL jobs]

[AWS Lake Formation jobs]

Question: 77

A data engineer is configuring Amazon SageMaker Studio to use AWS Glue interactive sessions to prepare data for machine learning (ML) models.

The data engineer receives an access denied error when the data engineer tries to prepare the data by using SageMaker Studio.

Which change should the engineer make to gain access to SageMaker Studio?

A. Add the AWSGlueServiceRole managed policy to the data engineer's IAM user.

B. Add a policy to the data engineer's IAM user that includes the sts:AssumeRole action for the AWS Glue and SageMaker service principals in the trust policy.

C. Add the AmazonSageMakerFullAccess managed policy to the data engineer's IAM user.

D. Add a policy to the data engineer's IAM user that allows the sts:AddAssociation action for the AWS Glue and SageMaker service principals in the trust policy.

Answer: B

Explanation:

This solution meets the requirement of gaining access to SageMaker Studio to use AWS Glue interactive sessions. AWS Glue interactive sessions are a way to use AWS Glue DataBrew and AWS Glue Data Catalog from within SageMaker Studio. To use AWS Glue interactive sessions, the data engineer's IAM user needs to have permissions to assume the AWS Glue service role and the SageMaker execution role. By adding a policy to the data engineer's IAM user that includes the sts:AssumeRole action for the AWS Glue and SageMaker service principals in the trust policy, the data engineer can grant these permissions and avoid the access denied error. The other options are not sufficient or necessary to resolve the error. Reference:

Get started with data integration from Amazon S3 to Amazon Redshift using AWS Glue interactive sessions

Troubleshoot Errors - Amazon SageMaker

AccessDeniedException on sagemaker:CreateDomain in AWS SageMaker Studio, despite having SageMakerFullAccess

Question: 78

A company extracts approximately 1 TB of data every day from data sources such as SAP HANA, Microsoft SQL Server, MongoDB, Apache Kafka, and Amazon DynamoDB. Some of the data sources have undefined data schemas or data schemas that change.

A data engineer must implement a solution that can detect the schema for these data sources. The solution must extract, transform, and load the data to an Amazon S3 bucket. The company has a service level agreement (SLA) to load the data into the S3 bucket within 15 minutes of data creation. Which solution will meet these requirements with the LEAST operational overhead?

A. Use Amazon EMR to detect the schema and to extract, transform, and load the data into the S3 bucket. Create a pipeline in Apache Spark.

B. Use AWS Glue to detect the schema and to extract, transform, and load the data into the S3 bucket. Create a pipeline in Apache Spark.

C. Create a PySpark program in AWS Lambda to extract, transform, and load the data into the S3 bucket.

D. Create a stored procedure in Amazon Redshift to detect the schema and to extract, transform, and load the data into a Redshift Spectrum table. Access the table from Amazon S3.

Answer: B

Explanation:

AWS Glue is a fully managed service that provides a serverless data integration platform. It can automatically discover and categorize data from various sources, including SAP HANA, Microsoft SQL Server, MongoDB, Apache Kafka, and Amazon DynamoDB. It can also infer the schema of the data and store it in the AWS Glue Data Catalog, which is a central metadata repository. AWS Glue can then use the schema information to generate and run Apache Spark code to extract, transform, and load the data into an Amazon S3 bucket. AWS Glue can also monitor and optimize the performance and cost of the data pipeline, and handle any schema changes that may occur in the source data. AWS Glue can meet the SLA of loading the data into the S3 bucket within 15 minutes of data creation, as it can trigger the data pipeline based on events, schedules, or on-demand. AWS Glue has the least operational overhead among the options, as it does not require provisioning, configuring, or managing any servers or clusters. It also handles scaling, patching, and security automatically. Reference:

AWS Glue

[AWS Glue Data Catalog]

[AWS Glue Developer Guide]

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

Question: 79

A company has multiple applications that use datasets that are stored in an Amazon S3 bucket. The company has an ecommerce application that generates a dataset that contains personally identifiable information (PII). The company has an internal analytics application that does not require access to the PII.

To comply with regulations, the company must not share PII unnecessarily. A data engineer needs to implement a solution that with redact PII dynamically, based on the needs of each application that accesses the dataset.

Which solution will meet the requirements with the LEAST operational overhead?

A. Create an S3 bucket policy to limit the access each application has. Create multiple copies of the dataset. Give each dataset copy the appropriate level of redaction for the needs of the application that accesses the copy.

B. Create an S3 Object Lambda endpoint. Use the S3 Object Lambda endpoint to read data from the S3 bucket. Implement redaction logic within an S3 Object Lambda function to dynamically redact PII based on the needs of each application that accesses the data.

C. Use AWS Glue to transform the data for each application. Create multiple copies of the dataset. Give each dataset copy the appropriate level of redaction for the needs of the application that accesses the copy.

D. Create an API Gateway endpoint that has custom authorizers. Use the API Gateway endpoint to read data from the S3 bucket. Initiate a REST API call to dynamically redact PII based on the needs of each application that accesses the data.

Answer: B

Explanation:

Option B is the best solution to meet the requirements with the least operational overhead because S3 Object Lambda is a feature that allows you to add your own code to process data retrieved from S3 before returning it to

an application. S3 Object Lambda works with S3 GET requests and can modify both the object metadata and the object data. By using S3 Object Lambda, you can implement redaction logic within an S3 Object Lambda function to dynamically redact PII based on the needs of each application that accesses the data. This way, you can avoid creating and maintaining multiple copies of the dataset with different levels of redaction.

Option A is not a good solution because it involves creating and managing multiple copies of the dataset with different levels of redaction for each application. This option adds complexity and storage cost to the data protection process and requires additional resources and configuration.

Moreover, S3 bucket policies cannot enforce fine-grained data access control at the row and column level, so they are not sufficient to redact PII.

Option C is not a good solution because it involves using AWS Glue to transform the data for each application. AWS Glue is a fully managed service that can extract, transform, and load (ETL) data from various sources to various destinations, including S3. AWS Glue can also convert data to different formats, such as Parquet, which is a columnar storage format that is optimized for analytics. However, in this scenario, using AWS Glue to redact PII is not the best option because it requires creating and maintaining multiple copies of the dataset with different levels of redaction for each application. This option also adds extra time and cost to the data protection process and requires additional resources and configuration.

Option D is not a good solution because it involves creating and configuring an API Gateway endpoint that has custom authorizers. API Gateway is a service that allows you to create, publish, maintain, monitor, and secure APIs at any scale. API Gateway can also integrate with other AWS services, such as Lambda, to provide custom logic for processing requests. However, in this scenario, using API Gateway to redact PII is not the best option because it requires writing and maintaining custom code and configuration for the API endpoint, the custom authorizers, and the REST API call. This option also adds complexity and latency to the data protection process and requires additional resources and configuration.

Reference:

AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

Introducing Amazon S3 Object Lambda - Use Your Code to Process Data as It Is Being Retrieved from S3

Using Bucket Policies and User Policies - Amazon Simple Storage Service

AWS Glue Documentation

What is Amazon API Gateway? - Amazon API Gateway

Question: 80

A data engineer needs to build an extract, transform, and load (ETL) job. The ETL job will process daily incoming .csv files that users upload to an Amazon S3 bucket. The size of each S3 object is less than 100 MB. Which solution will meet these requirements MOST cost-effectively?

A. Write a custom Python application. Host the application on an Amazon Elastic Kubernetes Service (Amazon EKS) cluster.

B. Write a PySpark ETL script. Host the script on an Amazon EMR cluster.

C. Write an AWS Glue PySpark job. Use Apache Spark to transform the data.

D. Write an AWS Glue Python shell job. Use pandas to transform the data.

Answer: D

Explanation:

AWS Glue is a fully managed serverless ETL service that can handle various data sources and formats, including .csv files in Amazon S3. AWS Glue provides two types of jobs: PySpark and Python shell. PySpark jobs use Apache Spark to process large-scale data in parallel, while Python shell jobs use Python scripts to process small-scale data

in a single execution environment. For this requirement, a Python shell job is more suitable and cost-effective, as the size of each S3 object is less than 100 MB, which does not require distributed processing. A Python shell job can use pandas, a popular Python library for data analysis, to transform the .csv data as needed. The other solutions are not optimal or relevant for this requirement. Writing a custom Python application and hosting it on an Amazon EKS cluster would require more effort and resources to set up and manage the Kubernetes environment, as well as to handle the data ingestion and transformation logic. Writing a PySpark ETL script and hosting it on an Amazon EMR cluster would also incur more costs and complexity to provision and configure the EMR cluster, as well as to use Apache Spark for processing small data files. Writing an AWS Glue PySpark job would also be less efficient and economical than a Python shell job, as it would involve unnecessary overhead and charges for using Apache Spark for small data files. Reference:

AWS Glue

Working with Python Shell Jobs

pandas

[AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

Question: 81

The company stores a large volume of customer records in Amazon S3. To comply with regulations, the company must be able to access new customer records immediately for the first 30 days after the records are created. The company accesses records that are older than 30 days infrequently.

The company needs to cost-optimize its Amazon S3 storage.

Which solution will meet these requirements MOST cost-effectively?

A. Apply a lifecycle policy to transition records to S3 Standard Infrequent-Access (S3 Standard-IA) storage after 30 days.
B. Use S3 Intelligent-Tiering storage.
C. Transition records to S3 Glacier Deep Archive storage after 30 days.
D. Use S3 Standard-Infrequent Access (S3 Standard-IA) storage for all customer records.

Answer: A

Explanation:

The most cost-effective solution in this case is to apply a lifecycle policy to transition records to Amazon S3 Standard-IA storage after 30 days. Here's why:

Amazon S3 Lifecycle Policies: Amazon S3 offers lifecycle policies that allow you to automatically transition objects between different storage classes to optimize costs. For data that is frequently accessed in the first 30 days and infrequently accessed after that, transitioning from the S3 Standard storage class to S3 Standard-Infrequent Access (S3 Standard-IA) after 30 days makes the most sense. S3 Standard-IA is designed for data that is accessed less frequently but still needs to be retained, offering lower storage costs than S3 Standard with a retrieval cost for access.

Cost Optimization: S3 Standard-IA offers a lower price per GB than S3 Standard. Since the data will be accessed infrequently after 30 days, using S3 Standard-IA will lower storage costs while still allowing for immediate retrieval when necessary.

Compliance with Regulations: Since the records need to be immediately accessible for the first 30 days, the use of S3 Standard for that period ensures compliance with regulatory requirements. After 30 days, transitioning to S3 Standard-IA continues to meet access requirements for infrequent access while reducing storage costs.

Alternatives Considered:

Option B (S3 Intelligent-Tiering): While S3 Intelligent-Tiering automatically moves data between access tiers based on access patterns, it incurs a small monthly monitoring and automation charge per object. It could be a viable option, but transitioning data to S3 Standard-IA directly would be more cost-effective since the pattern of access is well-known (frequent for 30 days, infrequent thereafter).

Option C (S3 Glacier Deep Archive): Glacier Deep Archive is the lowest-cost storage class, but it is not suitable in this case because the data needs to be accessed immediately within 30 days and on an infrequent basis thereafter. Glacier Deep Archive requires hours for data retrieval, which is not acceptable for infrequent access needs.

Option D (S3 Standard-IA for all records): Using S3 Standard-IA for all records would result in higher costs for the first 30 days, as the data is frequently accessed. S3 Standard-IA incurs retrieval charges, making it less suitable for frequently accessed data.

Reference:

Amazon S3 Lifecycle Policies

S3 Storage Classes

Cost Management and Data Optimization Using Lifecycle Policies

AWS Data Engineering Documentation

Question: 82

A retail company is expanding its operations globally. The company needs to use Amazon QuickSight to accurately calculate currency exchange rates for financial reports. The company has an existing dashboard that includes a visual that is based on an analysis of a dataset that contains global currency values and exchange rates.

A data engineer needs to ensure that exchange rates are calculated with a precision of four decimal places. The calculations must be precomputed. The data engineer must materialize results in QuickSight super-fast, parallel, in-memory calculation engine (SPICE).

Which solution will meet these requirements?

A. Define and create the calculated field in the dataset.

B. Define and create the calculated field in the analysis.

C. Define and create the calculated field in the visual.

D. Define and create the calculated field in the dashboard.

Answer: A

Explanation:

Question: 83

A retail company uses an Amazon Redshift data warehouse and an Amazon S3 bucket. The company ingests retail order data into the S3 bucket every day.

The company stores all order data at a single path within the S3 bucket. The data has more than 100 columns. The company ingests the order data from a third-party application that generates more than 30 files in CSV format every day. Each CSV file is between 50 and 70 MB in size.

The company uses Amazon Redshift Spectrum to run queries that select sets of columns. Users aggregate metrics based on daily orders. Recently, users have reported that the performance of the queries has degraded. A data engineer must resolve the performance issues for the queries.

Which combination of steps will meet this requirement with LEAST developmental effort? (Select TWO.)

A. Configure the third-party application to create the files in a columnar format.
B. Develop an AWS Glue ETL job to convert the multiple daily CSV files to one file for each day.
C. Partition the order data in the S3 bucket based on order date.
D. Configure the third-party application to create the files in JSON format.
E. Load the JSON data into the Amazon Redshift table in a SUPER type column.

Answer: A, C

Explanation:
The performance issue in Amazon Redshift Spectrum queries arises due to the nature of CSV files, which are row-based storage formats. Spectrum is more optimized for columnar formats, which significantly improve performance by reducing the amount of data scanned. Also, partitioning data based on relevant columns like order date can further reduce the amount of data scanned, as queries can focus only on the necessary partitions.

A . Configure the third-party application to create the files in a columnar format:
Columnar formats (like Parquet or ORC) store data in a way that is optimized for analytical queries because they allow queries to scan only the columns required, rather than scanning all columns in a row-based format like CSV.
Amazon Redshift Spectrum works much more efficiently with columnar formats, reducing the amount of data that needs to be scanned, which improves query performance.
Reference: Amazon Redshift Spectrum and Columnar File Formats

C . Partition the order data in the S3 bucket based on order date:
Partitioning the data on columns like order date allows Redshift Spectrum to skip scanning unnecessary partitions, leading to improved query performance.
By organizing data into partitions, you minimize the number of files Spectrum has to read, further optimizing performance.
Reference: Best Practices for Amazon Redshift Spectrum Performance

Alternatives Considered:
B (Develop an AWS Glue ETL job): While consolidating files can improve performance by reducing the number of small files (which can be inefficient to process), it adds additional ETL complexity. Switching to a columnar format (Option A) and partitioning (Option C) provides more significant performance improvements with less development effort.
D and E (JSON-related options): Using JSON format or the SUPER type in Redshift introduces complexity and isn't as efficient as the proposed solutions, especially since JSON is not a columnar format.

Reference:
Amazon Redshift Spectrum Documentation
Columnar Formats and Data Partitioning in S3

Question: 84

A company ingests data from multiple data sources and stores the data in an Amazon S3 bucket. An AWS Glue extract, transform, and load (ETL) job transforms the data and writes the transformed data to an Amazon S3 based data lake. The company uses Amazon Athena to query the data that is in the data lake.
The company needs to identify matching records even when the records do not have a common unique identifier.
Which solution will meet this requirement?

A. Use Amazon Made pattern matching as part of the ETL job.
B. Train and use the AWS Glue PySpark Filter class in the ETL job.
C. Partition tables and use the ETL job to partition the data on a unique identifier.
D. Train and use the AWS Lake Formation FindMatches transform in the ETL job.

Answer: D

Explanation:
The problem described requires identifying matching records even when there is no unique identifier. AWS Lake Formation FindMatches is designed for this purpose. It uses machine learning (ML) to deduplicate and find matching records in datasets that do not share a common identifier. D. Train and use the AWS Lake Formation FindMatches transform in the ETL job:
FindMatches is a transform available in AWS Lake Formation that uses ML to discover duplicate records or related records that might not have a common unique identifier.
It can be integrated into an AWS Glue ETL job to perform deduplication or matching tasks. FindMatches is highly effective in scenarios where records do not share a key, such as customer records from different sources that need to be merged or reconciled.
Reference: AWS Lake Formation FindMatches
Alternatives Considered:
A (Amazon Made pattern matching): Amazon Made is not a service in AWS, and pattern matching typically refers to regular expressions, which are not suitable for deduplication without a common identifier.
B (AWS Glue PySpark Filter class): PySpark's Filter class can help refine datasets, but it does not offer the ML-based matching capabilities required to find matches between records without unique identifiers.
C (Partition tables on a unique identifier): Partitioning requires a unique identifier, which the question states is unavailable.
Reference:
AWS Glue Documentation on Lake Formation FindMatches
FindMatches in AWS Lake Formation

Question: 85

A company stores its processed data in an S3 bucket. The company has a strict data access policy. The company uses IAM roles to grant teams within the company different levels of access to the S3 bucket.
The company wants to receive notifications when a user violates the data access policy. Each notification must include the username of the user who violated the policy.
Which solution will meet these requirements?

A. Use AWS Config rules to detect violations of the data access policy. Set up compliance alarms.
B. Use Amazon CloudWatch metrics to gather object-level metrics. Set up CloudWatch alarms.
C. Use AWS CloudTrail to track object-level events for the S3 bucket. Forward events to Amazon CloudWatch to set up CloudWatch alarms.
D. Use Amazon S3 server access logs to monitor access to the bucket. Forward the access logs to an Amazon CloudWatch log group. Use metric filters on the log group to set up CloudWatch alarms.

Answer: C

Explanation:

The requirement is to detect violations of data access policies and receive notifications with the username of the violator. AWS CloudTrail can provide object-level tracking for S3 to capture detailed API actions on specific S3 objects, including the user who performed the action.

AWS CloudTrail:

CloudTrail can monitor API calls made to an S3 bucket, including object-level API actions such as GetObject, PutObject, and DeleteObject. This will help detect access violations based on the API calls made by different users.

CloudTrail logs include details such as the user identity, which is essential for meeting the requirement of including the username in notifications.

The CloudTrail logs can be forwarded to Amazon CloudWatch to trigger alarms based on certain access patterns (e.g., violations of specific policies).

Reference: Monitoring Amazon S3 Activity Using AWS CloudTrail

Amazon CloudWatch:

By forwarding CloudTrail logs to CloudWatch, you can set up alarms that are triggered when a specific condition is met, such as unauthorized access or policy violations. The alarm can include detailed information from the CloudTrail log, including the username.

Alternatives Considered:

A (AWS Config rules): While AWS Config can track resource configurations and compliance, it does not provide real-time, detailed tracking of object-level events like CloudTrail does.

B (CloudWatch metrics): CloudWatch does not gather object-level metrics for S3 directly. For this use case, CloudTrail provides better granularity.

D (S3 server access logs): S3 server access logs can monitor access, but they do not provide the realtime monitoring and alerting features that CloudTrail with CloudWatch alarms offer. They also do not include API-level granularity like CloudTrail.

Reference:

AWS CloudTrail Integration with S3

Amazon CloudWatch Alarms

Question: 86

A company stores logs in an Amazon S3 bucket. When a data engineer attempts to access several log files, the data engineer discovers that some files have been unintentionally deleted.

The data engineer needs a solution that will prevent unintentional file deletion in the future.

Which solution will meet this requirement with the LEAST operational overhead?

A. Manually back up the S3 bucket on a regular basis.
B. Enable S3 Versioning for the S3 bucket.
C. Configure replication for the S3 bucket.
D. Use an Amazon S3 Glacier storage class to archive the data that is in the S3 bucket.

Answer: B

Explanation:

To prevent unintentional file deletions and meet the requirement with minimal operational overhead, enabling S3 Versioning is the best solution.

S3 Versioning:

S3 Versioning allows multiple versions of an object to be stored in the same S3 bucket. When a file is deleted or overwritten, S3 preserves the previous versions, which means you can recover from accidental deletions or

modifications.

Enabling versioning requires minimal overhead, as it is a bucket-level setting and does not require additional backup processes or data replication.

Users can recover specific versions of files that were unintentionally deleted, meeting the needs of the data engineer to avoid accidental data loss.

Reference: Amazon S3 Versioning

Alternatives Considered:

A (Manual backups): Manually backing up the bucket requires higher operational effort and maintenance compared to enabling S3 Versioning, which is automated.

C (S3 Replication): Replication ensures data is copied to another bucket but does not provide protection against accidental deletion. It would increase operational costs without solving the core issue of accidental deletion.

D (S3 Glacier): Storing data in Glacier provides long-term archival storage but is not designed to prevent accidental deletion. Glacier is also more suitable for archival and infrequently accessed data, not for active logs.

Reference:

Amazon S3 Versioning Documentation

S3 Data Protection Best Practices

Question: 87

A company currently uses a provisioned Amazon EMR cluster that includes general purpose Amazon EC2 instances. The EMR cluster uses EMR managed scaling between one to five task nodes for the company's long-running Apache Spark extract, transform, and load (ETL) job. The company runs the ETL job every day.

When the company runs the ETL job, the EMR cluster quickly scales up to five nodes. The EMR cluster often reaches maximum CPU usage, but the memory usage remains under 30%.

The company wants to modify the EMR cluster configuration to reduce the EMR costs to run the daily ETL job. Which solution will meet these requirements MOST cost-effectively?

A. Increase the maximum number of task nodes for EMR managed scaling to 10.
B. Change the task node type from general purpose EC2 instances to memory optimized EC2 instances.
C. Switch the task node type from general purpose EC2 instances to compute optimized EC2 instances.
D. Reduce the scaling cooldown period for the provisioned EMR cluster.

Answer: C

Explanation: The company's Apache Spark ETL job on Amazon EMR uses high CPU but low memory, meaning that compute-optimized EC2 instances would be the most cost-effective choice. These instances are designed for high-performance compute applications, where CPU usage is high, but memory needs are minimal, which is exactly the case here.

Compute Optimized Instances:

Compute-optimized instances, such as the C5 series, provide a higher ratio of CPU to memory, which is more suitable for jobs with high CPU usage and relatively low memory consumption.

Switching from general-purpose EC2 instances to compute-optimized instances can reduce costs while improving performance, as these instances are optimized for workloads like Spark jobs that perform a lot of computation.

Reference: Amazon EC2 Compute Optimized Instances

Managed Scaling: The EMR cluster's scaling is currently managed between 1 and 5 nodes, so changing the instance type will leverage the current scaling strategy but optimize it for the workload. Alternatives Considered:

A (Increase task nodes to 10): Increasing the number of task nodes would increase costs without necessarily improving performance. Since memory usage is low, the bottleneck is more likely the CPU, which compute-optimized instances can handle better.

B (Memory optimized instances): Memory-optimized instances are not suitable since the current job is CPU-bound, and memory usage remains low (under 30%).

D (Reduce scaling cooldown): This could marginally improve scaling speed but does not address the need for cost optimization and improved CPU performance.

Reference:

Amazon EMR Cluster Optimization

Compute Optimized EC2 Instances

Question: 88

A company is building a data stream processing application. The application runs in an Amazon Elastic Kubernetes Service (Amazon EKS) cluster. The application stores processed data in an Amazon DynamoDB table.

The company needs the application containers in the EKS cluster to have secure access to the DynamoDB table. The company does not want to embed AWS credentials in the containers. Which solution will meet these requirements?

A. Store the AWS credentials in an Amazon S3 bucket. Grant the EKS containers access to the S3 bucket to retrieve the credentials.

B. Attach an IAM role to the EKS worker nodes. Grant the IAM role access to DynamoDB. Use the IAM role to set up IAM roles service accounts (IRSA) functionality.

C. Create an IAM user that has an access key to access the DynamoDB table. Use environment variables in the EKS containers to store the IAM user access key data.

D. Create an IAM user that has an access key to access the DynamoDB table. Use Kubernetes secrets that are mounted in a volume of the EKS cluster nodes to store the user access key data.

Answer: B

Explanation:

In this scenario, the company is using Amazon Elastic Kubernetes Service (EKS) and wants secure access to DynamoDB without embedding credentials inside the application containers. The best practice is to use IAM roles for service accounts (IRSA), which allows assigning IAM roles to Kubernetes service accounts. This lets the EKS pods assume specific IAM roles securely, without the need to store credentials in containers.

IAM Roles for Service Accounts (IRSA):

With IRSA, each pod in the EKS cluster can assume an IAM role that grants access to DynamoDB without needing to manage long-term credentials. The IAM role can be attached to the service account associated with the pod.

This ensures least privilege access, improving security by preventing credentials from being embedded in the containers.

Reference: IAM Roles for Service Accounts (IRSA)

Alternatives Considered:

A (Storing AWS credentials in S3): Storing AWS credentials in S3 and retrieving them introduces security risks and violates the principle of not embedding credentials.

C (IAM user access keys in environment variables): This also embeds credentials, which is not recommended.

D (Kubernetes secrets): Storing user access keys as secrets is an option, but it still involves handling long-term credentials manually, which is less secure than using IRSA.

Reference:

IAM Best Practices for Amazon EKS

Secure Access to DynamoDB from EKS

Question: 89

A technology company currently uses Amazon Kinesis Data Streams to collect log data in real time. The company wants to use Amazon Redshift for downstream real-time queries and to enrich the log data.

Which solution will ingest data into Amazon Redshift with the LEAST operational overhead?

A. Set up an Amazon Data Firehose delivery stream to send data to a Redshift provisioned cluster table.

B. Set up an Amazon Data Firehose delivery stream to send data to Amazon S3. Configure a Redshift provisioned cluster to load data every minute.

C. Configure Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to send data directly to a Redshift provisioned cluster table.

D. Use Amazon Redshift streaming ingestion from Kinesis Data Streams and to present data as a materialized view.

Answer: D

Explanation:

The most efficient and low-operational-overhead solution for ingesting data into Amazon Redshift from Amazon Kinesis Data Streams is to use Amazon Redshift streaming ingestion. This feature allows Redshift to directly ingest streaming data from Kinesis Data Streams and process it in real time.

Amazon Redshift Streaming Ingestion:

Redshift supports native streaming ingestion from Kinesis Data Streams, allowing real-time data to be queried using materialized views.

This solution reduces operational complexity because you don't need intermediary services like Amazon Kinesis Data Firehose or S3 for batch loading.

Reference: Amazon Redshift Streaming Ingestion

Alternatives Considered:

A (Data Firehose to Redshift): This option is more suitable for batch processing but incurs additional operational overhead with the Firehose setup.

B (Firehose to S3): This involves an intermediate step, which adds complexity and delays the real time requirement.

C (Managed Service for Apache Flink): This would work but introduces unnecessary complexity compared to Redshift's native streaming ingestion.

Reference:

Amazon Redshift Streaming Ingestion from Kinesis

Materialized Views in Redshift

Question: 90

A banking company uses an application to collect large volumes of transactional dat

a. The company uses Amazon Kinesis Data Streams for real-time analytics. The company's application uses the PutRecord action to send data to Kinesis Data Streams.

A data engineer has observed network outages during certain times of day. The data engineer wants to configure exactly-once delivery for the entire processing pipeline.

Which solution will meet this requirement?

A. Design the application so it can remove duplicates during processing by embedding a unique ID in each record at the source.

B. Update the checkpoint configuration of the Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) data collection application to avoid duplicate processing of events.

C. Design the data source so events are not ingested into Kinesis Data Streams multiple times.

D. Stop using Kinesis Data Streams. Use Amazon EMR instead. Use Apache Flink and Apache Spark Streaming in Amazon EMR.

Answer: A

Explanation:

For exactly-once delivery and processing in Amazon Kinesis Data Streams, the best approach is to design the application so that it handles idempotency. By embedding a unique ID in each record, the application can identify and remove duplicate records during processing.

Exactly-Once Processing:

Kinesis Data Streams does not natively support exactly-once processing. Therefore, idempotency should be designed into the application, ensuring that each record has a unique identifier so that the same event is processed only once, even if it is ingested multiple times.

This pattern is widely used for achieving exactly-once semantics in distributed systems.

Reference: [Building Idempotent Applications with Kinesis]

Alternatives Considered:

B (Checkpoint configuration): While updating the checkpoint configuration can help with some aspects of duplicate processing, it is not a full solution for exactly-once delivery.

C (Design data source): Ensuring events are not ingested multiple times is ideal, but network outages can make this difficult, and it doesn't guarantee exactly-once delivery.

D (Using EMR): While using EMR with Flink or Spark could work, it introduces unnecessary complexity compared to handling idempotency at the application level.

Reference:

[Amazon Kinesis Best Practices for Exactly-Once Processing]

[Achieving Idempotency with Amazon Kinesis]

Question: 91

A company has three subsidiaries. Each subsidiary uses a different data warehousing solution. The first subsidiary hosts its data warehouse in Amazon Redshift. The second subsidiary uses Teradata Vantage on AWS. The third subsidiary uses Google BigQuery.

The company wants to aggregate all the data into a central Amazon S3 data lake. The company wants to use Apache Iceberg as the table format.

A data engineer needs to build a new pipeline to connect to all the data sources, run transformations by using each source engine, join the data, and write the data to Iceberg.

Which solution will meet these requirements with the LEAST operational effort?

A. Use native Amazon Redshift, Teradata, and BigQuery connectors to build the pipeline in AWS Glue. Use native AWS Glue transforms to join the data. Run a Merge operation on the data lake Iceberg table.

B. Use the Amazon Athena federated query connectors for Amazon Redshift, Teradata, and BigQuery to build the pipeline in Athena. Write a SQL query to read from all the data sources, join the data, and run a Merge operation on the data lake Iceberg table.

C. Use the native Amazon Redshift connector, the Java Database Connectivity (JDBC) connector for Teradata, and the open source Apache Spark BigQuery connector to build the pipeline in Amazon EMR. Write code in PySpark to join the data. Run a Merge operation on the data lake Iceberg table.

D. Use the native Amazon Redshift, Teradata, and BigQuery connectors in Amazon Appflow to write data to Amazon S3 and AWS Glue Data Catalog. Use Amazon Athena to join the data. Run a Merge operation on the data lake Iceberg table.

Answer: B

Explanation:

Amazon Athena provides federated query connectors that allow querying multiple data sources, such as Amazon Redshift, Teradata, and Google BigQuery, without needing to extract the data from the original source. This solution is optimal because it offers the least operational effort by avoiding complex data movement and transformation processes.

Amazon Athena Federated Queries:

Athena's federated queries allow direct querying of data stored across multiple sources, including Amazon Redshift, Teradata, and BigQuery. With Athena's support for Apache Iceberg, the company can easily run a Merge operation on the Iceberg table.

The solution reduces complexity by centralizing the query execution and transformation process in Athena using SQL queries.

Reference: Amazon Athena Federated Query

Alternatives Considered:

A (AWS Glue pipeline): This would work but requires more operational effort to manage and transform the data in AWS Glue.

C (Amazon EMR): Using EMR and writing PySpark code introduces more operational overhead and complexity compared to a SQL-based solution in Athena.

D (Amazon AppFlow): AppFlow is more suitable for transferring data between services but is not as efficient for transformations and joins as Athena federated queries.

Reference:

Amazon Athena Documentation

Federated Queries in Amazon Athena

Question: 92

A company has a data lake in Amazon S3. The company collects AWS CloudTrail logs for multiple applications. The company stores the logs in the data lake, catalogs the logs in AWS Glue, and partitions the logs based on the year. The company uses Amazon Athena to analyze the logs. Recently, customers reported that a query on one of the Athena tables did not return any dat a. A data engineer must resolve the issue.

Which combination of troubleshooting steps should the data engineer take? (Select TWO.)

A. Confirm that Athena is pointing to the correct Amazon S3 location.

B. Increase the query timeout duration.

C. Use the MSCK REPAIR TABLE command.

D. Restart Athena.

E. Delete and recreate the problematic Athena table.

Answer: A, C

Explanation:

The problem likely arises from Athena not being able to read from the correct S3 location or missing partitions. The two most relevant troubleshooting steps involve checking the S3 location and repairing the table metadata.

A . Confirm that Athena is pointing to the correct Amazon S3 location:

One of the most common issues with missing data in Athena queries is that the query is pointed to an incorrect or outdated S3 location. Checking the S3 path ensures Athena is querying the correct data.

Reference: Amazon Athena Troubleshooting

C . Use the MSCK REPAIR TABLE command:

When new partitions are added to the S3 bucket without being reflected in the Glue Data Catalog, Athena queries will not return data from those partitions. The MSCK REPAIR TABLE command updates the Glue Data Catalog with the latest partitions.

Reference: MSCK REPAIR TABLE Command

Alternatives Considered:

B (Increase query timeout): Timeout issues are unrelated to missing data.

D (Restart Athena): Athena does not require restarting.

E (Delete and recreate table): This introduces unnecessary overhead when the issue can be resolved by repairing the table and confirming the S3 location.

Reference:

Athena Query Fails to Return Data

Question: 93

A company is using an AWS Transfer Family server to migrate data from an on-premises environment to AWS. Company policy mandates the use of TLS 1.2 or above to encrypt the data in transit. Which solution will meet these requirements?

A. Generate new SSH keys for the Transfer Family server. Make the old keys and the new keys available for use.

B. Update the security group rules for the on-premises network to allow only connections that use TLS 1.2 or above.

C. Update the security policy of the Transfer Family server to specify a minimum protocol version of TLS 1.2.

D. Install an SSL certificate on the Transfer Family server to encrypt data transfers by using TLS 1.2.

Answer: C

Explanation:

The AWS Transfer Family server's security policy can be updated to enforce TLS 1.2 or higher, ensuring compliance with company policy for encrypted data transfers.

AWS Transfer Family Security Policy:

AWS Transfer Family supports setting a minimum TLS version through its security policy configuration. This

ensures that only connections using TLS 1.2 or above are allowed. Reference: AWS Transfer Family Security Policy

Alternatives Considered:

A (Generate new SSH keys): SSH keys are unrelated to TLS and do not enforce encryption protocols like TLS 1.2.

B (Update security group rules): Security groups control IP-level access, not TLS versions.

D (Install SSL certificate): SSL certificates ensure secure connections, but the TLS version is controlled via the security policy.

Reference:

AWS Transfer Family Documentation

Question: 94

A data engineer configured an AWS Glue Data Catalog for data that is stored in Amazon S3 buckets. The data engineer needs to configure the Data Catalog to receive incremental updates.

The data engineer sets up event notifications for the S3 bucket and creates an Amazon Simple Queue Service (Amazon SQS) queue to receive the S3 events.

Which combination of steps should the data engineer take to meet these requirements with LEAST operational overhead? (Select TWO.)

A. Create an S3 event-based AWS Glue crawler to consume events from the SQS queue.

B. Define a time-based schedule to run the AWS Glue crawler, and perform incremental updates to the Data Catalog.

C. Use an AWS Lambda function to directly update the Data Catalog based on S3 events that the SQS queue receives.

D. Manually initiate the AWS Glue crawler to perform updates to the Data Catalog when there is a change in the S3 bucket.

E. Use AWS Step Functions to orchestrate the process of updating the Data Catalog based on 53 events that the SQS queue receives.

Answer: A, C

Explanation:

The requirement is to update the AWS Glue Data Catalog incrementally based on S3 events. Using an S3 event-based approach is the most automated and operationally efficient solution.

A . Create an S3 event-based AWS Glue crawler:

An event-based Glue crawler can automatically update the Data Catalog when new data arrives in the S3 bucket. This ensures incremental updates with minimal operational overhead.

Reference: AWS Glue Event-Driven Crawlers

C . Use an AWS Lambda function to directly update the Data Catalog:

Lambda can be triggered by S3 events delivered to the SQS queue and can directly update the Glue Data Catalog, ensuring that new data is reflected in near real-time without running a full crawler. Reference: Automating AWS Glue Data Catalog Updates

Alternatives Considered:

B (Time-based schedule): Scheduling a crawler to run periodically adds unnecessary latency and operational overhead.

D (Manual crawler initiation): Manually starting the crawler defeats the purpose of automation.

E (AWS Step Functions): Step Functions add complexity that is not needed when Lambda can handle the updates directly.

Reference:
AWS Glue Event-Driven Crawlers
Using AWS Lambda to Update Glue Catalog

Question: 95

A company uploads .csv files to an Amazon S3 bucket. The company's data platform team has set up an AWS Glue crawler to perform data discovery and to create the tables and schemas.
An AWS Glue job writes processed data from the tables to an Amazon Redshift database. The AWS Glue job handles column mapping and creates the Amazon Redshift tables in the Redshift database appropriately.
If the company reruns the AWS Glue job for any reason, duplicate records are introduced into the Amazon Redshift tables. The company needs a solution that will update the Redshift tables without duplicates.
Which solution will meet these requirements?

A. Modify the AWS Glue job to copy the rows into a staging Redshift table. Add SQL commands to update the existing rows with new values from the staging Redshift table.
B. Modify the AWS Glue job to load the previously inserted data into a MySQL database. Perform an upsert operation in the MySQL database. Copy the results to the Amazon Redshift tables.
C. Use Apache Spark's DataFrame dropDuplicates() API to eliminate duplicates. Write the data to the Redshift tables.
D. Use the AWS Glue ResolveChoice built-in transform to select the value of the column from the most recent record.

Answer: A

Explanation:
To avoid duplicate records in Amazon Redshift, the most effective solution is to perform the ETL in a way that first loads the data into a staging table and then uses SQL commands like MERGE or UPDATE to insert new records and update existing records without introducing duplicates.
Using Staging Tables in Redshift:
The AWS Glue job can write data to a staging table in Redshift. Once the data is loaded, SQL commands can be executed to compare the staging data with the target table and update or insert records appropriately. This ensures no duplicates are introduced during re-runs of the Glue job. Reference: Amazon Redshift Best Practices
Alternatives Considered:
B (MySQL upsert): This introduces unnecessary complexity by involving another database (MySQL).
C (Spark dropDuplicates): While Spark can eliminate duplicates, handling duplicates at the Redshift level with a staging table is a more reliable and Redshift-native solution.
D (AWS Glue ResolveChoice): The ResolveChoice transform in Glue helps with column conflicts but does not handle record-level duplicates effectively.
Reference:
Amazon Redshift MERGE Statements
Staging Tables in Amazon Redshift

Question: 96

A financial company recently added more features to its mobile app. The new features required the company to create a new topic in an existing Amazon Managed Streaming for Apache Kafka (Amazon MSK) cluster.

A few days after the company added the new topic, Amazon CloudWatch raised an alarm on the RootDiskUsed metric for the MSK cluster.

How should the company address the CloudWatch alarm?

A. Expand the storage of the MSK broker. Configure the MSK cluster storage to expand automatically.
B. Expand the storage of the Apache ZooKeeper nodes.
C. Update the MSK broker instance to a larger instance type. Restart the MSK cluster.
D. Specify the Target-Volume-in-GiB parameter for the existing topic.

Answer: A

Explanation:

The RootDiskUsed metric for the MSK cluster indicates that the storage on the broker is reaching its capacity. The best solution is to expand the storage of the MSK broker and enable automatic storage expansion to prevent future alarms.

Expand MSK Broker Storage:

AWS Managed Streaming for Apache Kafka (MSK) allows you to expand the broker storage to accommodate growing data volumes. Additionally, auto-expansion of storage can be configured to ensure that storage grows automatically as the data increases.

Reference: Amazon MSK Cluster Storage Expansion

Alternatives Considered:

B (Expand Zookeeper storage): Zookeeper is responsible for managing Kafka metadata and not for storing data, so increasing Zookeeper storage won't resolve the root disk issue.

C (Update instance type): Changing the instance type would increase computational resources but not directly address the storage problem.

D (Target-Volume-in-GiB): This parameter is irrelevant for the existing topic and will not solve the storage issue.

Reference:

Amazon MSK Storage Auto Scaling

Question: 97

A company hosts its applications on Amazon EC2 instances. The company must use SSL/TLS connections that encrypt data in transit to communicate securely with AWS infrastructure that is managed by a customer.

A data engineer needs to implement a solution to simplify the generation, distribution, and rotation of digital certificates. The solution must automatically renew and deploy SSL/TLS certificates. Which solution will meet these requirements with the LEAST operational overhead?

A. Store self-managed certificates on the EC2 instances.
B. Use AWS Certificate Manager (ACM).
C. Implement custom automation scripts in AWS Secrets Manager.
D. Use Amazon Elastic Container Service (Amazon ECS) Service Connect.

Answer: B

Explanation:

The best solution for managing SSL/TLS certificates on EC2 instances with minimal operational overhead is to use AWS Certificate Manager (ACM). ACM simplifies certificate management by automating the provisioning, renewal, and deployment of certificates.

AWS Certificate Manager (ACM):

ACM manages SSL/TLS certificates for EC2 and other AWS resources, including automatic certificate renewal. This reduces the need for manual management and avoids operational complexity.

ACM also integrates with other AWS services to simplify secure connections between AWS infrastructure and customer-managed environments.

Reference: AWS Certificate Manager

Alternatives Considered:

A (Self-managed certificates): Managing certificates manually on EC2 instances increases operational overhead and lacks automatic renewal.

C (Secrets Manager automation): While Secrets Manager can store keys and certificates, it requires custom automation for rotation and does not handle SSL/TLS certificates directly.

D (ECS Service Connect): This is unrelated to SSL/TLS certificate management and would not address the operational need.

Reference:

AWS Certificate Manager Documentation

Question: 98

A company saves customer data to an Amazon S3 bucket. The company uses server-side encryption with AWS KMS keys (SSE-KMS) to encrypt the bucket. The dataset includes personally identifiable information (PII) such as social security numbers and account details.

Data that is tagged as PII must be masked before the company uses customer data for analysis. Some users must have secure access to the PII data during the preprocessing phase. The company needs a low-maintenance solution to mask and secure the PII data throughout the entire engineering pipeline.

Which combination of solutions will meet these requirements? (Select TWO.)

A. Use AWS Glue DataBrew to perform extract, transform, and load (ETL) tasks that mask the PII data before analysis.

B. Use Amazon GuardDuty to monitor access patterns for the PII data that is used in the engineering pipeline.

C. Configure an Amazon Made discovery job for the S3 bucket.

D. Use AWS Identity and Access Management (IAM) to manage permissions and to control access to the PII data.

E. Write custom scripts in an application to mask the PII data and to control access.

Answer: A, D

Explanation:

To address the requirement of masking PII data and ensuring secure access throughout the data pipeline, the combination of AWS Glue DataBrew and IAM provides a low-maintenance solution. A . AWS Glue DataBrew for Masking:

AWS Glue DataBrew provides a visual tool to perform data transformations, including masking PII data. It allows

for easy configuration of data transformation tasks without requiring manual coding, making it ideal for this use case.

Reference: AWS Glue DataBrew

D . AWS Identity and Access Management (IAM):

Using IAM policies allows fine-grained control over access to PII data, ensuring that only authorized users can view or process sensitive data during the pipeline stages.

Reference: AWS IAM Best Practices

Alternatives Considered:

B (Amazon GuardDuty): GuardDuty is for threat detection and does not handle data masking or access control for PII.

C (Amazon Macie): Macie can help discover sensitive data but does not handle the masking of PII or access control.

E (Custom scripts): Custom scripting increases the operational burden compared to a built-in solution like DataBrew.

Reference:

AWS Glue DataBrew for Data Masking

IAM Policies for PII Access Control

Question: 99

A data engineer needs to onboard a new data producer into AWS. The data producer needs to migrate data products to AWS.

The data producer maintains many data pipelines that support a business application. Each pipeline must have service accounts and their corresponding credentials. The data engineer must establish a secure connection from the data producer's on-premises data center to AWS. The data engineer must not use the public internet to transfer data from an on-premises data center to AWS.

Which solution will meet these requirements?

A. Instruct the new data producer to create Amazon Machine Images (AMIs) on Amazon Elastic Container Service (Amazon ECS) to store the code base of the application. Create security groups in a public subnet that allow connections only to the on-premises data center.

B. Create an AWS Direct Connect connection to the on-premises data center. Store the service account credentials in AWS Secrets manager.

C. Create a security group in a public subnet. Configure the security group to allow only connections from the CIDR blocks that correspond to the data producer. Create Amazon S3 buckets than contain presigned URLS that have one-day expiration dates.

D. Create an AWS Direct Connect connection to the on-premises data center. Store the application keys in AWS Secrets Manager. Create Amazon S3 buckets that contain resigned URLS that have one- day expiration dates.

Answer: B

Explanation:

For secure migration of data from an on-premises data center to AWS without using the public internet, AWS Direct Connect is the most secure and reliable method. Using Secrets Manager to store service account credentials ensures that the credentials are managed securely with automatic rotation.

AWS Direct Connect:

Direct Connect establishes a dedicated, private connection between the on-premises data center and AWS, avoiding the public internet. This is ideal for secure, high-speed data transfers.

Reference: AWS Direct Connect

AWS Secrets Manager:

Secrets Manager securely stores and rotates service account credentials, reducing operational overhead while ensuring security.

Reference: AWS Secrets Manager

Alternatives Considered:

A (ECS with security groups): This does not address the need for a secure, private connection from the on-premises data center.

C (Public subnet with presigned URLs): This involves using the public internet, which does not meet the requirement.

D (Direct Connect with presigned URLs): While Direct Connect is correct, presigned URLs with short expiration dates are unnecessary for this use case.

Reference:

AWS Direct Connect Documentation

AWS Secrets Manager Documentation

Question: 100

A company uses AWS Glue Data Catalog to index data that is uploaded to an Amazon S3 bucket every day. The company uses a daily batch processes in an extract, transform, and load (ETL) pipeline to upload data from external sources into the S3 bucket.

The company runs a daily report on the S3 dat

a. Some days, the company runs the report before all the daily data has been uploaded to the S3 bucket. A data engineer must be able to send a message that identifies any incomplete data to an existing Amazon Simple Notification Service (Amazon SNS) topic.

Which solution will meet this requirement with the LEAST operational overhead?

A. Create data quality checks for the source datasets that the daily reports use. Create a new AWS managed Apache Airflow cluster. Run the data quality checks by using Airflow tasks that run data quality queries on the columns data type and the presence of null values. Configure Airflow Directed Acyclic Graphs (DAGs) to send an email notification that informs the data engineer about the incomplete datasets to the SNS topic.

B. Create data quality checks on the source datasets that the daily reports use. Create a new Amazon EMR cluster. Use Apache Spark SQL to create Apache Spark jobs in the EMR cluster that run data quality queries on the columns data type and the presence of null values. Orchestrate the ETL pipeline by using an AWS Step Functions workflow. Configure the workflow to send an email notification that informs the data engineer about the incomplete datasets to the SNS topic.

C. Create data quality checks on the source datasets that the daily reports use. Create data quality actions by using AWS Glue workflows to confirm the completeness and consistency of the datasets. Configure the data quality actions to create an event in Amazon EventBridge if a dataset is incomplete. Configure EventBridge to send the event that informs the data engineer about the incomplete datasets to the Amazon SNS topic.

D. Create AWS Lambda functions that run data quality queries on the columns data type and the presence of null values. Orchestrate the ETL pipeline by using an AWS Step Functions workflow that runs the Lambda functions. Configure the Step Functions workflow to send an email notification that informs the data engineer about the incomplete datasets to the SNS topic.

Answer: C

Explanation:

AWS Glue workflows are designed to orchestrate the ETL pipeline, and you can create data quality checks to ensure the uploaded datasets are complete before running reports. If there is an issue with the data, AWS Glue workflows can trigger an Amazon EventBridge event that sends a message to an SNS topic.

AWS Glue Workflows:

AWS Glue workflows allow users to automate and monitor complex ETL processes. You can include data quality actions to check for null values, data types, and other consistency checks.

In the event of incomplete data, an EventBridge event can be generated to notify via SNS. Reference: AWS Glue Workflows

Alternatives Considered:

A (Airflow cluster): Managed Airflow introduces more operational overhead and complexity compared to Glue workflows.

B (EMR cluster): Setting up an EMR cluster is also more complex compared to the Glue-centric solution.

D (Lambda functions): While Lambda functions can work, using Glue workflows offers a more integrated and lower operational overhead solution.

Reference:

AWS Glue Workflow Documentation

Question: 101

Two developers are working on separate application releases. The developers have created feature branches named Branch A and Branch B by using a GitHub repository's master branch as the source. The developer for Branch A deployed code to the production system. The code for Branch B will merge into a master branch in the following week's scheduled application release.

Which command should the developer for Branch B run before the developer raises a pull request to the master branch?

A. git diff branchB master
git commit -m <message>

B. git pull master

C. git rebase master

D. git fetch -b master

Answer: C

Explanation:

To ensure that Branch B is up to date with the latest changes in the master branch before submitting a pull request, the correct approach is to perform a git rebase. This command rewrites the commit history so that Branch B will be based on the latest changes in the master branch.

git rebase master:

This command moves the commits of Branch B to be based on top of the latest state of the master branch. It allows the developer to resolve any conflicts and create a clean history.

Reference: Git Rebase Documentation

Alternatives Considered:

A (git diff): This will only show differences between Branch B and master but won't resolve conflicts or bring Branch B up to date.

B (git pull master): Pulling the master branch directly does not offer the same clean history management as rebase.

D (git fetch -b): This is an incorrect command.

Reference:

Git Rebase Best Practices

Question: 102

A company needs to load customer data that comes from a third party into an Amazon Redshift data warehouse. The company stores order data and product data in the same data warehouse. The company wants to use the combined dataset to identify potential new customers.

A data engineer notices that one of the fields in the source data includes values that are in JSON format.

How should the data engineer load the JSON data into the data warehouse with the LEAST effort?

A. Use the SUPER data type to store the data in the Amazon Redshift table.

B. Use AWS Glue to flatten the JSON data and ingest it into the Amazon Redshift table.

C. Use Amazon S3 to store the JSON data. Use Amazon Athena to query the data.

D. Use an AWS Lambda function to flatten the JSON data. Store the data in Amazon S3.

Answer: A

Explanation:

In Amazon Redshift, the SUPER data type is designed specifically to handle semi-structured data like JSON, Parquet, ORC, and others. By using the SUPER data type, Redshift can ingest and query JSON data without requiring complex data flattening processes, thus reducing the amount of preprocessing required before loading the data. The SUPER data type also works seamlessly with Redshift Spectrum, enabling complex queries that can combine both structured and semi-structured datasets, which aligns with the company's need to use combined datasets to identify potential new customers. Using the SUPER data type also allows for automatic parsing and query processing of nested data structures through Amazon Redshift's PARTITION BY and JSONPATH expressions, which makes this option the most efficient approach with the least effort involved. This reduces the overhead associated with using tools like AWS Glue or Lambda for data transformation.

Reference:

Amazon Redshift Documentation - SUPER Data Type

AWS Certified Data Engineer - Associate Training: Building Batch Data Analytics Solutions on AWS AWS Certified Data Engineer - Associate Study Guide

By directly leveraging the capabilities of Redshift with the SUPER data type, the data engineer ensures streamlined JSON ingestion with minimal effort while maintaining query efficiency.

Question: 103

A company stores employee data in Amazon Redshift A table named Employee uses columns named Region ID, Department ID, and Role ID as a compound sort key. Which queries will MOST increase the speed of a query by using a compound sort key of the table? (Select TWO.)

A. Select * from Employee where Region ID='North America';

B. Select * from Employee where Region ID='North America' and Department ID=20;

C. Select * from Employee where Department ID=20 and Region ID='North America';

D. Select '' from Employee where Role ID=50;

E. Select * from Employee where Region ID='North America' and Role ID=50;

Answer: BC

Explanation:

In Amazon Redshift, a compound sort key is designed to optimize the performance of queries that use filtering and join conditions on the columns in the sort key. A compound sort key orders the data based on the first column, followed by the second, and so on. In the scenario given, the compound sort key consists of Region ID, Department ID, and Role ID. Therefore, queries that filter on the leading columns of the sort key are more likely to benefit from this order.

Option B: "Select * from Employee where Region ID='North America' and Department ID=20;" This query will perform well because it uses both the Region ID and Department ID, which are the first two columns of the compound sort key. The order of the columns in the WHERE clause matches the order in the sort key, thus allowing the query to scan fewer rows and improve performance. Option C: "Select * from Employee where Department ID=20 and Region ID='North America';" This query also benefits from the compound sort key because it includes both Region ID and Department ID, which are the first two columns in the sort key. Although the order in the WHERE clause does not match exactly, Amazon Redshift will still leverage the sort key to reduce the amount of data scanned, improving query speed.

Options A, D, and E are less optimal because they do not utilize the sort key as effectively: Option A only filters by the Region ID, which may still use the sort key but does not take full advantage of the compound nature.

Option D uses only Role ID, the last column in the compound sort key, which will not benefit much from sorting since it is the third key in the sort order.

Option E filters on Region ID and Role ID but skips the Department ID column, making it less efficient for the compound sort key.

Reference:

Amazon Redshift Documentation - Sorting Data

AWS Certified Data Analytics Study Guide

AWS Certification - Data Engineer Associate Exam Guide

Question: 104

A car sales company maintains data about cars that are listed for sale in an are
a. The company receives data about new car listings from vendors who upload the data daily as compressed files into Amazon S3. The compressed files are up to 5 KB in size. The company wants to see the most up-to-date listings as soon as the data is uploaded to Amazon S3.

A data engineer must automate and orchestrate the data processing workflow of the listings to feed a dashboard. The data engineer must also provide the ability to perform one-time queries and analytical reporting. The query solution must be scalable.

Which solution will meet these requirements MOST cost-effectively?

A. Use an Amazon EMR cluster to process incoming data. Use AWS Step Functions to orchestrate workflows. Use Apache Hive for one-time queries and analytical reporting. Use Amazon OpenSearch Service to bulk ingest the data into compute optimized instances. Use OpenSearch Dashboards in OpenSearch Service for the dashboard.

B. Use a provisioned Amazon EMR cluster to process incoming data. Use AWS Step Functions to orchestrate workflows. Use Amazon Athena for one-time queries and analytical reporting. Use Amazon QuickSight for the dashboard.

C. Use AWS Glue to process incoming data. Use AWS Step Functions to orchestrate workflows. Use Amazon Redshift Spectrum for one-time queries and analytical reporting. Use OpenSearch Dashboards in Amazon

OpenSearch Service for the dashboard.

D. Use AWS Glue to process incoming data. Use AWS Lambda and S3 Event Notifications to orchestrate workflows. Use Amazon Athena for one-time queries and analytical reporting. Use Amazon QuickSight for the dashboard.

Answer: D

Explanation:

For processing the incoming car listings in a cost-effective, scalable, and automated way, the ideal approach involves using AWS Glue for data processing, AWS Lambda with S3 Event Notifications for orchestration, Amazon Athena for one-time queries and analytical reporting, and Amazon QuickSight for visualization on the dashboard. Let's break this down:

AWS Glue: This is a fully managed ETL (Extract, Transform, Load) service that automatically processes the incoming data files. Glue is serverless and supports diverse data sources, including Amazon S3 and Redshift.

AWS Lambda and S3 Event Notifications: Using Lambda and S3 Event Notifications allows near realtime triggering of processing workflows as soon as new data is uploaded into S3. This approach is event-driven, ensuring that the listings are processed as soon as they are uploaded, reducing the latency for data processing.

Amazon Athena: A serverless, pay-per-query service that allows interactive queries directly against data in S3 using standard SQL. It is ideal for the requirement of one-time queries and analytical reporting without the need for provisioning or managing servers.

Amazon QuickSight: A business intelligence tool that integrates with a wide range of AWS data sources, including Athena, and is used for creating interactive dashboards. It scales well and provides real-time insights for the car listings.

This solution (Option D) is the most cost-effective, because both Glue and Athena are serverless and priced based on usage, reducing costs when compared to provisioning EMR clusters in the other options. Moreover, using Lambda for orchestration is more cost-effective than AWS Step Functions due to its lightweight nature.

Reference:

AWS Glue Documentation

Amazon Athena Documentation

Amazon QuickSight Documentation

S3 Event Notifications and Lambda

Question: 105

A data engineer needs to debug an AWS Glue job that reads from Amazon S3 and writes to Amazon Redshift. The data engineer enabled the bookmark feature for the AWS Glue job. The data engineer has set the maximum concurrency for the AWS Glue job to 1.

The AWS Glue job is successfully writing the output to Amazon Redshift. However, the Amazon S3 files that were loaded during previous runs of the AWS Glue job are being reprocessed by subsequent runs.

What is the likely reason the AWS Glue job is reprocessing the files?

A. The AWS Glue job does not have the s3:GetObjectAcl permission that is required for bookmarks to work correctly.

B. The maximum concurrency for the AWS Glue job is set to 1.

C. The data engineer incorrectly specified an older version of AWS Glue for the Glue job.

D. The AWS Glue job does not have a required commit statement.

Answer: A

Explanation:

The issue described is that the AWS Glue job is reprocessing files from previous runs despite the bookmark feature being enabled. Bookmarks in AWS Glue allow jobs to keep track of which files or data have already been processed to avoid reprocessing. The most likely reason for reprocessing the files is missing S3 permissions, specifically s3

s3

is a permission required by AWS Glue when bookmarks are enabled to ensure Glue can retrieve metadata from the files in S3, which is necessary for the bookmark mechanism to function correctly. Without this permission, Glue cannot track which files have been processed, resulting in reprocessing during subsequent runs.

Concurrency settings (Option B) and the version of AWS Glue (Option C) do not affect the bookmark behavior. Similarly, the lack of a commit statement (Option D) is not applicable in this context, as Glue handles commits internally when interacting with Redshift and S3.

Thus, the root cause is likely related to insufficient permissions on the S3 bucket, specifically s3

, which is required for bookmarks to work as expected.

Reference:

AWS Glue Job Bookmarks Documentation

AWS Glue Permissions for Bookmarks

Question: 106

A company is building a data lake for a new analytics team. The company is using Amazon S3 for storage and Amazon Athena for query analysis. All data that is in Amazon S3 is in Apache Parquet format.

The company is running a new Oracle database as a source system in the company's data center. The company has 70 tables in the Oracle database. All the tables have primary keys. Data can occasionally change in the source system. The company wants to ingest the tables every day into the data lake. Which solution will meet this requirement with the LEAST effort?

A. Create an Apache Sqoop job in Amazon EMR to read the data from the Oracle database. Configure the Sqoop job to write the data to Amazon S3 in Parquet format.

B. Create an AWS Glue connection to the Oracle database. Create an AWS Glue bookmark job to ingest the data incrementally and to write the data to Amazon S3 in Parquet format.

C. Create an AWS Database Migration Service (AWS DMS) task for ongoing replication. Set the Oracle database as the source. Set Amazon S3 as the target. Configure the task to write the data in Parquet format.

D. Create an Oracle database in Amazon RDS. Use AWS Database Migration Service (AWS DMS) to migrate the on-premises Oracle database to Amazon RDS. Configure triggers on the tables to invoke AWS Lambda functions to write changed records to Amazon S3 in Parquet format.

Answer: C

Explanation:

The company needs to ingest tables from an on-premises Oracle database into a data lake on Amazon S3 in Apache Parquet format. The most efficient solution, requiring the least manual effort, would be to use AWS Database Migration Service (DMS) for continuous data replication.

Option C: Create an AWS Database Migration Service (AWS DMS) task for ongoing replication. Set the Oracle

database as the source. Set Amazon S3 as the target. Configure the task to write the data in Parquet format. AWS DMS can continuously replicate data from the Oracle database into Amazon S3, transforming it into Parquet format as it ingests the data. DMS simplifies the process by providing ongoing replication with minimal setup, and it automatically handles the conversion to Parquet format without requiring manual transformations or separate jobs. This option is the least effort solution since it automates both the ingestion and transformation processes.

Other options:

Option A (Apache Sqoop on EMR) involves more manual configuration and management, including setting up EMR clusters and writing Sqoop jobs.

Option B (AWS Glue bookmark job) involves configuring Glue jobs, which adds complexity. While Glue supports data transformations, DMS offers a more seamless solution for database replication. Option D (RDS and Lambda triggers) introduces unnecessary complexity by involving RDS and Lambda for a task that DMS can handle more efficiently.

Reference:

AWS Database Migration Service (DMS)

DMS S3 Target Documentation

Question: 107

A retail company uses Amazon Aurora PostgreSQL to process and store live transactional dat
a. The company uses an Amazon Redshift cluster for a data warehouse.
An extract, transform, and load (ETL) job runs every morning to update the Redshift cluster with new data from the PostgreSQL database. The company has grown rapidly and needs to cost optimize the Redshift cluster.
A data engineer needs to create a solution to archive historical data. The data engineer must be able to run analytics queries that effectively combine data from live transactional data in PostgreSQL, current data in Redshift, and archived historical data. The solution must keep only the most recent 15 months of data in Amazon Redshift to reduce costs.

Which combination of steps will meet these requirements? (Select TWO.)

A. Configure the Amazon Redshift Federated Query feature to query live transactional data that is in the PostgreSQL database.

B. Configure Amazon Redshift Spectrum to query live transactional data that is in the PostgreSQL database.

C. Schedule a monthly job to copy data that is older than 15 months to Amazon S3 by using the UNLOAD command. Delete the old data from the Redshift cluster. Configure Amazon Redshift Spectrum to access historical data in Amazon S3.

D. Schedule a monthly job to copy data that is older than 15 months to Amazon S3 Glacier Flexible Retrieval by using the UNLOAD command. Delete the old data from the Redshift duster. Configure Redshift Spectrum to access historical data from S3 Glacier Flexible Retrieval.

E. Create a materialized view in Amazon Redshift that combines live, current, and historical data from different sources.

Answer: A, C

Explanation:

The goal is to archive historical data from an Amazon Redshift data warehouse while combining live transactional data from Amazon Aurora PostgreSQL with current and historical data in a cost-efficient manner. The company wants to keep only the last 15 months of data in Redshift to reduce costs. Option A: "Configure the Amazon Redshift Federated Query feature to query live transactional data that is in the PostgreSQL database."

Redshift Federated Query allows querying live transactional data directly from Aurora PostgreSQL without having to move it into Redshift, thereby enabling seamless integration of the current data in Redshift and live data in PostgreSQL. This is a cost-effective approach, as it avoids unnecessary data duplication.

Option C: "Schedule a monthly job to copy data that is older than 15 months to Amazon S3 by using the UNLOAD command. Delete the old data from the Redshift cluster. Configure Amazon Redshift Spectrum to access historical data in Amazon S3."

This option uses Amazon Redshift Spectrum, which enables Redshift to query data directly in S3 without moving it into Redshift. By unloading older data (older than 15 months) to S3, and then using Spectrum to access it, this approach reduces storage costs significantly while still allowing the data to be queried when necessary.

Option B (Redshift Spectrum for live PostgreSQL data) is not applicable, as Redshift Spectrum is intended for querying data in Amazon S3, not live transactional data in Aurora.

Option D (S3 Glacier Flexible Retrieval) is not suitable because Glacier is designed for long-term archival storage with infrequent access, and querying data in Glacier for analytics purposes would incur higher retrieval times and costs.

Option E (materialized views) would not meet the need to archive data or combine it from multiple sources; it is best suited for combining frequently accessed data already in Redshift.

Reference:

Amazon Redshift Federated Query

Amazon Redshift Spectrum Documentation

Amazon Redshift UNLOAD Command

Question: 108

A data engineer needs to create an Amazon Athena table based on a subset of data from an existing Athena table named cities_world. The cities_world table contains cities that are located around the world. The data engineer must create a new table named cities_us to contain only the cities from cities_world that are located in the US.

Which SQL statement should the data engineer use to meet this requirement?

A.

```
INSERT INTO cities usa (city,state) SELECT city, state FROM cities world WHERE country''usa';
```

B.

```
MOVE city, state FROM cities_world TO cities_usa WHERE country-'usa';
```

C.

```
INSERT INTO citiesjjsa SELECT city, state FROM citiesworld WHERE country^'usa';
```

D.

```
UPDATE cities_usa SET (city, state) = (SELECT city, state FROM cities_world WHERE
```

A. Option A

B. Option B

C. Option C

D. Option D

Answer: A

Explanation:

To create a new table named cities_usa in Amazon Athena based on a subset of data from the existing cities_world table, you should use an INSERT INTO statement combined with a SELECT statement to filter only the records where the country is 'usa'. The correct SQL syntax would be: Option A: INSERT INTO cities_usa (city, state) SELECT city, state FROM cities_world WHERE country='usa';

This statement inserts only the cities and states where the country column has a value of 'usa' from the cities_world table into the cities_usa table. This is a correct approach to create a new table with data filtered from an existing table in Athena.

Options B, C, and D are incorrect due to syntax errors or incorrect SQL usage (e.g., the MOVE command or the use of UPDATE in a non-relevant context).

Reference:

Amazon Athena SQL Reference

Creating Tables in Athena

Question: 109

A company uses Amazon S3 as a data lake. The company sets up a data warehouse by using a multinode Amazon Redshift cluster. The company organizes the data files in the data lake based on the data source of each data file.

The company loads all the data files into one table in the Redshift cluster by using a separate COPY command for each data file location. This approach takes a long time to load all the data files into the table. The company must increase the speed of the data ingestion. The company does not want to increase the cost of the process.

Which solution will meet these requirements?

A. Use a provisioned Amazon EMR cluster to copy all the data files into one folder. Use a COPY command to load the data into Amazon Redshift.

B. Load all the data files in parallel into Amazon Aurora. Run an AWS Glue job to load the data into Amazon Redshift.

C. Use an AWS Glue job to copy all the data files into one folder. Use a COPY command to load the data into Amazon Redshift.

D. Create a manifest file that contains the data file locations. Use a COPY command to load the data into Amazon Redshift.

Answer: D

Explanation:

The company is facing performance issues loading data into Amazon Redshift because it is issuing separate COPY commands for each data file location. The most efficient way to increase the speed of data ingestion into Redshift without increasing the cost is to use a manifest file.

Option D: Create a manifest file that contains the data file locations. Use a COPY command to load the data into Amazon Redshift.

A manifest file provides a list of all the data files, allowing the COPY command to load all files in parallel from different locations in Amazon S3. This significantly improves the loading speed without adding costs, as it optimizes the data loading process in a single COPY operation.

Other options (A, B, C) involve additional steps that would either increase the cost (provisioning clusters, using Glue, etc.) or do not address the core issue of needing a unified and efficient COPY process.

Reference:
Amazon Redshift COPY Command
Redshift Manifest File Documentation

Question: 110

A company stores customer records in Amazon S3. The company must not delete or modify the customer record data for 7 years after each record is created. The root user also must not have the ability to delete or modify the data.
A data engineer wants to use S3 Object Lock to secure the data.
Which solution will meet these requirements?

A. Enable governance mode on the S3 bucket. Use a default retention period of 7 years.
B. Enable compliance mode on the S3 bucket. Use a default retention period of 7 years.
C. Place a legal hold on individual objects in the S3 bucket. Set the retention period to 7 years.
D. Set the retention period for individual objects in the S3 bucket to 7 years.

Answer: B

Explanation:
The company wants to ensure that no customer records are deleted or modified for 7 years, and even the root user should not have the ability to change the data. S3 Object Lock in Compliance Mode is the correct solution for this scenario.
Option B: Enable compliance mode on the S3 bucket. Use a default retention period of 7 years. In Compliance Mode, even the root user cannot delete or modify locked objects during the retention period. This ensures that the data is protected for the entire 7-year duration as required. Compliance mode is stricter than governance mode and prevents all forms of alteration, even by privileged users. Option A (Governance Mode) still allows certain privileged users (like the root user) to bypass the lock, which does not meet the company's requirement. Option C (legal hold) and Option D (setting retention per object) do not fully address the requirement to block root user modifications. Reference:
Amazon S3 Object Lock Documentation

Question: 111

A data engineer wants to orchestrate a set of extract, transform, and load (ETL) jobs that run on AWS. The ETL jobs contain tasks that must run Apache Spark jobs on Amazon EMR, make API calls to Salesforce, and load data into Amazon Redshift.
The ETL jobs need to handle failures and retries automatically. The data engineer needs to use Python to orchestrate the jobs.
Which service will meet these requirements?

A. Amazon Managed Workflows for Apache Airflow (Amazon MWAA)
B. AWS Step Functions
C. AWS Glue
D. Amazon EventBridge

Answer: A

Explanation:

The data engineer needs to orchestrate ETL jobs that include Spark jobs on Amazon EMR, API calls to Salesforce, and loading data into Redshift. They also need automatic failure handling and retries. Amazon Managed Workflows for Apache Airflow (Amazon MWAA) is the best solution for this requirement.

Option A: Amazon Managed Workflows for Apache Airflow (Amazon MWAA)

Apache Airflow is designed for complex job orchestration, allowing users to define workflows (DAGs) in Python. MWAA manages Airflow and its integrations with other AWS services, including Amazon EMR, Redshift, and external APIs like Salesforce. It provides automatic retry handling, failure detection, and detailed monitoring, which fits the use case perfectly.

Option B (AWS Step Functions) can orchestrate tasks but doesn't natively support complex workflow definitions with Python like Airflow does.

Option C (AWS Glue) is more focused on ETL and doesn't handle the orchestration of external systems like Salesforce as well as Airflow.

Option D (Amazon EventBridge) is more suited for event-driven architectures rather than complex workflow orchestration.

Reference:

Amazon Managed Workflows for Apache Airflow

Apache Airflow on AWS

Question: 112

A data engineer needs to build an enterprise data catalog based on the company's Amazon S3 buckets and Amazon RDS databases. The data catalog must include storage format metadata for the data in the catalog. Which solution will meet these requirements with the LEAST effort?

A. Use an AWS Glue crawler to scan the S3 buckets and RDS databases and build a data catalog. Use data stewards to inspect the data and update the data catalog with the data format.

B. Use an AWS Glue crawler to build a data catalog. Use AWS Glue crawler classifiers to recognize the format of data and store the format in the catalog.

C. Use Amazon Macie to build a data catalog and to identify sensitive data elements. Collect the data format information from Macie.

D. Use scripts to scan data elements and to assign data classifications based on the format of the data.

Answer: B

Explanation:

To build an enterprise data catalog with metadata for storage formats, the easiest and most efficient solution is using an AWS Glue crawler. The Glue crawler can scan Amazon S3 buckets and Amazon RDS databases to automatically create a data catalog that includes metadata such as the schema and storage format (e.g., CSV, Parquet, etc.). By using AWS Glue crawler classifiers, you can configure the crawler to recognize the format of the data and store this information directly in the catalog. Option B: Use an AWS Glue crawler to build a data catalog. Use AWS Glue crawler classifiers to recognize the format of data and store the format in the catalog. This option meets the requirements with the least effort because Glue crawlers automate the discovery and cataloging of data from multiple sources, including S3 and RDS, while recognizing various file formats via

classifiers.

Other options (A, C, D) involve additional manual steps, like having data stewards inspect the data, or using services like Amazon Macie that focus more on sensitive data detection rather than format cataloging.

Reference:

AWS Glue Crawler Documentation

AWS Glue Classifiers

Question: 113

A company needs a solution to manage costs for an existing Amazon DynamoDB table. The company also needs to control the size of the table. The solution must not disrupt any ongoing read or write operations. The company wants to use a solution that automatically deletes data from the table after 1 month.

Which solution will meet these requirements with the LEAST ongoing maintenance?

A. Use the DynamoDB TTL feature to automatically expire data based on timestamps.

B. Configure a scheduled Amazon EventBridge rule to invoke an AWS Lambda function to check for data that is older than 1 month. Configure the Lambda function to delete old data.

C. Configure a stream on the DynamoDB table to invoke an AWS Lambda function. Configure the Lambda function to delete data in the table that is older than 1 month.

D. Use an AWS Lambda function to periodically scan the DynamoDB table for data that is older than 1 month. Configure the Lambda function to delete old data.

Answer: A

Explanation:

The requirement is to manage the size of an Amazon DynamoDB table by automatically deleting data older than 1 month without disrupting ongoing read or write operations. The simplest and most maintenance-free solution is to use DynamoDB Time-to-Live (TTL).

Option A: Use the DynamoDB TTL feature to automatically expire data based on timestamps. DynamoDB TTL allows you to specify an attribute (e.g., a timestamp) that defines when items in the table should expire. After the expiration time, DynamoDB automatically deletes the items, freeing up storage space and keeping the table size under control without manual intervention or disruptions to ongoing operations.

Other options involve higher maintenance and manual scheduling or scanning operations, which increase complexity unnecessarily compared to the native TTL feature.

Reference:

DynamoDB Time-to-Live (TTL)

Question: 114

A company stores CSV files in an Amazon S3 bucket. A data engineer needs to process the data in the CSV files and store the processed data in a new S3 bucket.

The process needs to rename a column, remove specific columns, ignore the second row of each file, create a new column based on the values of the first row of the data, and filter the results by a numeric value of a column.

Which solution will meet these requirements with the LEAST development effort?

A. Use AWS Glue Python jobs to read and transform the CSV files.
B. Use an AWS Glue custom crawler to read and transform the CSV files.
C. Use an AWS Glue workflow to build a set of jobs to crawl and transform the CSV files.
D. Use AWS Glue DataBrew recipes to read and transform the CSV files.

Answer: D

Explanation:
The requirement involves transforming CSV files by renaming columns, removing rows, and other operations with minimal development effort. AWS Glue DataBrew is the best solution here because it allows you to visually create transformation recipes without writing extensive code.
Option D: Use AWS Glue DataBrew recipes to read and transform the CSV files.
DataBrew provides a visual interface where you can build transformation steps (e.g., renaming columns, filtering rows, creating new columns, etc.) as a "recipe" that can be applied to datasets, making it easy to handle complex transformations on CSV files with minimal coding.
Other options (A, B, C) involve more manual development and configuration effort (e.g., writing Python jobs or creating custom workflows in Glue) compared to the low-code/no-code approach of DataBrew.
Reference:
AWS Glue DataBrew Documentation

Question: 115

A company uses Amazon Redshift as its data warehouse. Data encoding is applied to the existing tables of the data warehouse. A data engineer discovers that the compression encoding applied to some of the tables is not the best fit for the data.
The data engineer needs to improve the data encoding for the tables that have sub-optimal encoding.
Which solution will meet this requirement?

A. Run the ANALYZE command against the identified tables. Manually update the compression encoding of columns based on the output of the command.
B. Run the ANALYZE COMPRESSION command against the identified tables. Manually update the compression encoding of columns based on the output of the command.
C. Run the VACUUM REINDEX command against the identified tables.
D. Run the VACUUM RECLUSTER command against the identified tables.

Answer: B

Explanation:
To improve data encoding for Amazon Redshift tables where sub-optimal encoding has been applied, the correct approach is to analyze the table to determine the optimal encoding based on the data distribution and characteristics.
Option B: Run the ANALYZE COMPRESSION command against the identified tables. Manually update the compression encoding of columns based on the output of the command.
The ANALYZE COMPRESSION command in Amazon Redshift analyzes the columnar data and suggests the best compression encoding for each column. The output provides recommendations for changing the current encoding to improve storage efficiency and query performance. After analyzing, you can manually apply the recommended encoding to the columns.

Option A (ANALYZE command) is incorrect because it is primarily used to update statistics on tables, not to analyze or suggest compression encoding.

Options C and D (VACUUM commands) deal with reclaiming disk space and reorganizing data, not optimizing compression encoding.

Reference:

Amazon Redshift ANALYZE COMPRESSION Command

Question: 116

A retail company is using an Amazon Redshift cluster to support real-time inventory management. The company has deployed an ML model on a real-time endpoint in Amazon SageMaker.

The company wants to make real-time inventory recommendations. The company also wants to make predictions about future inventory needs.

Which solutions will meet these requirements? (Select TWO.)

A. Use Amazon Redshift ML to generate inventory recommendations.
B. Use SQL to invoke a remote SageMaker endpoint for prediction.
C. Use Amazon Redshift ML to schedule regular data exports for ofline model training.
D. Use SageMaker Autopilot to create inventory management dashboards in Amazon Redshift.
E. Use Amazon Redshift as a file storage system to archive old inventory management reports.

Answer: AB

Explanation:

The company needs to use machine learning models for real-time inventory recommendations and future inventory predictions while leveraging both Amazon Redshift and Amazon SageMaker. Option A: Use Amazon Redshift ML to generate inventory recommendations.

Amazon Redshift ML allows you to build, train, and deploy machine learning models directly from Redshift using SQL statements. It integrates with SageMaker to train models and run inference. This feature is useful for generating inventory recommendations directly from the data stored in Redshift. Option B: Use SQL to invoke a remote SageMaker endpoint for prediction.

You can use SQL in Redshift to call a SageMaker endpoint for real-time inference. By invoking a SageMaker endpoint from within Redshift, the company can get real-time predictions on inventory, allowing for integration between the data warehouse and the machine learning model hosted in SageMaker.

Option C (ofline model training) and Option D (creating dashboards with SageMaker Autopilot) are not relevant to the real-time prediction and recommendation requirements.

Option E (archiving inventory reports in Redshift) is not related to making predictions or recommendations.

Reference:

Amazon Redshift ML Documentation

Invoking SageMaker Endpoints from SQL

Question: 117

A company implements a data mesh that has a central governance account. The company needs to catalog all data in the governance account. The governance account uses AWS Lake Formation to centrally share data and grant access permissions.

The company has created a new data product that includes a group of Amazon Redshift Serverless tables. A data

engineer needs to share the data product with a marketing team. The marketing team must have access to only a subset of columns. The data engineer needs to share the same data product with a compliance team. The compliance team must have access to a different subset of columns than the marketing team needs access to.

Which combination of steps should the data engineer take to meet these requirements? (Select TWO.)

A. Create views of the tables that need to be shared. Include only the required columns.

B. Create an Amazon Redshift data than that includes the tables that need to be shared.

C. Create an Amazon Redshift managed VPC endpoint in the marketing team's account. Grant the marketing team access to the views.

D. Share the Amazon Redshift data share to the Lake Formation catalog in the governance account.

E. Share the Amazon Redshift data share to the Amazon Redshift Serverless workgroup in the marketing team's account.

Answer: A, E

Explanation:

The company is using a data mesh architecture with AWS Lake Formation for governance and needs to share specific subsets of data with different teams (marketing and compliance) using Amazon Redshift Serverless.

Option A: Create views of the tables that need to be shared. Include only the required columns. Creating views in Amazon Redshift that include only the necessary columns allows for fine-grained access control. This method ensures that each team has access to only the data they are authorized to view.

Option E: Share the Amazon Redshift data share to the Amazon Redshift Serverless workgroup in the marketing team's account.

Amazon Redshift data sharing enables live access to data across Redshift clusters or Serverless workgroups. By sharing data with specific workgroups, you can ensure that the marketing team and compliance team each access the relevant subset of data based on the views created.

Option B (creating a Redshift data share) is close but does not address the fine-grained column-level access.

Option C (creating a managed VPC endpoint) is unnecessary for sharing data with specific teams.

Option D (sharing with the Lake Formation catalog) is incorrect because Redshift data shares do not integrate directly with Lake Formation catalogs; they are specific to Redshift workgroups.

Reference:

Amazon Redshift Data Sharing

AWS Lake Formation Documentation

Question: 118

A data engineer is launching an Amazon EMR duster. The data that the data engineer needs to load into the new cluster is currently in an Amazon S3 bucket. The data engineer needs to ensure that data is encrypted both at rest and in transit.

The data that is in the S3 bucket is encrypted by an AWS Key Management Service (AWS KMS) key. The data engineer has an Amazon S3 path that has a Privacy Enhanced Mail (PEM) file.

Which solution will meet these requirements?

A. Create an Amazon EMR security configuration. Specify the appropriate AWS KMS key for at-rest encryption for the S3 bucket. Create a second security configuration. Specify the Amazon S3 path of the PEM file for in-transit encryption. Create the EMR cluster, and attach both security configurations to the cluster.

B. Create an Amazon EMR security configuration. Specify the appropriate AWS KMS key for local disk encryption

for the S3 bucket. Specify the Amazon S3 path of the PEM file for in-transit encryption. Use the security configuration during EMR cluster creation.

C. Create an Amazon EMR security configuration. Specify the appropriate AWS KMS key for at-rest encryption for the S3 bucket. Specify the Amazon S3 path of the PEM file for in-transit encryption. Use the security configuration during EMR cluster creation.

D. Create an Amazon EMR security configuration. Specify the appropriate AWS KMS key for at-rest encryption for the S3 bucket. Specify the Amazon S3 path of the PEM file for in-transit encryption. Create the EMR cluster, and attach the security configuration to the cluster.

Answer: C

Explanation:

The data engineer needs to ensure that the data in an Amazon EMR cluster is encrypted both at rest and in transit. The data in Amazon S3 is already encrypted using an AWS KMS key. To meet the requirements, the most suitable solution is to create an EMR security configuration that specifies the correct KMS key for at-rest encryption and use the PEM file for in-transit encryption.

Option C: Create an Amazon EMR security configuration. Specify the appropriate AWS KMS key for at-rest encryption for the S3 bucket. Specify the Amazon S3 path of the PEM file for in-transit encryption. Use the security configuration during EMR cluster creation.

This option configures encryption for both data at rest (using KMS keys) and data in transit (using the PEM file for SSL/TLS encryption). This approach ensures that data is fully protected during storage and transfer.

Options A, B, and D either involve creating unnecessary additional security configurations or make inaccurate assumptions about the way encryption configurations are attached.

Reference:

Amazon EMR Security Configuration

Amazon S3 Encryption

Question: 119

A company analyzes data in a data lake every quarter to perform inventory assessments. A data engineer uses AWS Glue DataBrew to detect any personally identifiable information (PII) about customers within the dat a. The company's privacy policy considers some custom categories of information to be PII. However, the categories are not included in standard DataBrew data quality rules.

The data engineer needs to modify the current process to scan for the custom PII categories across multiple datasets within the data lake.

Which solution will meet these requirements with the LEAST operational overhead?

A. Manually review the data for custom PII categories.

B. Implement custom data quality rules in Data Brew. Apply the custom rules across datasets.

C. Develop custom Python scripts to detect the custom PII categories. Call the scripts from DataBrew.

D. Implement regex patterns to extract PII information from fields during extract transform, and load (ETL) operations into the data lake.

Answer: B

Explanation:

The data engineer needs to detect custom categories of PII within the data lake using AWS Glue DataBrew. While DataBrew provides standard data quality rules, the solution must support custom PII categories.

Option B: Implement custom data quality rules in DataBrew. Apply the custom rules across datasets. This option is the most efficient because DataBrew allows the creation of custom data quality rules that can be applied to detect specific data patterns, including custom PII categories. This approach minimizes operational overhead while ensuring that the specific privacy requirements are met. Options A, C, and D either involve manual intervention or developing custom scripts, both of which increase operational effort compared to using DataBrew's built-in capabilities.

Reference:

AWS Glue DataBrew Documentation

Question: 120

A marketing company uses Amazon S3 to store marketing dat

a. The company uses versioning in some buckets. The company runs several jobs to read and load data into the buckets.

To help cost-optimize its storage, the company wants to gather information about incomplete multipart uploads and outdated versions that are present in the S3 buckets.

Which solution will meet these requirements with the LEAST operational effort?

A. Use AWS CLI to gather the information.

B. Use Amazon S3 Inventory configurations reports to gather the information.

C. Use the Amazon S3 Storage Lens dashboard to gather the information.

D. Use AWS usage reports for Amazon S3 to gather the information.

Answer: B

Explanation:

The company wants to gather information about incomplete multipart uploads and outdated versions in its Amazon S3 buckets to optimize storage costs.

Option B: Use Amazon S3 Inventory configurations reports to gather the information.

S3 Inventory provides reports that can list incomplete multipart uploads and versions of objects stored in S3. It offers an easy, automated way to track object metadata across buckets, including data necessary for cost optimization, without manual effort.

Options A (AWS CLI), C (S3 Storage Lens), and D (usage reports) either do not specifically gather the required information about incomplete uploads and outdated versions or require more manual intervention.

Reference:

Amazon S3 Inventory Documentation

Question: 121

A telecommunications company collects network usage data throughout each day at a rate of several thousand data points each second. The company runs an application to process the usage data in real time. The company aggregates and stores the data in an Amazon Aurora DB instance.

Sudden drops in network usage usually indicate a network outage. The company must be able to identify sudden drops in network usage so the company can take immediate remedial actions. Which solution will meet this requirement with the LEAST latency?

A. Create an AWS Lambda function to query Aurora for drops in network usage. Use Amazon EventBridge to automatically invoke the Lambda function every minute.

B. Modify the processing application to publish the data to an Amazon Kinesis data stream. Create an Amazon

Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) application to detect drops in network usage.

C. Replace the Aurora database with an Amazon DynamoDB table. Create an AWS Lambda function to query the DynamoDB table for drops in network usage every minute. Use DynamoDB Accelerator (DAX) between the processing application and DynamoDB table.

D. Create an AWS Lambda function within the Database Activity Streams feature of Aurora to detect drops in network usage.

Answer: B

Explanation:

The telecommunications company needs a low-latency solution to detect sudden drops in network usage from real-time data collected throughout the day.

Option B: Modify the processing application to publish the data to an Amazon Kinesis data stream. Create an Amazon Managed Service for Apache Flink (Amazon Kinesis Data Analytics) application to detect drops in network usage.

Using Amazon Kinesis with Managed Service for Apache Flink (formerly Kinesis Data Analytics) is ideal for real-time stream processing with minimal latency. Flink can analyze the incoming data stream in real-time and detect anomalies, such as sudden drops in usage, which makes it the best fit for this scenario.

Other options (A, C, and D) either introduce unnecessary delays (e.g., querying databases) or do not provide the same real-time, low-latency processing that is critical for this use case.

Reference:

Amazon Kinesis Data Analytics for Apache Flink

Amazon Kinesis Documentation

Question: 122

A company maintains a data warehouse in an on-premises Oracle database. The company wants to build a data lake on AWS. The company wants to load data warehouse tables into Amazon S3 and synchronize the tables with incremental data that arrives from the data warehouse every day. Each table has a column that contains monotonically increasing values. The size of each table is less than 50 GB. The data warehouse tables are refreshed every night between 1 AM and 2 AM. A business intelligence team queries the tables between 10 AM and 8 PM every day.

Which solution will meet these requirements in the MOST operationally efficient way?

A. Use an AWS Database Migration Service (AWS DMS) full load plus CDC job to load tables that contain monotonically increasing data columns from the on-premises data warehouse to Amazon S3. Use custom logic in AWS Glue to append the daily incremental data to a full-load copy that is in Amazon S3.

B. Use an AWS Glue Java Database Connectivity (JDBC) connection. Configure a job bookmark for a column that contains monotonically increasing values. Write custom logic to append the daily incremental data to a full-load copy that is in Amazon S3.

C. Use an AWS Database Migration Service (AWS DMS) full load migration to load the data warehouse tables into Amazon S3 every day Overwrite the previous day's full-load copy every day.

D. Use AWS Glue to load a full copy of the data warehouse tables into Amazon S3 every day. Overwrite the previous day's full-load copy every day.

Answer: A

Explanation:

The company needs to load data warehouse tables into Amazon S3 and perform incremental synchronization with daily updates. The most efficient solution is to use AWS Database Migration Service (AWS DMS) with a combination of full load and change data capture (CDC) to handle the initial load and daily incremental updates.

Option A: Use an AWS Database Migration Service (AWS DMS) full load plus CDC job to load tables that contain monotonically increasing data columns from the on-premises data warehouse to Amazon S3. Use custom logic in AWS Glue to append the daily incremental data to a full-load copy that is in Amazon S3.

DMS is designed to migrate databases to AWS, and the combination of full load plus CDC is ideal for handling incremental data changes efficiently. AWS Glue can then be used to append the incremental data to the full data set in S3. This solution is highly operationally efficient because it automates both the full load and incremental updates.

Options B, C, and D are less operationally efficient because they either require writing custom logic to handle bookmarks manually or involve unnecessary daily full loads.

Reference:

AWS Database Migration Service Documentation

AWS Glue Documentation

Question: 123

A company is using Amazon Redshift to build a data warehouse solution. The company is loading hundreds of tiles into a tact table that is in a Redshift cluster.

The company wants the data warehouse solution to achieve the greatest possible throughput. The solution must use cluster resources optimally when the company loads data into the tact table. Which solution will meet these requirements?

A. Use multiple COPY commands to load the data into the Redshift cluster.

B. Use S3DistCp to load multiple files into Hadoop Distributed File System (HDFS). Use an HDFS connector to ingest the data into the Redshift cluster.

C. Use a number of INSERT statements equal to the number of Redshift cluster nodes. Load the data in parallel into each node.

D. Use a single COPY command to load the data into the Redshift cluster.

Answer: D

Explanation:

To achieve the highest throughput and efficiently use cluster resources while loading data into an Amazon Redshift cluster, the optimal approach is to use a single COPY command that ingests data in parallel.

Option D: Use a single COPY command to load the data into the Redshift cluster.

The COPY command is designed to load data from multiple files in parallel into a Redshift table, using all the cluster nodes to optimize the load process. Redshift is optimized for parallel processing, and a single COPY command can load multiple files at once, maximizing throughput.

Options A, B, and C either involve unnecessary complexity or inefficient approaches, such as using multiple COPY commands or INSERT statements, which are not optimized for bulk loading. Reference:

Amazon Redshift COPY Command Documentation

Question: 124

A company has a data warehouse that contains a table that is named Sales. The company stores the table in Amazon Redshift The table includes a column that is named city_name. The company wants to query the table to find all rows that have a city_name that starts with "San" or "El." Which SQL query will meet this requirement?

A. Select * from Sales where city_name - '$(San|El)";
B. Select * from Sales where city_name -, ^(San | El) *';
C. Select * from Sales where city_name - '$(San&El)";
D. Select * from Sales where city_name -, A(San&El)";

Answer: B

Explanation:
To query the Sales table in Amazon Redshift for city names that start with "San" or "El," the appropriate query uses a regular expression (regex) pattern to match city names that begin with those prefixes.
Option B: Select * from Sales where city_name ~ 'A(San|El)';
In Amazon Redshift, the ~ operator is used to perform pattern matching using regular expressions. The A(San|El) pattern matches city names that start with "San" or "El." This is the correct SQL syntax for this use case.
Other options (A, C, D) contain incorrect syntax or incorrect use of special characters, making them invalid queries.
Reference:
Amazon Redshift Pattern Matching Documentation

Question: 125

A company plans to use Amazon Kinesis Data Firehose to store data in Amazon S3. The source data consists of 2 MB csv files. The company must convert the .csv files to JSON format. The company must store the files in Apache Parquet format.
Which solution will meet these requirements with the LEAST development effort?

A. Use Kinesis Data Firehose to convert the csv files to JSON. Use an AWS Lambda function to store the files in Parquet format.
B. Use Kinesis Data Firehose to convert the csv files to JSON and to store the files in Parquet format.
C. Use Kinesis Data Firehose to invoke an AWS Lambda function that transforms the .csv files to JSON and stores the files in Parquet format.
D. Use Kinesis Data Firehose to invoke an AWS Lambda function that transforms the .csv files to JSON. Use Kinesis Data Firehose to store the files in Parquet format.

Answer: B

Explanation:
The company wants to use Amazon Kinesis Data Firehose to transform CSV files into JSON format and store the files in Apache Parquet format with the least development effort.
Option B: Use Kinesis Data Firehose to convert the CSV files to JSON and to store the files in Parquet format.
Kinesis Data Firehose supports data format conversion natively, including converting incoming CSV data to JSON

format and storing the resulting files in Parquet format in Amazon S3. This solution requires the least development effort because it uses built-in transformation features of Kinesis Data Firehose.

Other options (A, C, D) involve invoking AWS Lambda functions, which would introduce additional complexity and development effort compared to Kinesis Data Firehose's native format conversion capabilities.

Reference:

Amazon Kinesis Data Firehose Documentation

Question: 126

A data engineer maintains a materialized view that is based on an Amazon Redshift database. The view has a column named load_date that stores the date when each row was loaded.

The data engineer needs to reclaim database storage space by deleting all the rows from the materialized view.

Which command will reclaim the MOST database storage space?

A.

```
DELETE FROM materialized_view_name where 1=1
```

B.

```
TRUNCATE materialized_view_name
```

C.

```
VACUUM table_name where load_date<-current_date
```

materializedvtew

D.

```
DELETE FROM materialized_view_name wher e load _date<»current_date
```

A. Option A
B. Option B
C. Option C
D. Option D

Answer: A

Explanation:

To reclaim the most storage space from a materialized view in Amazon Redshift, you should use a DELETE operation that removes all rows from the view. The most efficient way to remove all rows is to use a condition that always evaluates to true, such as 1=1. This will delete all rows without needing to evaluate each row individually based on specific column values like load_date.

Option A: DELETE FROM materialized_view_name WHERE 1=1;

This statement will delete all rows in the materialized view and free up the space. Since materialized views in Redshift store precomputed data, performing a DELETE operation will remove all stored rows.

Other options either involve inappropriate SQL statements (e.g., VACUUM in option C is used for reclaiming

storage space in tables, not materialized views), or they don't remove data effectively in the context of a materialized view (e.g., TRUNCATE cannot be used directly on a materialized view). Reference:

Amazon Redshift Materialized Views Documentation

Deleting Data from Redshift

Question: 127

A company wants to migrate data from an Amazon RDS for PostgreSQL DB instance in the eu-east-1 Region of an AWS account named Account_

A. The company will migrate the data to an Amazon Redshift cluster in the eu-west-1 Region of an AWS account named Account_B.

Which solution will give AWS Database Migration Service (AWS DMS) the ability to replicate data between two data stores?

A. Set up an AWS DMS replication instance in Account_B in eu-west-1.
B. Set up an AWS DMS replication instance in Account_B in eu-east-1.
C. Set up an AWS DMS replication instance in a new AWS account in eu-west-1
D. Set up an AWS DMS replication instance in Account_A in eu-east-1.

Answer: A

Explanation:

To migrate data from an Amazon RDS for PostgreSQL DB instance in the eu-east-1 Region (Account_A) to an Amazon Redshift cluster in the eu-west-1 Region (Account_B), AWS DMS needs a replication instance located in the target region (in this case, eu-west-1) to facilitate the data transfer between regions.

Option A: Set up an AWS DMS replication instance in Account_B in eu-west-1.

Placing the DMS replication instance in the target account and region (Account_B in eu-west-1) is the most efficient solution. The replication instance can connect to the source RDS PostgreSQL in eu-east- 1 and migrate the data to the Redshift cluster in eu-west-1. This setup ensures data is replicated across AWS accounts and regions.

Options B, C, and D place the replication instance in either the wrong account or region, which increases complexity without adding any benefit.

Reference:

AWS Database Migration Service (DMS) Documentation

Cross-Region and Cross-Account Replication

Question: 128

A data engineer uses Amazon Managed Workflows for Apache Airflow (Amazon MWAA) to run data pipelines in an AWS account. A workflow recently failed to run. The data engineer needs to use Apache Airflow logs to diagnose the failure of the workflow. Which log type should the data engineer use to diagnose the cause of the failure?

A. YourEnvironmentName-WebServer
B. YourEnvironmentName-Scheduler
C. YourEnvironmentName-DAGProcessing
D. YourEnvironmentName-Task

Answer: D

Explanation:
In Amazon Managed Workflows for Apache Airflow (MWAA), the type of log that is most useful for diagnosing workflow (DAG) failures is the Task logs. These logs provide detailed information on the execution of each task within the DAG, including error messages, exceptions, and other critical details necessary for diagnosing failures.
Option D: YourEnvironmentName-Task
Task logs capture the output from the execution of each task within a workflow (DAG), which is crucial for understanding what went wrong when a DAG fails. These logs contain detailed execution information, including errors and stack traces, making them the best source for debugging.
Other options (WebServer, Scheduler, and DAGProcessing logs) provide general environment-level logs or logs related to scheduling and DAG parsing, but they do not provide the granular task-level execution details needed for diagnosing workflow failures.
Reference:
Amazon MWAA Logging and Monitoring
Apache Airflow Task Logs

Question: 129

An ecommerce company wants to use AWS to migrate data pipelines from an on-premises environment into the AWS Cloud. The company currently uses a third-party too in the on-premises environment to orchestrate data ingestion processes.
The company wants a migration solution that does not require the company to manage servers. The solution must be able to orchestrate Python and Bash scripts. The solution must not require the company to refactor any code.
Which solution will meet these requirements with the LEAST operational overhead?

A. AWS Lambda
B. Amazon Managed Workflows for Apache Airflow (Amazon MWAA)
C. AWS Step Functions
D. AWS Glue

Answer: B

Explanation:
The ecommerce company wants to migrate its data pipelines into the AWS Cloud without managing servers, and the solution must orchestrate Python and Bash scripts without refactoring code. Amazon Managed Workflows for Apache Airflow (Amazon MWAA) is the most suitable solution for this scenario.
Option B: Amazon Managed Workflows for Apache Airflow (Amazon MWAA)
MWAA is a managed orchestration service that supports Python and Bash scripts via Directed Acyclic Graphs (DAGs) for workflows. It is a serverless, managed version of Apache Airflow, which is commonly used for orchestrating complex data workflows, making it an ideal choice for migrating existing pipelines without refactoring. It supports Python, Bash, and other scripting languages, and the company would not need to manage the underlying infrastructure.
Other options:
AWS Lambda (Option A) is more suited for event-driven workflows but would require breaking down the pipeline

into individual Lambda functions, which may require refactoring.

AWS Step Functions (Option C) is good for orchestration but lacks native support for Python and Bash without using Lambda functions, and it may require code changes.

AWS Glue (Option D) is an ETL service primarily for data transformation and not suitable for orchestrating general scripts without modification.

Reference:

Amazon Managed Workflows for Apache Airflow (MWAA) Documentation

---

Question: 130 A data engineer needs to use Amazon Neptune to develop graph applications.

Which programming languages should the engineer use to develop the graph applications? (Select TWO.)

A. Gremlin

B. SQL

C. ANSI SQL

D. SPARQL

E. Spark SQL

Answer: A, D

Explanation:

Amazon Neptune supports graph applications using Gremlin and SPARQL as query languages. Neptune is a fully managed graph database service that supports both property graph and RDF graph models.

Option A: Gremlin

Gremlin is a query language for property graph databases, which is supported by Amazon Neptune. It allows the traversal and manipulation of graph data in the property graph model.

Option D: SPARQL

SPARQL is a query language for querying RDF graph data in Neptune. It is used to query, manipulate, and retrieve information stored in RDF format.

Other options:

SQL (Option B) and ANSI SQL (Option C) are traditional relational database query languages and are not used for graph databases.

Spark SQL (Option E) is related to Apache Spark for big data processing, not for querying graph databases.

Reference:

Amazon Neptune Documentation

Gremlin Documentation

SPARQL Documentation

Question: 131

A company has an application that uses a microservice architecture. The company hosts the application on an Amazon Elastic Kubernetes Services (Amazon EKS) cluster.

The company wants to set up a robust monitoring system for the application. The company needs to analyze the logs from the EKS cluster and the application. The company needs to correlate the cluster's logs with the application's traces to identify points of failure in the whole application request flow.

Which combination of steps will meet these requirements with the LEAST development effort? (Select TWO.)

A. Use FluentBit to collect logs. Use OpenTelemetry to collect traces.
B. Use Amazon CloudWatch to collect logs. Use Amazon Kinesis to collect traces.

C. Use Amazon CloudWatch to collect logs. Use Amazon Managed Streaming for Apache Kafka (Amazon MSK) to collect traces.
D. Use Amazon OpenSearch to correlate the logs and traces.

E. Use AWS Glue to correlate the logs and traces.

Answer: A, D

Explanation:

Step 1: Log Collection (FluentBit and CloudWatch)

Option A suggests using FluentBit to collect logs and OpenTelemetry to collect traces.

FluentBit is a lightweight log processor that integrates with Amazon EKS to collect and forward logs from Kubernetes clusters. It is widely used with minimal overhead, making it an ideal choice for log collection in this scenario. FluentBit is also natively compatible with AWS services.

OpenTelemetry is a popular framework to collect traces from distributed applications. It provides observability, making it easier to monitor microservices.

This combination allows you to effectively gather both logs and traces with minimal setup and configuration, aligning with the goal of least development effort.

CloudWatch can be used to monitor logs (Option B and C). However, for applications that need more custom and fine-grained control over logging mechanisms, FluentBit and OpenTelemetry are the preferred choice in microservice environments.

Step 2: Log and Trace Correlation (Amazon OpenSearch)

Option D (Amazon OpenSearch) is specifically designed to search, analyze, and visualize logs, metrics, and traces in real-time. OpenSearch allows you to correlate logs and traces effectively.

With Amazon OpenSearch, you can set up dashboards that help in visualizing both logs and traces together, which assists in identifying any failure points across the entire request flow.

It offers integrations with FluentBit and OpenTelemetry, ensuring that both logs from the EKS cluster and application traces are centrally collected, stored, and correlated without additional heavy development.

Step 3: Why Other Options Are Not Suitable

Option B (Amazon Kinesis) is designed for real-time data streaming and analytics but is not as well- suited for tracing microservice requests and logs correlation compared to OpenSearch.

Option C (Amazon MSK) provides a managed Kafka streaming service, but this adds complexity when trying to integrate and correlate logs and traces from a microservice environment. Setting up Kafka requires more development effort compared to using FluentBit and OpenTelemetry.

Option E (AWS Glue) is primarily an ETL (Extract, Transform, Load) service. While Glue is powerful for data processing, it is not a native tool for log and trace correlation, and using it would add unnecessary complexity for this use case.

Conclusion:

To meet the requirements with the least development effort:

Use FluentBit for log collection and OpenTelemetry for tracing (Option A).

Correlate logs and traces using Amazon OpenSearch (Option D).

This approach leverages AWS-native services designed for seamless integration with microservices hosted on Amazon EKS and ensures effective monitoring with minimal overhead.

Question: 132

A company stores customer data that contains personally identifiable information (PII) in an Amazon Redshift cluster. The company's marketing, claims, and analytics teams need to be able to access the customer data.

The marketing team should have access to obfuscated claim information but should have full access to customer contact information.

The claims team should have access to customer information for each claim that the team processes.

The analytics team should have access only to obfuscated PII data.

Which solution will enforce these data access requirements with the LEAST administrative overhead?

A. Create a separate Redshift cluster for each team. Load only the required data for each team. Restrict access to clusters based on the teams.

B. Create views that include required fields for each of the data requirements. Grant the teams access only to the view that each team requires.

C. Create a separate Amazon Redshift database role for each team. Define masking policies that apply for each team separately. Attach appropriate masking policies to each team role.

D. Move the customer data to an Amazon S3 bucket. Use AWS Lake Formation to create a data lake. Use fine-grained security capabilities to grant each team appropriate permissions to access the data.

Answer: B

Explanation:

Step 1: Understand the Data Access Requirements

The question presents distinct access needs for three teams:

Marketing team: Needs full access to customer contact info but only obfuscated claim information.

Claims team: Needs access to customer information relevant to the claims they process.

Analytics team: Needs only obfuscated PII data.

These teams require different levels of access, and the solution needs to enforce data security while keeping administrative overhead low.

Step 2: Why Option B is Correct

Option B (Creating Views) is a common best practice in Amazon Redshift to restrict access to specific data without duplicating data or managing multiple clusters. By creating views:

You can define customized views of the data with obfuscated fields for the analytics team and marketing team while still providing full access where necessary.

Views provide a logical separation of data and allow Redshift administrators to grant access permissions based on roles or groups, ensuring that each team sees only what they are allowed to.

Obfuscation or masking of PII can be easily applied to the views by transforming or hiding sensitive data fields.

This approach avoids the complexity of managing multiple Redshift clusters or S3-based data lakes, which introduces higher operational and administrative overhead.

Step 3: Why Other Options Are Not Ideal

Option A (Separate Redshift Clusters) introduces unnecessary administrative overhead by managing multiple clusters. Maintaining several clusters for each team is costly, redundant, and inefficient.

Option C (Separate Redshift Roles) involves creating multiple roles and managing complex masking policies, which adds to administrative burden and complexity. While Redshift does support columnlevel access control, it's still more overhead than managing simple views.

Option D (Move to S3 and Lake Formation) is a more complex and heavy-handed solution, especially when the data is already stored in Redshift. Migrating the data to S3 and setting up a data lake with Lake Formation introduces significant operational complexity that isn't needed for this specific requirement.

Conclusion:

Creating views in Amazon Redshift allows for flexible, fine-grained access control with minimal overhead, making it the optimal solution to meet the data access requirements of the marketing, claims, and analytics teams.

Question: 133

A company stores customer data in an Amazon S3 bucket. Multiple teams in the company want to use the customer data for downstream analysis. The company needs to ensure that the teams do not have access to personally identifiable information (PII) about the customers.

Which solution will meet this requirement with LEAST operational overhead?

A. Use Amazon Macie to create and run a sensitive data discovery job to detect and remove PII.

B. Use S3 Object Lambda to access the data, and use Amazon Comprehend to detect and remove PII.

C. Use Amazon Kinesis Data Firehose and Amazon Comprehend to detect and remove PII.

D. Use an AWS Glue DataBrew job to store the PII data in a second S3 bucket. Perform analysis on the data that remains in the original S3 bucket.

Answer: D

Explanation:

Step 1: Understanding the Data Use Case

The company has data stored in an Amazon S3 bucket and needs to provide teams access for analysis, ensuring

that PII data is not included in the analysis. The solution should be simple to implement and maintain, ensuring minimal operational overhead.

Step 2: Why Option D is Correct

Option D (AWS Glue DataBrew) allows you to visually prepare and transform data without needing to write code. By using a DataBrew job, the company can:

Automatically detect and separate PII data from non-PII data.

Store PII data in a second S3 bucket for security, while keeping the original S3 bucket clean for analysis.

This approach keeps operational overhead low by utilizing DataBrew's pre-built transformations and the easy-to-use interface for non-technical users. It also ensures compliance by separating sensitive PII data from the main dataset.

Step 3: Why Other Options Are Not Ideal

Option A (Amazon Macie) is a powerful tool for detecting sensitive data, but Macie doesn't inherently remove or mask PII. You would still need additional steps to clean the data after Macie identifies PII.

Option B (S3 Object Lambda with Amazon Comprehend) introduces more complexity by requiring custom logic at the point of data access. Amazon Comprehend can detect PII, but using S3 Object Lambda to filter data would involve more overhead.

Option C (Kinesis Data Firehose and Comprehend) is more suitable for real-time streaming data use cases rather than batch analysis. Setting up and managing a streaming solution like Kinesis adds unnecessary complexity.

Conclusion:

Using AWS Glue DataBrew provides a low-overhead, no-code solution to detect and separate PII data, ensuring the analysis teams only have access to non-sensitive data. This approach is simple, compliant, and easy to manage compared to other options.

Question: 134

A gaming company uses Amazon Kinesis Data Streams to collect clickstream data. The company uses Amazon Kinesis Data Firehose delivery streams to store the data in JSON format in Amazon S3. Data scientists at the

company use Amazon Athena to query the most recent data to obtain business insights.

The company wants to reduce Athena costs but does not want to recreate the data pipeline.

Which solution will meet these requirements with the LEAST management effort?

A. Change the Firehose output format to Apache Parquet. Provide a custom S3 object YYYYMMDD prefix expression and specify a large buffer size. For the existing data, create an AWS Glue extract, transform, and load (ETL) job. Configure the ETL job to combine small JSON files, convert the JSON files to large Parquet files, and add the YYYYMMDD prefix. Use the ALTER TABLE ADD PARTITION statement to reflect the partition on the existing Athena table.

B. Create an Apache Spark job that combines JSON files and converts the JSON files to Apache Parquet files. Launch an Amazon EMR ephemeral cluster every day to run the Spark job to create new Parquet files in a different S3 location. Use the ALTER TABLE SET LOCATION statement to reflect the new S3 location on the existing Athena table.

C. Create a Kinesis data stream as a delivery destination for Firehose. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to run Apache Flink on the Kinesis data stream. Use Flink to aggregate the data and save the data to Amazon S3 in Apache Parquet format with a custom S3 object YYYYMMDD prefix. Use the ALTER TABLE ADD PARTITION statement to reflect the partition on the existing Athena table.

D. Integrate an AWS Lambda function with Firehose to convert source records to Apache Parquet and write them to Amazon S3. In parallel, run an AWS Glue extract, transform, and load (ETL) job to combine the JSON files and convert the JSON files to large Parquet files. Create a custom S3 object YYYYMMDD prefix. Use the ALTER TABLE ADD PARTITION statement to reflect the partition on the existing Athena table.

Answer: A

Explanation:

Step 1: Understanding the Problem

The company collects clickstream data via Amazon Kinesis Data Streams and stores it in JSON format in Amazon S3 using Kinesis Data Firehose. They use Amazon Athena to query the data, but they want to reduce Athena costs while maintaining the same data pipeline.

Since Athena charges based on the amount of data scanned during queries, reducing the data size (by converting JSON to a more efficient format like Apache Parquet) is a key solution to lowering costs.

Step 2: Why Option A is Correct

Option A provides a straightforward way to reduce costs with minimal management overhead:

Changing the Firehose output format to Parquet: Parquet is a columnar data format, which is more compact and efficient than JSON for Athena queries. It significantly reduces the amount of data scanned, which in turn

reduces Athena query costs.

Custom S3 Object Prefix (YYYYMMDD): Adding a date-based prefix helps in partitioning the data, which further improves query efficiency in Athena by limiting the data scanned to only relevant partitions.

AWS Glue ETL Job for Existing Data: To handle existing data stored in JSON format, a one-time AWS Glue ETL job can combine small JSON files, convert them to Parquet, and apply the YYYYMMDD prefix. This ensures consistency in the S3 bucket structure and allows Athena to efficiently query historical data.

ALTER TABLE ADD PARTITION: This command updates Athena's table metadata to reflect the new partitions, ensuring that future queries target only the required data.

Step 3: Why Other Options Are Not Ideal

Option B (Apache Spark on EMR) introduces higher management effort by requiring the setup of Apache Spark jobs and an Amazon EMR cluster. While it achieves the goal of converting JSON to Parquet, it involves running and maintaining an EMR cluster, which adds operational complexity.

Option C (Kinesis and Apache Flink) is a more complex solution involving Apache Flink, which adds a real-time streaming layer to aggregate data. Although Flink is a powerful tool for stream processing, it adds unnecessary overhead in this scenario since the company already uses Kinesis Data Firehose for batch delivery to S3.

Option D (AWS Lambda with Firehose) suggests using AWS Lambda to convert records in real time. While Lambda can work in some cases, it's generally not the best tool for handling large-scale data transformations like JSON-to-Parquet conversion due to potential scaling and invocation limitations. Additionally, running parallel Glue jobs further complicates the setup.

Step 4: How Option A Minimizes Costs

By using Apache Parquet, Athena queries become more efficient, as Athena will scan significantly less data, directly reducing query costs.

Firehose natively supports Parquet as an output format, so enabling this conversion in Firehose requires minimal effort. Once set, new data will automatically be stored in Parquet format in S3, without requiring any custom coding or ongoing management.

The AWS Glue ETL job for historical data ensures that existing JSON files are also converted to Parquet format, ensuring consistency across the data stored in S3.

Conclusion:

Option A meets the requirement to reduce Athena costs without recreating the data pipeline, using Firehose's native support for Apache Parquet and a simple one-time AWS Glue ETL job for existing data. This approach involves minimal management effort compared to the other solutions.

www.atmicnetworks.com

Question: 135

A retail company stores customer data in an Amazon S3 bucket. Some of the customer data contains personally identifiable information (PII) about customers. The company must not share PII data with business partners.

A data engineer must determine whether a dataset contains PII before making objects in the dataset available to business partners.

Which solution will meet this requirement with the LEAST manual intervention?

A. Configure the S3 bucket and S3 objects to allow access to Amazon Macie. Use automated sensitive data discovery in Macie.

B. Configure AWS CloudTrail to monitor S3 PUT operations. Inspect the CloudTrail trails to identify operations that save PII.

C. Create an AWS Lambda function to identify PII in S3 objects. Schedule the function to run periodically.

D. Create a table in AWS Glue Data Catalog. Write custom SQL queries to identify PII in the table. Use Amazon Athena to run the queries.

Answer: A

Explanation:

Amazon Macie is a fully managed data security and privacy service that uses machine learning to automatically discover, classify, and protect sensitive data in AWS, such as PII. By configuring Macie for automated sensitive data discovery, the company can minimize manual intervention while ensuring PII is identified before data is shared.

Question: 136

A company uses Amazon S3 buckets, AWS Glue tables, and Amazon Athena as components of a data lake. Recently, the company expanded its sales range to multiple new states. The company wants to introduce state names as a new partition to the existing S3 bucket, which is currently partitioned by date.

The company needs to ensure that additional partitions will not disrupt daily synchronization between the AWS Glue Data Catalog and the S3 buckets.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use the AWS Glue API to manually update the Data Catalog.

B. Run an MSCK REPAIR TABLE command in Athena.

C. Schedule an AWS Glue crawler to periodically update the Data Catalog.

D. Run a REFRESH TABLE command in Athena.

Answer: C

Explanation:

Scheduling an AWS Glue crawler to periodically update the Data Catalog automates the process of detecting new partitions and updating the catalog, which minimizes manual maintenance and operational overhead.

Question: 137

A company uses a variety of AWS and third-party data stores. The company wants to consolidate all the data into a

central data warehouse to perform analytics. Users need fast response times for analytics queries.

The company uses Amazon QuickSight in direct query mode to visualize the data. Users normally run queries during a few hours each day with unpredictable spikes.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use Amazon Redshift Serverless to load all the data into Amazon Redshift managed storage (RMS).

B. Use Amazon Athena to load all the data into Amazon S3 in Apache Parquet format.

C. Use Amazon Redshift provisioned clusters to load all the data into Amazon Redshift managed storage (RMS).

D. Use Amazon Aurora PostgreSQL to load all the data into Aurora.

Answer: A

Explanation:

Problem Analysis:

The company requires a centralized data warehouse for consolidating data from various sources.

They use Amazon QuickSight in direct query mode, necessitating fast response times for analytical queries.

Users query the data intermittently, with unpredictable spikes during the day.

Operational overhead should be minimal.

Key Considerations:

The solution must support fast, SQL-based analytics.

It must handle unpredictable spikes efficiently.

Must integrate seamlessly with QuickSight for direct querying.

Minimize operational complexity and scaling concerns.

Solution Analysis:

Option A: Amazon Redshift Serverless

Redshift Serverless eliminates the need for provisioning and managing clusters.

Automatically scales compute capacity up or down based on query demand.

Reduces operational overhead by handling performance optimization.

Fully integrates with Amazon QuickSight, ensuring low-latency analytics.

Reduces costs as it charges only for usage, making it ideal for workloads with intermittent spikes.

Option B: Amazon Athena with S3 (Apache Parquet)

Athena supports querying data directly from S3 in Parquet format.

While it's cost-effective, performance depends on the size and complexity of the data.

It is not optimized for high-speed analytics needed by QuickSight in direct query mode.

Option C: Amazon Redshift Provisioned Clusters

Requires manual cluster provisioning, scaling, and maintenance.

Higher operational overhead compared to Redshift Serverless.

Option D: Amazon Aurora PostgreSQL

Aurora is optimized for transactional databases, not data warehousing or analytics.

Does not meet the requirement for fast analytics queries.

Final Recommendation:

Amazon Redshift Serverless is the best choice for this use case because it provides fast analytics, integrates natively with QuickSight, and minimizes operational complexity while efficiently handling unpredictable spikes.

Reference:

Amazon Redshift Serverless Overview

Amazon QuickSight and Redshift Integration

Athena vs. Redshift

Question: 138

A company receives a data file from a partner each day in an Amazon S3 bucket. The company uses a daily AW5 Glue extract, transform, and load (ETL) pipeline to clean and transform each data file. The output of the ETL pipeline is written to a CSV file named Dairy.csv in a second 53 bucket.

Occasionally, the daily data file is empty or is missing values for required fields. When the file is missing data, the company can use the previous day's CSV file.

A data engineer needs to ensure that the previous day's data file is overwritten only if the new daily file is complete and valid.

Which solution will meet these requirements with the LEAST effort?

A. Invoke an AWS Lambda function to check the file for missing data and to fill in missing values in required fields.

B. Configure the AWS Glue ETL pipeline to use AWS Glue Data Quality rules. Develop rules in Data Quality Definition Language (DQDL) to check for missing values in required files and empty files.

C. Use AWS Glue Studio to change the code in the ETL pipeline to fill in any missing values in the required fields with the most common values for each field.

D. Run a SQL query in Amazon Athena to read the CSV file and drop missing rows. Copy the corrected CSV file to the second S3 bucket.

Answer: B

Explanation:

Problem Analysis:

The company runs a daily AWS Glue ETL pipeline to clean and transform files received in an S3 bucket.

If a file is incomplete or empty, the previous day's file should be retained.

Need a solution to validate files before overwriting the existing file.

Key Considerations:

Automate data validation with minimal human intervention.

Use built-in AWS Glue capabilities for ease of integration.

Ensure robust validation for missing or incomplete data.

Solution Analysis:

Option A: Lambda Function for Validation

Lambda can validate files, but it would require custom code.

Does not leverage AWS Glue's built-in features, adding operational complexity.

Option B: AWS Glue Data Quality Rules

AWS Glue Data Quality allows defining Data Quality Definition Language (DQDL) rules.

Rules can validate if required fields are missing or if the file is empty.

Automatically integrates into the existing ETL pipeline.

If validation fails, retain the previous day's file.

Option C: AWS Glue Studio with Filling Missing Values

Modifying ETL code to fill missing values with most common values risks introducing inaccuracies.

Does not handle empty files effectively.

Option D: Athena Query for Validation

Athena can drop rows with missing values, but this is a post-hoc solution.

Requires manual intervention to copy the corrected file to S3, increasing complexity.

Final Recommendation:

Use AWS Glue Data Quality to define validation rules in DQDL for identifying missing or incomplete data.

This solution integrates seamlessly with the ETL pipeline and minimizes manual effort.

Implementation Steps:

Enable AWS Glue Data Quality in the existing ETL pipeline.

Define DQDL Rules, such as:

Check if a file is empty.

Verify required fields are present and non-null.

Configure the pipeline to proceed with overwriting only if the file passes validation.

In case of failure, retain the previous day's file.

Reference:

AWS Glue Data Quality Overview

Defining DQDL Rules

AWS Glue Studio Documentation

Question: 139_____

A company has a gaming application that stores data in Amazon DynamoDB tables. A data engineer needs to ingest the game data into an Amazon OpenSearch Service cluster. Data updates must occur in near real time.

Which solution will meet these requirements?

A. Use AWS Step Functions to periodically export data from the Amazon DynamoDB tables to an Amazon S3 bucket. Use an AWS Lambda function to load the data into Amazon OpenSearch Service.

B. Configure an AW5 Glue job to have a source of Amazon DynamoDB and a destination of Amazon OpenSearch Service to transfer data in near real time.

C. Use Amazon DynamoDB Streams to capture table changes. Use an AWS Lambda function to process and update the data in Amazon OpenSearch Service.

D. Use a custom OpenSearch plugin to sync data from the Amazon DynamoDB tables.

Answer: C

Explanation:

Problem Analysis:

The company uses DynamoDB for gaming data storage and needs to ingest data into Amazon OpenSearch Service in near real time.

Data updates must propagate quickly to OpenSearch for analytics or search purposes.

Key Considerations:

DynamoDB Streams provide near-real-time capture of table changes (inserts, updates, and deletes).

Integration with AWS Lambda allows seamless processing of these changes.

OpenSearch offers APIs for indexing and updating documents, which Lambda can invoke.

Solution Analysis:

Option A: Step Functions with Periodic Export

Not suitable for near-real-time updates; introduces significant latency.

Operationally complex to manage periodic exports and S3 data ingestion.

Option B: AWS Glue Job

AWS Glue is designed for ETL workloads but lacks real-time processing capabilities.

Option C: DynamoDB Streams + Lambda

DynamoDB Streams capture changes in near real time.

Lambda can process these streams and use the OpenSearch API to update the index.

This approach provides low latency and seamless integration with minimal operational overhead.

Option D: Custom OpenSearch Plugin

Writing a custom plugin adds complexity and is unnecessary with existing AWS integrations.

Implementation Steps:

Enable DynamoDB Streams for the relevant DynamoDB tables.

Create a Lambda function to process stream records:

Parse insert, update, and delete events.

Use OpenSearch APIs to index or update documents based on the event type.

Set up a trigger to invoke the Lambda function whenever there are changes in the DynamoDB Stream.

Monitor and log errors for debugging and operational health.

Reference:

Amazon DynamoDB Streams Documentation

AWS Lambda and DynamoDB Integration

Amazon OpenSearch Service APIs

Question: 140

A company uses Amazon S3 to store data and Amazon QuickSight to create visualizations.

The company has an S3 bucket in an AWS account named Hub-Account. The S3 bucket is encrypted by an AWS Key Management Service (AWS KMS) key. The company's QuickSight instance is in a separate account named BI-Account

The company updates the S3 bucket policy to grant access to the QuickSight service role. The company wants to enable cross-account access to allow QuickSight to interact with the S3 bucket.

Which combination of steps will meet this requirement? (Select TWO.)

A. Use the existing AWS KMS key to encrypt connections from QuickSight to the S3 bucket.

B. Add the 53 bucket as a resource that the QuickSight service role can access.

C. Use AWS Resource Access Manager (AWS RAM) to share the S3 bucket with the Bl-Account account.

D. Add an IAM policy to the QuickSight service role to give QuickSight access to the KMS key that encrypts the S3 bucket.

E. Add the KMS key as a resource that the QuickSight service role can access.

Answer: DE

Explanation:

Problem Analysis:

The company needs cross-account access to allow QuickSight in BI-Account to interact with an S3 bucket in Hub-Account.

The bucket is encrypted with an AWS KMS key.

Appropriate permissions must be set for both S3 access and KMS decryption.

Key Considerations:

QuickSight requires IAM permissions to access S3 data and decrypt files using the KMS key.

Both S3 and KMS permissions need to be properly configured across accounts.

Solution Analysis:

Option A: Use Existing KMS Key for Encryption

While the existing KMS key is used for encryption, it must also grant decryption permissions to QuickSight.

Option B: Add S3 Bucket to QuickSight Role

Granting S3 bucket access to the QuickSight service role is necessary for cross-account access.

Option C: AWS RAM for Bucket Sharing

AWS RAM is not required; bucket policies and IAM roles suffice for granting cross-account access.

Option D: IAM Policy for KMS Access

QuickSight's service role in BI-Account needs explicit permissions to use the KMS key for decryption.

Option E: Add KMS Key as Resource for Role

The KMS key must explicitly list the QuickSight role as an entity that can access it.

Implementation Steps:

S3 Bucket Policy in Hub-Account:

Add a policy to the S3 bucket granting the QuickSight service role access:

www.atmicnetworks.com

```
json

{
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Principal": { "AWS": "arn:aws:iam::<BI-Account-ID>:role/service-role/QuickSightRole" },
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::<Bucket-Name>/*"
  }
]
}
```

KMS Key Policy in Hub-Account:

Add permissions for the QuickSight role:

```
{
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Principal": { "AWS": "arn:aws:iam::<BI-Account-ID>:role/service-role/QuickSightRole" },
    "Action": [
      "kms:Decrypt",
      "kms:DescribeKey"
    ],
```

```
    "Resource": "*"
  }
]
}
```

IAM Policy for QuickSight Role in BI-Account:

Attach the following policy to the QuickSight service role:

```
{
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
    "s3:GetObject",
    "kms:Decrypt"
    ],
    "Resource": [
      "arn:aws:s3:::<Bucket-Name>/*",
      "arn:aws:kms:<region>:<Hub-Account-ID>:key/<KMS-Key-ID>"
    ]
  }
]
}
```

Reference:

Setting Up Cross-Account S3 Access

AWS KMS Key Policy Examples

www.atmicnetworks.com

Amazon QuickSight Cross-Account Access

Question: 141

A company stores server logs in an Amazon 53 bucket. The company needs to keep the logs for 1 year. The logs are not required after 1 year.

A data engineer needs a solution to automatically delete logs that are older than 1 year.

Which solution will meet these requirements with the LEAST operational overhead?

A. Define an S3 Lifecycle configuration to delete the logs after 1 year.

B. Create an AWS Lambda function to delete the logs after 1 year.

C. Schedule a cron job on an Amazon EC2 instance to delete the logs after 1 year.

D. Configure an AWS Step Functions state machine to delete the logs after 1 year.

Answer: B

Explanation:

Problem Analysis:

The company uses AWS Glue for ETL pipelines and requires automatic data quality checks during pipeline execution.

The solution must integrate with existing AWS Glue pipelines and evaluate data quality rules based on predefined thresholds.

Key Considerations:

Ensure minimal implementation effort by leveraging built-in AWS Glue features.

Use a standardized approach for defining and evaluating data quality rules.

Avoid custom libraries or external frameworks unless absolutely necessary.

Solution Analysis:

Option A: SQL Transform

Adding SQL transforms to define and evaluate data quality rules is possible but requires writing complex queries for each rule.

Increases operational overhead and deviates from Glue's declarative approach.

Option B: Evaluate Data Quality Transform with DQDL

AWS Glue provides a built-in Evaluate Data Quality transform.

Allows defining rules in Data Quality Definition Language (DQDL), a concise and declarative way to define quality checks.

Fully integrated with Glue Studio, making it the least effort solution.

Option C: Custom Transform with PyDeequ

PyDeequ is a powerful library for data quality checks but requires custom code and integration.

Increases implementation effort compared to Glue's native capabilities.

Option D: Custom Transform with Great Expectations

Great Expectations is another powerful library for data quality but adds complexity and external dependencies.

Final Recommendation:

Use Evaluate Data Quality transform in AWS Glue.

Define rules in DQDL for checking thresholds, null values, or other quality criteria.

This approach minimizes development effort and ensures seamless integration with AWS Glue.

Reference:

AWS Glue Data Quality Overview

DQDL Syntax and Examples

Glue Studio Transformations

Question: 142

A company wants to analyze sales records that the company stores in a MySQL database. The company wants to correlate the records with sales opportunities identified by Salesforce.

The company receives 2 GB erf sales records every day. The company has 100 GB of identified sales opportunities.

A data engineer needs to develop a process that will analyze and correlate sales records and sales opportunities. The process must run once each night.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use Amazon Managed Workflows for Apache Airflow (Amazon MWAA) to fetch both datasets. Use AWS Lambda functions to correlate the datasets. Use AWS Step Functions to orchestrate the process.

B. Use Amazon AppFlow to fetch sales opportunities from Salesforce. Use AWS Glue to fetch sales records from the MySQL database. Correlate the sales records with the sales opportunities. Use Amazon Managed Workflows for Apache Airflow (Amazon MWAA) to orchestrate the process.

C. Use Amazon AppFlow to fetch sales opportunities from Salesforce. Use AWS Glue to fetch sales records from the MySQL database. Correlate the sales records with sales opportunities. Use AWS Step Functions to orchestrate the process.

D. Use Amazon AppFlow to fetch sales opportunities from Salesforce. Use Amazon Kinesis Data Streams to fetch sales records from the MySQL database. Use Amazon Managed Service for Apache Flink to correlate the datasets. Use AWS Step Functions to orchestrate the process.

Answer: C

Explanation:

Problem Analysis:

The company processes 2 GB of daily sales records and 100 GB of Salesforce sales opportunities.

The goal is to analyze and correlate the two datasets with low operational overhead.

The process must run once nightly.

Key Considerations:

Amazon AppFlow simplifies data integration with Salesforce.

AWS Glue can extract data from MySQL and perform ETL operations.

Step Functions can orchestrate workflows with minimal manual intervention.

Apache Airflow and Flink add complexity, which conflicts with the requirement for low operational overhead.

Solution Analysis:

Option A: MWAA + Lambda + Step Functions

Requires custom Lambda code for dataset correlation, increasing development and operational complexity.

Option B: AppFlow + Glue + MWAA

MWAA adds orchestration overhead compared to the simpler Step Functions.

Option C: AppFlow + Glue + Step Functions

AppFlow fetches Salesforce data, Glue extracts MySQL data, and Step Functions orchestrate the entire process.

Minimal setup and operational overhead, making it the best choice.

Option D: AppFlow + Kinesis + Flink + Step Functions

Using Kinesis and Flink for batch processing introduces unnecessary complexity.

Final Recommendation:

Use Amazon AppFlow to fetch Salesforce data, AWS Glue to process MySQL data, and Step Functions for orchestration.

Reference:

Amazon AppFlow Overview

AWS Glue ETL Documentation

AWS Step Functions

Question: 143

A company has a data lake in Amazon 53. The company uses AWS Glue to catalog data and AWS Glue Studio to implement data extract, transform, and load (ETL) pipelines.

The company needs to ensure that data quality issues are checked every time the pipelines run. A data engineer must enhance the existing pipelines to evaluate data quality rules based on predefined thresholds.

Which solution will meet these requirements with the LEAST implementation effort?

A. Add a new transform that is defined by a SQL query to each Glue ETL job. Use the SQL query to implement a ruleset that includes the data quality rules that need to be evaluated.

B. Add a new Evaluate Data Quality transform to each Glue ETL job. Use Data Quality Definition Language (DQDL) to implement a ruleset that includes the data quality rules that need to be evaluated.

C. Add a new custom transform to each Glue ETL job. Use the PyDeequ library to implement a ruleset that includes the data quality rules that need to be evaluated.

D. Add a new custom transform to each Glue ETL job. Use the Great Expectations library to implement a ruleset that includes the data quality rules that need to be evaluated.

Answer: B

Explanation:

Problem Analysis:

The company uses AWS Glue for ETL pipelines and must enforce data quality checks during pipeline execution.

The goal is to implement quality checks with minimal implementation effort.

Key Considerations:

AWS Glue provides an Evaluate Data Quality transform that allows for defining quality checks directly in the pipeline.

DQDL (Data Quality Definition Language) simplifies the process by allowing declarative rule definitions.

Solution Analysis:

Option A: SQL Transform

SQL queries can implement rules but require manual effort for each rule and do not integrate natively with Glue.

Option B: Evaluate Data Quality Transform + DQDL

AWS Glue's built-in Evaluate Data Quality transform is designed for this use case.

Allows defining thresholds and rules in DQDL with minimal coding effort.

Option C: Custom Transform with PyDeequ

PyDeequ is a powerful library but adds unnecessary complexity compared to Glue's native features.

Option D: Custom Transform with Great Expectations

Similar to PyDeequ, Great Expectations adds operational complexity and external dependencies.

Final Recommendation:

Use the Evaluate Data Quality transform with DQDL to implement data quality rules in AWS Glue pipelines.

Reference:

AWS Glue Data Quality

DQDL Syntax and Examples

AWS Glue Studio Documentation

Question: 144

A company is building an inventory management system and an inventory reordering system to automatically reorder products. Both systems use Amazon Kinesis Data Streams. The inventory management system uses the

Amazon Kinesis Producer Library (KPL) to publish data to a stream. The inventory reordering system uses the Amazon Kinesis Client Library (KCL) to consume data from the stream. The company configures the stream to scale up and down as needed.

Before the company deploys the systems to production, the company discovers that the inventory reordering system received duplicated data.

Which factors could have caused the reordering system to receive duplicated data? (Select TWO.)

A. The producer experienced network-related timeouts.

B. The stream's value for the IteratorAgeMilliseconds metric was too high.

C. There was a change in the number of shards, record processors, or both.

D. The AggregationEnabled configuration property was set to true.

E. The max_records configuration property was set to a number that was too high.

Answer: A, C

Explanation:

Problem Analysis:

The company uses Kinesis Data Streams for both inventory management and reordering.

The Kinesis Producer Library (KPL) publishes data, and the Kinesis Client Library (KCL) consumes data.

Duplicate records were observed in the inventory reordering system.

Key Considerations:

Kinesis streams are designed for durability but may produce duplicates under certain conditions.

Factors such as network timeouts, shard splits, or changes in record processors can cause duplication.

Solution Analysis:

Option A: Network-Related Timeouts

If the producer (KPL) experiences network timeouts, it retries data submission, potentially causing duplicates.

Option B: High IteratorAgeMilliseconds

High iterator age suggests delays in processing but does not directly cause duplication.

Option C: Changes in Shards or Processors

Changes in the number of shards or record processors can lead to re-processing of records, causing duplication.

Option D: AggregationEnabled Set to True

AggregationEnabled controls the aggregation of multiple records into one, but it does not cause duplication.

Option E: High max_records Value

A high max_records value increases batch size but does not lead to duplication.

Final Recommendation:

Network-related timeouts and changes in shards or processors are the most likely causes of duplicate data in this scenario.

Reference:

Amazon Kinesis Data Streams Best Practices

Kinesis Producer Library (KPL) Overview

Kinesis Client Library (KCL) Overview

Question: 145 _____

A company wants to migrate an application and an on-premises Apache Kafka server to AWS. The application processes incremental updates that an on-premises Oracle database sends to the Kafka server. The company wants to use the replatform migration strategy instead of the refactor strategy.

Which solution will meet these requirements with the LEAST management overhead?

A. Amazon Kinesis Data Streams

B. Amazon Managed Streaming for Apache Kafka (Amazon MSK) provisioned cluster

C. Amazon Data Firehose

D. Amazon Managed Streaming for Apache Kafka (Amazon MSK) Serverless

_____ **Answer: D**

Explanation:

Problem Analysis:

The company needs to migrate both an application and an on-premises Apache Kafka server to AWS. Incremental

updates from an on-premises Oracle database are processed by Kafka.

The solution must follow a replatform migration strategy, prioritizing minimal changes and low management overhead.

Key Considerations:

Replatform Strategy: This approach keeps the application and architecture as close to the original as possible, reducing the need for refactoring.

The solution must provide a managed Kafka service to minimize operational burden.

Low overhead solutions like serverless services are preferred.

Solution Analysis:

Option A: Kinesis Data Streams

Kinesis Data Streams is an AWS-native streaming service but is not a direct substitute for Kafka.

This option would require significant application refactoring, which does not align with the replatform strategy.

Option B: MSK Provisioned Cluster

Managed Kafka service with fully configurable clusters.

Provides the same Kafka APIs but requires cluster management (e.g., scaling, patching), increasing management overhead.

Option C: Amazon Kinesis Data Firehose

Kinesis Data Firehose is designed for data delivery rather than real-time streaming and processing.

Not suitable for Kafka-based applications.

Option D: MSK Serverless

MSK Serverless eliminates the need for cluster management while maintaining compatibility with Kafka APIs.

Automatically scales based on workload, reducing operational overhead.

Ideal for replatform migrations, as it requires minimal changes to the application.

Final Recommendation:

Amazon MSK Serverless is the best solution for migrating the Kafka server and application with minimal changes and the least management overhead.

Reference:

Amazon MSK Serverless Overview

Comparison of Amazon MSK and Kinesis

Question: 146

A data engineer is building an automated extract, transform, and load (ETL) ingestion pipeline by using AWS Glue. The pipeline ingests compressed files that are in an Amazon S3 bucket. The ingestion pipeline must support incremental data processing.

Which AWS Glue feature should the data engineer use to meet this requirement?

A. Workflows

B. Triggers

C. Job bookmarks

D. Classifiers

Answer: C

Explanation:

Problem Analysis:

The pipeline processes compressed files in S3 and must support incremental data processing.

AWS Glue features must facilitate tracking progress to avoid reprocessing the same data.

Key Considerations:

Incremental data processing requires tracking which files or partitions have already been processed.

The solution must be automated and efficient for large-scale ETL jobs.

Solution Analysis:

Option A: Workflows

Workflows organize and orchestrate multiple Glue jobs but do not track progress for incremental data processing.

Option B: Triggers

Triggers initiate Glue jobs based on a schedule or events but do not track which data has been processed.

Option C: Job Bookmarks

Job bookmarks track the state of the data that has been processed, enabling incremental processing.

Automatically skip files or partitions that were previously processed in Glue jobs.

Option D: Classifiers

Classifiers determine the schema of incoming data but do not handle incremental processing.

Final Recommendation:

Job bookmarks are specifically designed to enable incremental data processing in AWS Glue ETL pipelines.

Reference:

AWS Glue Job Bookmarks Documentation

AWS Glue ETL Features

Question: 147

A company receives test results from testing facilities that are located around the world. The company stores the test results in millions of 1 KB JSON files in an Amazon S3 bucket. A data engineer needs to process the files, convert them into Apache Parquet format, and load them into Amazon Redshift tables. The data engineer uses AWS Glue to process the files, AWS Step Functions to orchestrate the processes, and Amazon EventBridge to schedule jobs.

The company recently added more testing facilities. The time required to process files is increasing. The data engineer must reduce the data processing time.

Which solution will MOST reduce the data processing time?

A. Use AWS Lambda to group the raw input files into larger files. Write the larger files back to Amazon S3. Use AWS Glue to process the files. Load the files into the Amazon Redshift tables.

B. Use the AWS Glue dynamic frame file-grouping option to ingest the raw input files. Process the files. Load the files into the Amazon Redshift tables.

C. Use the Amazon Redshift COPY command to move the raw input files from Amazon S3 directly into the Amazon Redshift tables. Process the files in Amazon Redshift.

D. Use Amazon EMR instead of AWS Glue to group the raw input files. Process the files in Amazon EMR. Load the files into the Amazon Redshift tables.

Answer: B

Explanation:

Problem Analysis:

Millions of 1 KB JSON files in S3 are being processed and converted to Apache Parquet format using AWS Glue.

Processing time is increasing due to the additional testing facilities.

The goal is to reduce processing time while using the existing AWS Glue framework.

Key Considerations:

AWS Glue offers the dynamic frame file-grouping feature, which consolidates small files into larger, more efficient datasets during processing.

Grouping smaller files reduces overhead and speeds up processing.

Solution Analysis:

Option A: Lambda for File Grouping

Using Lambda to group files would add complexity and operational overhead. Glue already offers built-in grouping functionality.

Option B: AWS Glue Dynamic Frame File-Grouping

This option directly addresses the issue by grouping small files during Glue job execution.

Minimizes data processing time with no extra overhead.

Option C: Redshift COPY Command

COPY directly loads raw files but is not designed for pre-processing (conversion to Parquet).

Option D: Amazon EMR

While EMR is powerful, replacing Glue with EMR increases operational complexity.

Final Recommendation:

Use AWS Glue dynamic frame file-grouping for optimized data ingestion and processing.

Reference:

AWS Glue Dynamic Frames

Optimizing Glue Performance

www.atmicnetworks.com

Question: 148

A transportation company wants to track vehicle movements by capturing geolocation records. The records are 10 bytes in size. The company receives up to 10,000 records every second. Data transmission delays of a few minutes are acceptable because of unreliable network conditions.

The transportation company wants to use Amazon Kinesis Data Streams to ingest the geolocation data. The company needs a reliable mechanism to send data to Kinesis Data Streams. The company needs to maximize the throughput efficiency of the Kinesis shards.

Which solution will meet these requirements in the MOST operationally efficient way?

A. Kinesis Agent

B. Kinesis Producer Library (KPL)

C. Amazon Data Firehose

D. Kinesis SDK

Answer: B

Explanation:

Problem Analysis:

The company ingests geolocation records (10 bytes each) at 10,000 records per second into Kinesis Data Streams.

Data transmission delays are acceptable, but the solution must maximize throughput efficiency.

Key Considerations:

The Kinesis Producer Library (KPL) batches records and uses aggregation to optimize shard throughput.

Efficiently handles high-throughput scenarios with minimal operational overhead.

Solution Analysis:

Option A: Kinesis Agent

Designed for file-based ingestion; not optimized for geolocation records.

Option B: KPL

Aggregates records into larger payloads, significantly improving shard throughput.

Suitable for applications generating small, high-frequency records.

Option C: Kinesis Firehose

Firehose is for delivery to destinations like S3 or Redshift and is not optimized for direct ingestion to Kinesis Data Streams.

Option D: Kinesis SDK

The SDK lacks advanced features like aggregation, resulting in lower throughput efficiency.

Final Recommendation:

Use Kinesis Producer Library (KPL) for its built-in aggregation and batching capabilities.

Reference:

Kinesis Producer Library (KPL) Overview

Best Practices for Amazon Kinesis

Question: 149

A retail company stores data from a product lifecycle management (PLM) application in an onpremises MySQL database. The PLM application frequently updates the database when transactions occur.

The company wants to gather insights from the PLM application in near real time. The company wants to integrate the insights with other business datasets and to analyze the combined dataset by using an Amazon Redshift data warehouse.

The company has already established an AWS Direct Connect connection between the on-premises infrastructure and AWS.

Which solution will meet these requirements with the LEAST development effort?

A. Run a scheduled AWS Glue extract, transform, and load (ETL) job to get the MySQL database updates by using a Java Database Connectivity (JDBC) connection. Set Amazon Redshift as the destination for the ETL job.

B. Run a full load plus CDC task in AWS Database Migration Service (AWS DMS) to continuously replicate the MySQL database changes. Set Amazon Redshift as the destination for the task.

C. Use the Amazon AppFlow SDK to build a custom connector for the MySQL database to continuously replicate the database changes. Set Amazon Redshift as the destination for the connector.

D. Run scheduled AWS DataSync tasks to synchronize data from the MySQL database. Set Amazon Redshift as the destination for the tasks.

Answer: B

Explanation:

Problem Analysis:

The company needs near real-time replication of MySQL updates to Amazon Redshift.

Minimal development effort is required for this solution.

Key Considerations:

AWS DMS provides a full load + CDC (Change Data Capture) mode for continuous replication of database changes.

DMS integrates natively with both MySQL and Redshift, simplifying setup.

Solution Analysis:

Option A: AWS Glue Job

Glue is batch-oriented and does not support near real-time replication.

Option B: DMS with Full Load + CDC

Efficiently handles initial database load and continuous updates.

Requires minimal setup and operational overhead.

Option C: AppFlow SDK

AppFlow is not designed for database replication. Custom connectors increase development effort.

Option D: DataSync

DataSync is for file synchronization and not suitable for database updates.

Final Recommendation:

Use AWS DMS in full load + CDC mode for continuous replication.

Reference:

AWS Database Migration Service Documentation

Setting Up DMS with Redshift

Question: 150

A company has a data warehouse in Amazon Redshift. To comply with security regulations, the company needs to log and store all user activities and connection activities for the data warehouse.

Which solution will meet these requirements?

A. Create an Amazon S3 bucket. Enable logging for the Amazon Redshift cluster. Specify the S3 bucket in the logging configuration to store the logs.

B. Create an Amazon Elastic File System (Amazon EFS) file system. Enable logging for the Amazon Redshift cluster. Write logs to the EFS file system.

C. Create an Amazon Aurora MySQL database. Enable logging for the Amazon Redshift cluster. Write the logs to a table in the Aurora MySQL database.

D. Create an Amazon Elastic Block Store (Amazon EBS) volume. Enable logging for the Amazon Redshift cluster. Write the logs to the EBS volume.

Answer: A

Explanation:

Problem Analysis:

The company must log all user activities and connection activities in Amazon Redshift for security compliance.

Key Considerations:

Redshift supports audit logging, which can be configured to write logs to an S3 bucket.

S3 provides durable, scalable, and cost-effective storage for logs.

Solution Analysis:

Option A: S3 for Logging

Standard approach for storing Redshift logs.

Easy to set up and manage with minimal cost.

Option B: Amazon EFS

EFS is unnecessary for this use case and less cost-efficient than S3.

Option C: Aurora MySQL

Using a database to store logs increases complexity and cost.

Option D: EBS Volume

EBS is not a scalable option for log storage compared to S3.

Final Recommendation:

Enable Redshift audit logging and specify an S3 bucket as the destination.

Reference:

Amazon Redshift Audit Logging

Storing Logs in Amazon S3

Question: 151

A mobile gaming company wants to capture data from its gaming app. The company wants to make the data available to three internal consumers of the data. The data records are approximately 20 KB in size.

The company wants to achieve optimal throughput from each device that runs the gaming app. Additionally, the company wants to develop an application to process data streams. The streamprocessing application must have dedicated throughput for each internal consumer.

Which solution will meet these requirements?

A. Configure the mobile app to call the PutRecords API operation to send data to Amazon Kinesis Data Streams. Use the enhanced fan-out feature with a stream for each internal consumer.

B. Configure the mobile app to call the PutRecordBatch API operation to send data to Amazon Data Firehose. Submit an AWS Support case to turn on dedicated throughput for the company's AWS account. Allow each internal consumer to access the stream.

C. Configure the mobile app to use the Amazon Kinesis Producer Library (KPL) to send data to Amazon Data Firehose. Use the enhanced fan-out feature with a stream for each internal consumer.

D. Configure the mobile app to call the PutRecords API operation to send data to Amazon Kinesis Data Streams. Host the stream-processing application for each internal consumer on Amazon EC2 instances. Configure auto scaling for the EC2 instances.

Answer: A

Explanation:

Problem Analysis:

Input Requirements: Gaming app generates approximately 20 KB data records, which must be ingested and made available to three internal consumers with dedicated throughput.

Key Requirements:

High throughput for ingestion from each device.

Dedicated processing bandwidth for each consumer.

Key Considerations:

Amazon Kinesis Data Streams supports high-throughput ingestion with PutRecords API for batch writes.

The Enhanced Fan-Out feature provides dedicated throughput to each consumer, avoiding bandwidth contention.

This solution avoids bottlenecks and ensures optimal throughput for the gaming application and consumers.

Solution Analysis:

Option A: Kinesis Data Streams + Enhanced Fan-Out

PutRecords API is designed for batch writes, improving ingestion performance.

Enhanced Fan-Out allows each consumer to process the stream independently with dedicated throughput.

Option B: Data Firehose + Dedicated Throughput Request

Firehose is not designed for real-time stream processing or fan-out. It delivers data to destinations like S3, Redshift, or OpenSearch, not multiple independent consumers.

Option C: Data Firehose + Enhanced Fan-Out

Firehose does not support enhanced fan-out. This option is invalid.

Option D: Kinesis Data Streams + EC2 Instances

Hosting stream-processing applications on EC2 increases operational overhead compared to native enhanced fan-out.

Final Recommendation:

Use Kinesis Data Streams with Enhanced Fan-Out for high-throughput ingestion and dedicated consumer bandwidth.

Reference:

Kinesis Data Streams Enhanced Fan-Out

PutRecords API for Batch Writes

Question: 152

A data engineer needs to create a new empty table in Amazon Athena that has the same schema as an existing table named old-table.

Which SQL statement should the data engineer use to meet this requirement?

A.

CREATE TABLE new table AS SELECT •' FROM old zabio;

B.

INSERT INTO nestable SELECT T FROM

C.

CREATE TABLE newtable (LIKE oldtable);

D.

CREATE TABLE new_table AS (SELECT * FROM old_table) WITH MO DATA;

Answer: D

Explanation:

Problem Analysis:

The goal is to create a new empty table in Athena with the same schema as an existing table (old_table).

The solution must avoid copying any data.

Key Considerations:

CREATE TABLE AS (CTAS) is commonly used in Athena for creating new tables based on an existing table.

Adding the WITH NO DATA clause ensures only the schema is copied, without transferring any data.

Solution Analysis:

Option A: Copies both schema and data. Does not meet the requirement for an empty table.

Option B: Inserts data into an existing table, which does not create a new table.

Option C: Creates an empty table but does not copy the schema.

Option D: Creates a new table with the same schema and ensures it is empty by using WITH NO DATA.

Final Recommendation:

Use D. CREATE TABLE new_table AS (SELECT * FROM old_table) WITH NO DATA to create an empty table with the same schema.

Reference:

Athena CTAS Queries

CREATE TABLE Statement in Athena

Question: 153

A data engineer uses Amazon Kinesis Data Streams to ingest and process records that contain user behavior data from an application every day.

The data engineer notices that the data stream is experiencing throttling because hot shards receive much more data than other shards in the data stream.

How should the data engineer resolve the throttling issue?

A. Use a random partition key to distribute the ingested records.
B. Increase the number of shards in the data stream. Distribute the records across the shards.
C. Limit the number of records that are sent each second by the producer to match the capacity of the stream.
D. Decrease the size of the records that the producer sends to match the capacity of the stream.

Answer: C

Explanation:

Question: 154

A company uses AWS Glue jobs to implement several data pipelines. The pipelines are critical to the company. The company needs to implement a monitoring mechanism that will alert stakeholders if the pipelines fail. Which solution will meet these requirements with the LEAST operational overhead?

A. Create an Amazon EventBridge rule to match AWS Glue job failure events. Configure the rule to target an AWS Lambda function to process events. Configure the function to send notifications to an Amazon Simple Notification Service (Amazon SNS) topic.

B. Configure an Amazon CloudWatch Logs log group for the AWS Glue jobs. Create an Amazon EventBridge rule to match new log creation events in the log group. Configure the rule to target an AWS Lambda function that reads the logs and sends notifications to an Amazon Simple Notification Service (Amazon SNS) topic if AWS Glue job failure logs are present.

C. Create an Amazon EventBridge rule to match AWS Glue job failure events. Define an Amazon CloudWatch metric based on the EventBridge rule. Set up a CloudWatch alarm based on the metric to send notifications to an Amazon Simple Notification Service (Amazon SNS) topic.

D. Configure an Amazon CloudWatch Logs log group for the AWS Glue jobs. Create an Amazon EventBridge rule to match new log creation events in the log group. Configure the rule to send notifications to an Amazon Simple Notification Service (Amazon SNS) topic.

Answer: A

Explanation:

Creating an EventBridge rule that triggers a Lambda function on AWS Glue job failure events and then sends notifications via Amazon SNS is the most direct and operationally efficient method: “Practice Quiz 10: A data engineer must monitor the data pipeline... Which solution will meet these requirements?

A . Inspect the job run monitoring section of the AWS Glue console.

Correct Answer: A.”

Explanation:

- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf

Although this reference directly supports using AWS Glue's monitoring features via EventBridge, it implies that solutions like A—which directly use EventBridge failure events for automation—are more optimal and less complex than constructing custom logs and metrics.

Question: 155

An ecommerce company processes millions of orders each day. The company uses AWS Glue ETL to collect data from multiple sources, clean the data, and store the data in an Amazon S3 bucket in CSV format by using the S3 Standard storage class. The company uses the stored data to conduct daily analysis.

The company wants to optimize costs for data storage and retrieval.

Which solution will meet this requirement?

A. Transition the data to Amazon S3 Glacier Flexible Retrieval.

B. Transition the data from Amazon S3 to an Amazon Aurora cluster.

C. Configure AWS Glue ETL to transform the incoming data to Apache Parquet format.
D. Configure AWS Glue ETL to use Amazon EMR to process incoming data in parallel.

Answer: C

Explanation:

Apache Parquet is a columnar storage format that is much more space-efficient than row-based formats like CSV, especially for analytics workloads. Transforming data from CSV to Parquet significantly reduces storage costs and improves query performance. According to the study guide: "Parquet is a columnar storage file format that is optimized for use with analytics workloads, providing efficient storage and fast query performance."
- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf
By switching to Parquet, the company can reduce both storage size and retrieval times, making it the optimal choice for cost-effective data analysis.

Question: 156

A data engineer is launching an Amazon EMR cluster. The data that the data engineer needs to load into the new cluster is currently in an Amazon S3 bucket. The data engineer needs to ensure that data is encrypted both at rest and in transit.
The data that is in the S3 bucket is encrypted by an AWS Key Management Service (AWS KMS) key. The data engineer has an Amazon S3 path that has a Privacy Enhanced Mail (PEM) file.
Which solution will meet these requirements?

A. Create an Amazon EMR security configuration. Specify the appropriate AWS KMS key for at-rest encryption for the S3 bucket. Create a second security configuration. Specify the Amazon S3 path of the PEM file for in-transit encryption. Create the EMR cluster, and attach both security configurations to the cluster.
B. Create an Amazon EMR security configuration. Specify the appropriate AWS KMS key for local disk encryption for the S3 bucket. Specify the Amazon S3 path of the PEM file for in-transit encryption. Use the security configuration during EMR cluster creation.
C. Create an Amazon EMR security configuration. Specify the appropriate AWS KMS key for at-rest encryption for the S3 bucket. Specify the Amazon S3 path of the PEM file for in-transit encryption. Use the security configuration during EMR cluster creation.
D. Create an Amazon EMR security configuration. Specify the appropriate AWS KMS key for at-rest encryption for the S3 bucket. Specify the Amazon S3 path of the PEM file for in-transit encryption. Create the EMR cluster, and attach the security configuration to the cluster.

Answer: C

Explanation:

To meet both encryption at rest and in transit, a single Amazon EMR security configuration can be created specifying the AWS KMS key for encryption at rest and the PEM file for in-transit encryption. The study guide clearly states:

"AWS Key Management Service (KMS) provides encryption for data at rest, and SSL/TLS ensures encryption for data in transit, providing end-to-end encryption within an AWS environment."
- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf
A single security configuration is sufficient and the cleanest way to apply these security features during EMR cluster setup.

Question: 157

A company uses AWS Key Management Service (AWS KMS) to encrypt an Amazon Redshift cluster. The company wants to configure a cross-Region snapshot of the Redshift cluster as part of disaster recovery (DR) strategy. A data engineer needs to use the AWS CLI to create the cross-Region snapshot.
Which combination of steps will meet these requirements? (Select TWO.)

A. Create a KMS key and configure a snapshot copy grant in the source AWS Region.
B. In the source AWS Region, enable snapshot copying. Specify the name of the snapshot copy grant that is created in the destination AWS Region.
C. In the source AWS Region, enable snapshot copying. Specify the name of the snapshot copy grant that is created in the source AWS Region.
D. Create a KMS key and configure a snapshot copy grant in the destination AWS Region.
E. Convert the cluster to a Multi-AZ deployment.

Answer: C, D

Explanation:

To perform cross-Region snapshot copying of an encrypted Redshift cluster, AWS documentation and the exam study guide clearly outline two essential steps:
You must create a snapshot copy grant in the destination Region. This allows Amazon Redshift to encrypt the snapshots using the specified AWS KMS key.
You must enable snapshot copying in the source Region and specify the name of the snapshot copy grant that was created in the destination Region.
From the study guide:
"To enable cross-region copy of encrypted snapshots, you must create a snapshot copy grant in the destination Region and enable snapshot copying in the source Region by specifying the snapshot copy grant name."
- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf
Option E (Multi-AZ deployment) is not applicable to Amazon Redshift, which does not support MultiAZ configurations like Amazon RDS.

Question: 158

A company wants to ingest streaming data into an Amazon Redshift data warehouse from an Amazon Managed Streaming for Apache Kafka (Amazon MSK) cluster. A data engineer needs to develop a solution that provides low data access time and that optimizes storage costs.
Which solution will meet these requirements with the LEAST operational overhead?

A. Create an external schema that maps to the MSK cluster. Create a materialized view that references the external schema to consume the streaming data from the MSK topic.

B. Develop an AWS Glue streaming extract, transform, and load (ETL) job to process the incoming data from Amazon MSK. Load the data into Amazon S3. Use Amazon Redshift Spectrum to read the data from Amazon S3.

C. Create an external schema that maps to the streaming data source. Create a new Amazon Redshift table that references the external schema.

D. Create an Amazon S3 bucket. Ingest the data from Amazon MSK. Create an event-driven AWS Lambda function to load the data from the S3 bucket to a new Amazon Redshift table.

Answer: B

Explanation:

According to the guide:

"For integrating streaming data from Amazon MSK into Amazon Redshift efficiently and cost- effectively, AWS Glue streaming jobs can process and transform the data, storing it in Amazon S3.

Amazon Redshift Spectrum can then directly query the data from S3, minimizing operational overhead and reducing storage costs."

- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf

This setup offers:

Low latency via Glue streaming.

Low storage cost by using Parquet/ORC on S3.

Minimal operational overhead by avoiding complex pipelines or constantly updated materialized views.

Question: 159

A company stores sensitive data in an Amazon Redshift table. The company needs to give specific users the ability to access the sensitive dat

a. The company must not create duplication in the data.

Customer support users must be able to see the last four characters of the sensitive data. Audit users must be able to see the full value of the sensitive data. No other users can have the ability to access the sensitive information.

Which solution will meet these requirements?

A. Create a dynamic data masking policy to allow access based on each user role. Create IAM roles that have specific access permissions. Attach the masking policy to the column that contains sensitive data.

B. Enable metadata security on the Redshift cluster. Create IAM users and IAM roles for the customer support users and the audit users. Grant the IAM users and IAM roles permissions to view the metadata in the Redshift cluster.

C. Create a row-level security policy to allow access based on each user role. Create IAM roles that have specific access permissions. Attach the security policy to the table.

D. Create an AWS Glue job to redact the sensitive data and to load the data into a new Redshift table.

Answer: A

Explanation:

Amazon Redshift supports dynamic data masking, which enables you to limit sensitive data visibility to specific users and roles without duplicating the data. This approach supports showing only parts of a column's values (e.g.,

last four digits) and full visibility for authorized roles (e.g., auditors).

"With dynamic data masking, you can control how much sensitive data a user sees in query results without changing the data in the table."

- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf

IAM roles are used to associate users with the appropriate masking rules, keeping security tight and avoiding the creation of duplicate data views or tables.

Question: 160

A company uses Amazon DataZone as a data governance and business catalog solution. The company stores data in an Amazon S3 data lake. The company uses AWS Glue with an AWS Glue Data Catalog. A data engineer needs to publish AWS Glue Data Quality scores to the Amazon DataZone portal. Which solution will meet this requirement?

A. Create a data quality ruleset with Data Quality Definition Language (DQDL) rules that apply to a specific AWS Glue table. Schedule the ruleset to run daily. Configure the Amazon DataZone project to have an Amazon Redshift data source. Enable the data quality configuration for the data source.

B. Configure AWS Glue ETL jobs to use an Evaluate Data Quality transform. Define a data quality ruleset inside the jobs. Configure the Amazon DataZone project to have an AWS Glue data source. Enable the data quality configuration for the data source.

C. Create a data quality ruleset with Data Quality Definition Language (DQDL) rules that apply to a specific AWS Glue table. Schedule the ruleset to run daily. Configure the Amazon DataZone project to have an AWS Glue data source. Enable the data quality configuration for the data source.

D. Configure AWS Glue ETL jobs to use an Evaluate Data Quality transform. Define a data quality ruleset inside the jobs. Configure the Amazon DataZone project to have an Amazon Redshift data source. Enable the data quality configuration for the data source.

Answer: C

Explanation:

Publishing AWS Glue data quality scores to Amazon DataZone requires creating a DQDL ruleset, scheduling it to run regularly, and then linking the corresponding AWS Glue table as a data source in the DataZone project. The setup ensures that data quality scores from Glue are correctly published and accessible within Amazon DataZone:

"You can define DQDL rulesets for Glue tables and publish the data quality results to DataZone when the project is configured with an AWS Glue data source and the rulesets are scheduled."

- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf Option C follows the expected flow without unnecessary complexity and aligns perfectly with the integration flow supported by AWS.

Question: 161

A company runs multiple applications on AWS. The company configured each application to output logs. The company wants to query and visualize the application logs in near real time. Which solution will

meet these requirements?

A. Configure the applications to output logs to Amazon CloudWatch Logs log groups. Create an Amazon S3 bucket. Create an AWS Lambda function that runs on a schedule to export the required log groups to the S3 bucket. Use Amazon Athena to query the log data in the S3 bucket.

B. Create an Amazon OpenSearch Service domain. Configure the applications to output logs to Amazon CloudWatch Logs log groups. Create an OpenSearch Service subscription filter for each log group to stream the data to OpenSearch. Create the required queries and dashboards in OpenSearch Service to analyze and visualize the data.

C. Configure the applications to output logs to Amazon CloudWatch Logs log groups. Use CloudWatch log anomaly detection to query and visualize the log data.

D. Update the application code to send the log data to Amazon QuickSight by using Super-fast, Parallel, In-memory Calculation Engine (SPICE). Create the required analyses and dashboards in QuickSight.

Answer: B

Explanation:

The optimal solution for near-real-time querying and visualization of logs is to integrate Amazon CloudWatch Logs with Amazon OpenSearch Service using subscription filters, which stream the logs directly into OpenSearch for querying and dashboarding:

"Use OpenSearch Service with CloudWatch Logs and create a subscription filter to stream log data in near real time into OpenSearch. Then use OpenSearch dashboards for visualization."

- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf

This approach offers low latency and avoids batch exports, unlike the scheduled Athena + S3 pattern.

Question: 162

A company uses Amazon Redshift as a data warehouse solution. One of the datasets that the company stores in Amazon Redshift contains data for a vendor.

Recently, the vendor asked the company to transfer the vendor's data into the vendor's Amazon S3 bucket once each week.

Which solution will meet this requirement?

A. Create an AWS Lambda function to connect to the Redshift data warehouse. Configure the Lambda function to use the Redshift COPY command to copy the required data to the vendor's S3 bucket on a schedule.

B. Create an AWS Glue job to connect to the Redshift data warehouse. Configure the AWS Glue job to use the Redshift UNLOAD command to load the required data to the vendor's S3 bucket on a schedule.

C. Use the Amazon Redshift data sharing feature. Set the vendor's S3 bucket as the destination. Configure the source to be as a custom SQL query that selects the required data.

D. Configure Amazon Redshift Spectrum to use the vendor's S3 bucket as destination. Enable data querying in both directions.

Answer: B

Explanation:

The Redshift UNLOAD command is specifically designed to export query results to Amazon S3, and AWS Glue can orchestrate this as part of a scheduled job. This is the cleanest and most appropriate approach for recurring weekly data transfers:

"Use the Redshift UNLOAD command with AWS Glue to export data to Amazon S3. This pattern enables routine exports of selected data to external locations."

- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf

This avoids complexities of Redshift Spectrum or unsupported use of COPY commands in Lambda.

Question: 163

A data engineer has two datasets that contain sales information for multiple cities and states. One dataset is named reference, and the other dataset is named primary.

The data engineer needs a solution to determine whether a specific set of values in the city and state columns of the primary dataset exactly match the same specific values in the reference dataset. The data engineer wants to use Data Quality Definition Language (DQDL) rules in an AWS Glue Data Quality job.

Which rule will meet these requirements?

A. DatasetMatch "reference" "city->ref_city, state->ref_state" = 1.0
B. ReferentialIntegrity "city,state" "reference.{ref_city,ref_state}" = 1.0
C. DatasetMatch "reference" "city->ref_city, state->ref_state" = 100
D. ReferentialIntegrity "city,state" "reference.{ref_city,ref_state}" = 100

Answer: A

Explanation:

The DatasetMatch rule in DQDL checks for full value equivalence between mapped fields. A value of 1.0 indicates a 100% match. The correct syntax and metric for an exact match scenario are:

"Use DatasetMatch when comparing mapped fields between two datasets. The comparison score of 1.0 confirms a perfect match."

- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf

Options with "100" use incorrect syntax since DQDL uses floating-poi nt scores (e.g., 1.0, 0.95), not percentages.

Question: 164

A company has an Amazon Redshift data warehouse that users access by using a variety of IAM roles. More than 100 users access the data warehouse every day.

The company wants to control user access to the objects based on each user's job role, permissions, and how sensitive the data is.

Which solution will meet these requirements?

A. Use the role-based access control (RBAC) feature of Amazon Redshift.
B. Use the row-level security (RLS) feature of Amazon Redshift.
C. Use the column-level security (CLS) feature of Amazon Redshift.
D. Use dynamic data masking policies in Amazon Redshift.

Answer: A

Explanation:

Amazon Redshift supports Role-Based Access Control (RBAC) to manage access to database objects. RBAC allows administrators to create roles for job functions and assign privileges at the schema, table, or column level based on data sensitivity and user roles.
"RBAC in Amazon Redshift helps manage permissions more efficiently at scale by assigning users to roles that reflect their job function. It simplifies user management and secures access based on job role and data sensitivity."
- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf
RBAC is preferred over RLS or CLS alone because it offers a more comprehensive and scalable solution across multiple users and permissions.

Question: 165

A data engineer needs to create an empty copy of an existing table in Amazon Athena to perform data processing tasks. The existing table in Athena contains 1,000 rows.
Which query will meet this requirement?

A. CREATE TABLE new_table LIKE old_table;
B. CREATE TABLE new_table AS SELECT * FROM old_table WITH NO DATA;
C. CREATE TABLE new_table AS SELECT * FROM old_table;
D. CREATE TABLE new_table AS SELECT * FROM old_table WHERE 1=1;

Answer: B

Explanation:

In Amazon Athena, you can use CREATE TABLE AS SELECT with WITH NO DATA to create an empty copy of an existing table's schema:
"The query CREATE TABLE new_table AS SELECT * FROM old_table WITH NO DATA; creates a new table with the same schema but without copying over the data."
- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf
This is the most efficient way to create an empty version of the existing table.

Question: 166

A data engineer is building a data pipeline. A large data file is uploaded to an Amazon S3 bucket once each day at unpredictable times. An AWS Glue workflow uses hundreds of workers to process the file and load the data into

Amazon Redshift. The company wants to process the file as quickly as possible. Which solution will meet these requirements?

A. Create an on-demand AWS Glue trigger to start the workflow. Create an AWS Lambda function that runs every 15 minutes to check the S3 bucket for the daily file. Configure the function to start the AWS Glue workflow if the file is present.

B. Create an event-based AWS Glue trigger to start the workflow. Configure Amazon S3 to log events to AWS CloudTrail. Create a rule in Amazon EventBridge to forward PutObject events to the AWS Glue trigger.

C. Create a scheduled AWS Glue trigger to start the workflow. Create a cron job that runs the AWS Glue job every 15 minutes. Set up the AWS Glue job to check the S3 bucket for the daily file. Configure the job to stop if the file is not present.

D. Create an on-demand AWS Glue trigger to start the workflow. Create an AWS Database Migration Service (AWS DMS) migration task. Set the DMS source as the S3 bucket. Set the target endpoint as the AWS Glue workflow.

Answer: B

Explanation:

The best solution for fast, event-driven processing of unpredictable file uploads is to use S3 event notifications, CloudTrail, and EventBridge to automatically trigger the AWS Glue workflow:

"You can configure S3 PutObject events to be captured by CloudTrail and forwarded through EventBridge to trigger an AWS Glue job or workflow. This allows Glue to begin processing as soon as the file arrives, with minimal latency."

- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf

This option provides the lowest latency and least manual overhead compared to polling or scheduling solutions.

Question: 167

A company has an application that uses an Amazon API Gateway REST API and an AWS Lambda function to retrieve data from an Amazon DynamoDB instance. Users recently reported intermittent high latency in the application's response times. A data engineer finds that the Lambda function experiences frequent throttling when the company's other Lambda functions experience increased invocations.

The company wants to ensure the API's Lambda function operates without being affected by other Lambda functions.

Which solution will meet this requirement MOST cost-effectively?

A. Increase the number of read capacity unit (RCU) in DynamoDB.

B. Configure provisioned concurrency for the Lambda function.

C. Configure reserved concurrency for the Lambda function.

D. Increase the Lambda function timeout and allocated memory.

Answer: C

Explanation:

Reserved concurrency allows you to dedicate a portion of your account's available concurrency to a specific Lambda function. This ensures that the function always has sufficient capacity, regardless of how many other functions are running. It's more cost-effective than provisioned concurrency, which reserves warm environments at a higher cost.

"You can use reserved concurrency to guarantee that your critical Lambda functions have the capacity to run as needed without being throttled by other functions."

- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf

Provisioned concurrency would be overkill unless latency is also a concern for cold starts, which isn't stated in this case.

Question: 168

A company has a data processing pipeline that includes several dozen steps. The data processing pipeline needs to send alerts in real time when a step fails or succeeds. The data processing pipeline uses a combination of Amazon S3 buckets, AWS Lambda functions, and AWS Step Functions state machines.

A data engineer needs to create a solution to monitor the entire pipeline.

Which solution will meet these requirements?

A. Configure the Step Functions state machines to store notifications in an Amazon S3 bucket when the state machines finish running. Enable S3 event notifications on the S3 bucket.

B. Configure the AWS Lambda functions to store notifications in an Amazon S3 bucket when the state machines finish running. Enable S3 event notifications on the S3 bucket.

C. Use AWS CloudTrail to send a message to an Amazon Simple Notification Service (Amazon SNS) topic that sends notifications when a state machine fails to run or succeeds to run.

D. Configure an Amazon EventBridge rule to react when the execution status of a state machine changes. Configure the rule to send a message to an Amazon Simple Notification Service (Amazon SNS) topic that sends notifications.

Answer: D

Explanation:

AWS Step Functions natively emits state change events to Amazon EventBridge, which can trigger an Amazon SNS notification. This is the most direct and real-time way to alert on success/failure without relying on custom logging or polling.

"Step Functions automatically emits status changes that EventBridge can capture to trigger alerts or workflows. Use EventBridge to invoke an SNS topic for real-time alerts on job status."

- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf This provides real-time alerting and the least operational overhead.

Question: 169

A data engineer develops an AWS Glue Apache Spark ETL job to perform transformations on a dataset. When the data engineer runs the job, the job returns an error that reads, "No space left on device."

The data engineer needs to identify the source of the error and provide a solution.

Which combinations of steps will meet this requirement MOST cost-effectively? (Select TWO.)

A. Scale out the workers vertically to address data skewness.
B. Use the Spark UI and AWS Glue metrics to monitor data skew in the Spark executors.
C. Scale out the number of workers horizontally to address data skewness.
D. Enable the --write-shufle-files-to-s3 job parameter. Use the salting technique.
E. Use error logs in Amazon CloudWatch to monitor data skew.

Answer: B, D

Explanation:

A "No space left on device" error typically results from data skew or large shufle stages. The best actions are:
B. Monitor using Spark UI and Glue metrics to find skewed partitions or executor issues.
D. Use --write-shufle-files-to-s3 to ofload intermediate data to S3 instead of local disk, and apply salting to reduce skew.
"You can reduce the impact of data skew and large shufle operations by monitoring with Spark UI and enabling the --write-shufle-files-to-s3 option. Salting can help rebalance the skewed keys."
- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf Scaling out workers (A, C) is more costly and less efficient if the root cause (skew) is not fixed.

Question: 170

A company is setting up a data pipeline in AWS. The pipeline extracts client data from Amazon S3 buckets, performs quality checks, and transforms the dat
a. The pipeline stores the processed data in a relational database. The company will use the processed data for future queries.
Which solution will meet these requirements MOST cost-effectively?
A. Use AWS Glue ETL to extract the data from the S3 buckets and perform the transformations. Use AWS Glue Data Quality to enforce suggested quality rules. Load the data and the quality check results into an Amazon RDS for MySQL instance.
B. Use AWS Glue Studio to extract the data from the S3 buckets. Use AWS Glue DataBrew to perform the transformations and quality checks. Load the processed data into an Amazon RDS for MySQL instance. Load the quality check results into a new S3 bucket.
C. Use AWS Glue ETL to extract the data from the S3 buckets and perform the transformations. Use AWS Glue DataBrew to perform quality checks. Load the processed data and the quality check results into a new S3 bucket.
D. Use AWS Glue Studio to extract the data from the S3 buckets. Use AWS Glue DataBrew to perform the transformations and quality checks. Load the processed data and quality check results into an Amazon RDS for MySQL instance.

Answer: A

Explanation:

AWS Glue ETL is designed for scalable and serverless data processing, and it supports integrated quality enforcement using AWS Glue Data Quality, which makes it the most cost-effective and integrated option when

combined with Amazon RDS for MySQL as the relational database. “AWS Glue can perform data validation as part of the ETL process, ensuring data quality before storing the data in the target data store.”
- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf Using AWS Glue Data Quality directly in the ETL workflow is simpler and more cost-effective than separating transformation (Glue) and validation (DataBrew) into different services.

Question: 171

A sales company uses AWS Glue ETL to collect, process, and ingest data into an Amazon S3 bucket. The AWS Glue pipeline creates a new file in the S3 bucket every hour. File sizes vary from 200 KB to 300 KB. The company wants to build a sales prediction model by using data from the previous 5 years. The historic data includes 44,000 files.
The company builds a second AWS Glue ETL pipeline by using the smallest worker type. The second pipeline retrieves the historic files from the S3 bucket and processes the files for downstream analysis. The company notices significant performance issues with the second ETL pipeline.
The company needs to improve the performance of the second pipeline.
Which solution will meet this requirement MOST cost-effectively?

A. Use a larger worker type.
B. Increase the number of workers in the AWS Glue ETL jobs.
C. Use the AWS Glue DynamicFrame grouping option.
D. Enable AWS Glue auto scaling.

Answer: D

Explanation:

When processing many small files, the overhead of managing partitions can degrade performance. Rather than scaling workers manually, enabling AWS Glue auto scaling dynamically adjusts worker count based on resource requirements, which is cost-effective and addresses the performance concern.
“AWS Glue auto scaling automatically adds or removes workers based on the workload, which is efficient for handling performance bottlenecks when working with many small files.”
- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf
Manually scaling up workers or increasing their size (A, B) can be more expensive and less adaptive.

Question: 172

Files from multiple data sources arrive in an Amazon S3 bucket on a regular basis. A data engineer wants to ingest new files into Amazon Redshift in near real time when the new files arrive in the S3 bucket.
Which solution will meet these requirements?

A. Use the query editor v2 to schedule a COPY command to load new files into Amazon Redshift.
B. Use the zero-ETL integration between Amazon Aurora and Amazon Redshift to load new files into Amazon Redshift.

C. Use AWS Glue job bookmarks to extract, transform, and load (ETL) load new files into Amazon Redshift.

D. Use S3 Event Notifications to invoke an AWS Lambda function that loads new files into Amazon Redshift.

Answer: D

Explanation:

For near real-time processing of new files in S3, event-driven ingestion is optimal. S3 Event Notifications can trigger AWS Lambda to immediately load data into Amazon Redshift, eliminating latency associated with batch scheduling.

"Event-based triggers using S3 notifications and Lambda functions are effective for near real-time ingestion pipelines into Amazon Redshift."

- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf

Option C (Glue bookmarks) is best for batch jobs, and zero-ETL applies to Aurora to Redshift, not S3.

Question: 173

A company is designing a serverless data processing workflow in AWS Step Functions that involves multiple steps. The processing workflow ingests data from an external API, transforms the data by using multiple AWS Lambda functions, and loads the transformed data into Amazon DynamoDB. The company needs the workflow to perform specific steps based on the content of the incoming data.

Which Step Functions state type should the company use to meet this requirement?

A. Parallel

B. Choice

C. Task

D. Map

Answer: B

Explanation:

The Choice state type in AWS Step Functions is designed to perform branching logic, i.e., routing execution to different paths based on conditions in the input data.

"The Step Functions Choice state lets you branch the execution flow depending on values in the state's input. This allows you to run different processing logic based on dynamic conditions like values in the input JSON."

- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf This makes Choice the correct answer for content-driven conditional workflows.

Question: 174

A company is using Amazon S3 to build a data lake. The company needs to replicate records from multiple source databases into Apache Parquet format.

Most of the source databases are hosted on Amazon RDS. However, one source database is an onpremises

Microsoft SQL Server Enterprise instance. The company needs to implement a solution to replicate existing data from all source databases and all future changes to the target S3 data lake. Which solution will meet these requirements MOST cost-effectively?

A. Use one AWS Glue job to replicate existing data. Use a second AWS Glue job to replicate future changes.
B. Use AWS Database Migration Service (AWS DMS) to replicate existing data. Use AWS Glue jobs to replicate future changes.
C. Use AWS Database Migration Service (AWS DMS) to replicate existing data and future changes.
D. Use AWS Glue jobs to replicate existing data. Use Amazon Kinesis Data Streams to replicate future changes.

Answer: C

Explanation:

AWS Database Migration Service (AWS DMS) is purpose-built to migrate and continuously replicate data from both AWS-hosted and on-premises databases. It supports full-load (existing data) and change data capture (CDC) for ongoing changes, making it the most cost-effective and operationally simple solution in this scenario.
"DMS supports both full-load and continuous replication via CDC. This enables replicating existing and future data from various sources to a data lake in Amazon S3."
- Ace the AWS Certified Data Engineer - Associate Certification - version 2 - apple.pdf
AWS Glue is not suitable for real-time CDC replication across hybrid environments.

Question: 175 A company has as JSON file that contains personally identifiable information (PIT) data and non-PII dat

a. The company needs to make the data available for querying and analysis. The non-PII data must be available to everyone in the company. The PII data must be available only to a limited group of employees. Which solution will meet these requirements with the LEAST operational overhead?

A. Store the JSON file in an Amazon S3 bucket. Configure AWS Glue to split the file into one file that contains the PII data and one file that contains the non-PII data. Store the output files in separate S3 buckets. Grant the required access to the buckets based on the type of user.
B. Store the JSON file in an Amazon S3 bucket. Use Amazon Macie to identify PII data and to grant access based on the type of user.
C. Store the JSON file in an Amazon S3 bucket. Catalog the file schema in AWS Lake Formation. Use Lake Formation permissions to provide access to the required data based on the type of user.
D. Create two Amazon RDS PostgreSQL databases. Load the PII data and the non-PII data into the separate databases. Grant access to the databases based on the type of user.

Answer: C

Question: 176

A retail company stores order information in an Amazon Aurora table named Orders. The company needs to create operational reports from the Orders table with minimal latency. The Orders table contains billions of rows, and over 100,000 transactions can occur each second.

A marketing team needs to join the Orders data with an Amazon Redshift table named Campaigns in the marketing team's data warehouse. The operational Aurora database must not be affected. Which solution will meet these requirements with the LEAST operational effort?

A. Use AW5 Database Migration Service (AWS DMS) Serverless to replicate the Orders table to Amazon Redshift. Create a materialized view in Amazon Redshift to join with the Campaigns table.
B. Use the Aurora zero-ETL integration with Amazon Redshift to replicate the Orders table. Create a materialized view in Amazon Redshift to join with the Campaigns table.
C. Use AWS Glue to replicate the Orders table to Amazon Redshift. Create a materialized view in Amazon Redshift to join with the Campaigns table.
D. Use federated queries to query the Orders table directly from Aurora. Create a materialized view in Amazon Redshift to join with the Campaigns table.

Answer: C

Question: 177

A data engineer is optimizing query performance in Amazon Athena notebooks that use Apache Spark to analyze large datasets that are stored in Amazon S3. The data is partitioned. An AWS Glue crawler updates the partitions.
The data engineer wants to minimize the amount of data that is scanned to improve efficiency of Athena queries.
Which solution will meet these requirements?

A. Apply partition filters in the queries.
B. Increase the frequency of AWS Glue crawler invocations to update the data catalog more often.
C. Organize the data that is in Amazon S3 by using a nested directory structure.
D. Configure Spark to use in-memory caching for frequently accessed data.

Answer: A

Question: 178

A company manages an Amazon Redshift data warehouse. The data warehouse is in a public subnet inside a custom VPC A security group allows only traffic from within itself- An ACL is open to all traffic.
The company wants to generate several visualizations in Amazon QuickSight for an upcoming sales event. The company will run QuickSight Enterprise edition in a second AW5 account inside a public subnet within a second custom VPC. The new public subnet has a security group that allows outbound traffic to the existing Redshift cluster.
A data engineer needs to establish connections between Amazon Redshift and QuickSight. QuickSight must refresh dashboards by querying the Redshift cluster.
Which solution will meet these requirements?

A. Configure the Redshift security group to allow inbound traffic on the Redshift port from the QuickSight security group.

B. Assign Elastic IP addresses to the QuickSight visualizations. Configure the QuickSight security group to allow inbound traffic on the Redshift port from the Elastic IP addresses.

C. Confirm that the CIDR ranges of the Redshift VPC and the QuickSight VPC are the same. If CIDR ranges are different, reconfigure one CIDR range to match the other. Establish network peering between the VPCs.

D. Create a QuickSight gateway endpoint in the Redshift VPC. Attach an endpoint policy to the gateway endpoint to ensure only specific QuickSight accounts can use the endpoint.

Answer: A

Question: 179

A data engineer needs to run a data transformation job whenever a user adds a file to an Amazon S3 bucket. The job will run for less than 1 minute. The job must send the output through an email message to the data engineer. The data engineer expects users to add one file every hour of the day. Which solution will meet these requirements in the MOST operationally efficient way?

A. Create a small Amazon EC2 instance that polls the S3 bucket for new files. Run transformation code on a schedule to generate the output. Use operating system commands to send email messages.

B. Run an Amazon Elastic Container Service (Amazon ECS) task to poll the S3 bucket for new files. Run transformation code on a schedule to generate the output. Use operating system commands to send email messages.

C. Create an AWS Lambda function to transform the data. Use Amazon S3 Event Notifications to invoke the Lambda function when a new object is created. Publish the output to an Amazon Simple Notification Service (Amazon SNS) topic. Subscribe the data engineer's email account to the topic.

D. Deploy an Amazon EMR cluster. Use EMR File System (EMRFS) to access the files in the S3 bucket. Run transformation code on a schedule to generate the output to a second S3 bucket. Create an Amazon Simple Notification Service (Amazon SNS) topic. Configure Amazon S3 Event Notifications to notify the topic when a new object is created.

Answer: C

Question: 180

A data engineer is troubleshooting an AWS Glue workflow that occasionally fails. The engineer determines that the failures are a result of data quality issues. A business reporting team needs to receive an email notification any time the workflow fails in the future.

Which solution will meet this requirement?

A. Create an Amazon Simple Notification Service (Amazon SNS) FIFO topic. Subscribe the team's email account to

the SNS topic. Create an AWS Lambda function that initiates when the AWS Glue job state changes to FAILED. Set the SNS topic as the target.

B. Create an Amazon Simple Notification Service (Amazon SNS) standard topic. Subscribe the team's email account to the SNS topic. Create an Amazon EventBridge rule that triggers when the AWS Glue Job state changes to FAILED. Set the SNS topic as the target.

C. Create an Amazon Simple Queue Service (Amazon SQS) FIFO queue. Subscribe the team's email account to the SQS queue. Create an AWS Config rule that triggers when the AWS Glue job state changes to FAILED. Set the SQS queue as the target.

D. Create an Amazon Simple Queue Service (Amazon SQS) standard queue. Subscribe the team's email account to the SQS queue. Create an Amazon EventBridge rule that triggers when the AWS Glue job state changes to FAILED. Set the SQS queue as the target.

Answer: B

Question: 181

A company uses Amazon Redshift as its data warehouse service. A data engineer needs to design a physical data model.

The data engineer encounters a de-normalized table that is growing in size. The table does not have a suitable column to use as the distribution key.

Which distribution style should the data engineer use to meet these requirements with the LEAST maintenance overhead?

A. ALL distribution

B. EVEN distribution

C. AUTO distribution

D. KEY distribution

Answer: B

Question: 182

A company receives marketing campaign data from a vendor. The company ingests the data into an Amazon S3 bucket every 40 to 60 minutes. The data is in CSV format. File sizes are between 100 KB and 300 KB.

A data engineer needs to set-up an extract, transform, and load (ETL) pipeline to upload the content of each file to Amazon Redshift.

Which solution will meet these requirements with the LEAST operational overhead?

A. Create an AWS Lambda function that connects to Amazon Redshift and runs a COPY command. Use Amazon EventBridge to invoke the Lambda function based on an Amazon S3 upload trigger.

B. Create an Amazon Data Firehose stream. Configure the stream to use an AWS Lambda function as a source to pull data from the S3 bucket. Set Amazon Redshift as the destination.

C. Use Amazon Redshift Spectrum to query the S3 bucket. Configure an AWS Glue Crawler for the S3 bucket to update metadata in an AWS Glue Data Catalog.

D. Creates an AWS Database Migration Service (AWS DMS) task. Specify an appropriate data schema to migrate. Specify the appropriate type of migration to use.

Answer: D

Question: 183

A company has a data pipeline that uses an Amazon RDS instance, AWS Glue jobs, and an Amazon S3 bucket. The RDS instance and AWS Glue jobs run in a private subnet of a VPC and in the same security group.

A use' made a change to the security group that prevents the AWS Glue jobs from connecting to the RDS instance. After the change, the security group contains a single rule that allows inbound SSH traffic from a specific IP address.

The company must resolve the connectivity issue.

Which solution will meet this requirement?

A. Add an inbound rule that allows all TCP traffic on all TCP ports. Set the security group as the source.

B. Add an inbound rule that allows all TCP traffic on all UDP ports. Set the private IP address of the RDS instance as the source.

C. Add an inbound rule that allows all TCP traffic on all TCP ports. Set the DNS name of the RDS instance as the source.

D. Replace the source of the existing SSH rule with the private IP address of the RDS instance. Create an outbound rule with the same source, destination, and protocol as the inbound SSH rule.

Answer: A

Question: 184

A company wants to migrate a data warehouse from Teradata to Amazon Redshift. Which solution will meet this requirement with the LEAST operational effort?

A. Use AWS Database Migration Service (AWS DMS) Schema Conversion to migrate the schema. Use AWS DMS to migrate the data.

B. Use the AWS Schema Conversion Tool (AWS SCT) to migrate the schema. Use AWS Database Migration Service (AWS DMS) to migrate the data.

C. Use AWS Database Migration Service (AWS DMS) to migrate the data. Use automatic schema conversion.

D. Manually export the schema definition from Teradata. Apply the schema to the Amazon Redshift database. Use AWS Database Migration Service (AWS DMS) to migrate the data.

Answer: B

Question: 185

A company wants to combine data from multiple software as a service (SaaS) applications for analysis. A data engineering team needs to use Amazon QuickSight to perform the analysis and build dashboards. A data engineer needs to extract the data from the SaaS applications and make the data available for QuickSight queries.
Which solution will meet these requirements in the MOST operationally efficient way?

A. Create AWS Lambda functions that call the required APIs to extract the data from the applications. Store the data in an Amazon S3 bucket. Use AWS Glue to catalog the data in the S3 bucket. Create a data source and a dataset in QuickSight
B. Use AWS Lambda functions as Amazon Athena data source connectors to run federated queries against the SaaS applications. Create an Athena data source and a dataset in QuickSight.
C. Use Amazon AppFlow to create a Row for each SaaS application. Set an Amazon S3 bucket as the destination. Schedule the flows to extract the data to the bucket. Use AWS Glue to catalog the data in the S3 bucket. Create a data source and a dataset in QuickSight.
D. Export data the from the SaaS applications as Microsoft Excel files. Create a data source and a dataset in QuickSight by uploading the Excel files.

Answer: C

Question: 186

A company uses AWS Glue Apache Spark jobs to handle extract, transform, and load (ETL) workloads. The company has enabled logging and monitoring for all AWS Glue jobs. One of the AWS Glue jobs begins to fail. A data engineer investigates the error and wants to examine metrics for all individual stages within the job. How can the data engineer access the stage metrics?

A. Examine the AWS Glue job and stage details in the Spark UI.
B. Examine the AWS Glue job and stage metrics in Amazon CloudWatch.
C. Examine the AWS Glue job and stage logs in AWS CloudTrail logs.
D. Examine the AWS Glue job and stage details by using the run insights feature on the job.

Answer: A

Question: 187

A data engineer notices slow query performance on a highly partitioned table that is in Amazon Athen
a. The table contains daily data for the previous 5 years, partitioned by date. The data engineer wants to improve query performance and to automate partition management. Which solution will meet these requirements?

A. Use an AWS Lambda function that runs daily. Configure the function to manually create new partitions in AW5 Glue for each day's data.

B. Use partition projection in Athena. Configure the table properties by using a date range from 5 years ago to the present.

C. Reduce the number of partitions by changing the partitioning schema from dairy to monthly granularity.

D. Increase the processing capacity of Athena queries by allocating more compute resources.

Answer: B

Question: 188

A company uses an Amazon Redshift cluster as a data warehouse that is shared across two departments. To comply with a security policy, each department must have unique access permissions.

Department A must have access to tables and views for Department A. Department B must have access to tables and views for Department B.

The company often runs SQL queries that use objects from both departments in one query.

Which solution will meet these requirements with the LEAST operational overhead?

A. Group tables and views for each department into dedicated schemas. Manage permissions at the schema level.

B. Group tables and views for each department into dedicated databases. Manage permissions at the database level.

C. Update the names of the tables and views to follow a naming convention that contains the department names. Manage permissions based on the new naming convention.

D. Create an IAM user group for each department. Use identity-based IAM policies to grant table and view permissions based on the IAM user group.

Answer: A

Question: 189

A data engineer is using an Apache Iceberg framework to build a data lake that contains 100 TB of data. The data engineer wants to run AWS Glue Apache Spark Jobs that use the Iceberg framework.

What combination of steps will meet these requirements? (Select TWO.)

A. Create a key named -conf for an AWS Glue job. Set Iceberg as a value for the --datalake-formats job parameter.

B. Specify the path to a specific version of Iceberg by using the --extra-Jars job parameter. Set Iceberg as a value for the ~ datalake-formats job parameter.

C. Set Iceberg as a value for the -datalake-formats job parameter.

D. Set the -enable-auto-scaling parameter to true.

E. Add the -job-bookmark-option: job-bookmark-enable parameter to an AWS Glue job.

Answer: A, E

Question: 190

A data engineer is configuring an AWS Glue Apache Spark extract, transform, and load (ETL) Job. The job contains a sort-merge join of two large and equally sized DataFrames.

The job is failing with the following error: No space left on device.

Which solution will resolve the error?

A. Use the AWS Glue Spark shufle manager.

B. Deploy an Amazon Elastic Block Store (Amazon EBS) volume for the job to use.

C. Convert the sort-merge join in the job to be a broadcast join.

D. Convert the DataFrames to DynamicFrames, and perform a DynamicFrame join in the job.

Answer: D