ATMIC NETWORKS
YOUR GUIDE TO GLOBAL CERTIFIATION

"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."

www.atmicnetworks .com

Warning: Keep connected with our support team for latest updates

Question: 1

What built-in Snowflake features make use of the change tracking metadata for a table? (Choose two.)

A. The MERGE command
B. The UPSERT command
C. The CHANGES clause
D. A STREAM object
E. The CHANGE_DATA_CAPTURE command

Answer: C, D

Explanation:

The built-in Snowflake features that make use of the change tracking metadata for a table are the CHANGES clause and a STREAM object. The CHANGES clause enables querying the change tracking metadata for a table or view within a specified interval of time without having to create a stream with an explicit transactional offset1. A STREAM object records data manipulation language (DML) changes made to tables, including inserts, updates, and deletes, as well as metadata about each change, so that actions can be taken using the changed data. This process is referred to as change data capture (CDC)2. The other options are incorrect because they do not make use of the change tracking metadata for a table. The MERGE command performs insert, update, or delete operations on a target table based on the results of a join with a source table3. The UPSERT command is not a valid Snowflake command. The CHANGE_DATA_CAPTURE command is not a valid Snowflake command. Reference: CHANGES | Snowflake Documentation, Change Tracking Using Table Streams | Snowflake Documentation, MERGE | Snowflake Documentation

Question: 2

When using the Snowflake Connector for Kafka, what data formats are supported for the messages? (Choose two.)

A. CSV
B. XML
C. Avro
D. JSON
E. Parquet

Answer: C, D

Explanation:

The data formats that are supported for the messages when using the Snowflake Connector for Kafka are Avro and JSON. These are the two formats that the connector can parse and convert into Snowflake table rows. The connector supports both schemaless and schematized JSON, as well as Avro with or without a schema registry1. The other options are incorrect because they are not supported data formats for the messages. CSV, XML, and Parquet are not formats that the connector can parse and convert into Snowflake table rows. If the messages are in these formats, the connector will load them as VARIANT data type and store them as raw strings in the table2. Reference: Snowflake Connector for Kafka | Snowflake Documentation, Loading Protobuf Data using the Snowflake Connector for Kafka | Snowflake Documentation

Question: 3

At which object type level can the APPLY MASKING POLICY, APPLY ROW ACCESS POLICY and APPLY SESSION POLICY privileges be granted?

A. Global
B. Database
C. Schema
D. Table

Answer: A

Explanation:

The object type level at which the APPLY MASKING POLICY, APPLY ROW ACCESS POLICY and APPLY SESSION POLICY privileges can be granted is global. These are account-level privileges that control who can apply or unset these policies on objects such as columns, tables, views, accounts, or users. These privileges are granted to the ACCOUNTADMIN role by default, and can be granted to other roles as needed. The other options are incorrect because they are not the object type level at which these privileges can be granted. Database, schema, and table are lower-level object types that do not support these privileges. Reference: Access Control Privileges | Snowflake Documentation, Using Dynamic Data Masking | Snowflake Documentation, Using Row Access Policies | Snowflake Documentation, Using Session Policies | Snowflake Documentation

Question: 4

An Architect uses COPY INTO with the ON_ERROR=SKIP_FILE option to bulk load CSV files into a table called TABLEA, using its table stage. One file named file5.csv fails to load. The Architect fixes the file and re-loads it to the stage with the exact same file name it had previously.
Which commands should the Architect use to load only file5.csv file from the stage? (Choose two.)

A. COPY INTO tablea FROM @%tablea RETURN_FAILED_ONLY = TRUE;
B. COPY INTO tablea FROM @%tablea;

C. COPY INTO tablea FROM @%tablea FILES = ('file5.csv');
D. COPY INTO tablea FROM @%tablea FORCE = TRUE;
E. COPY INTO tablea FROM @%tablea NEW_FILES_ONLY = TRUE;
F. COPY INTO tablea FROM @%tablea MERGE = TRUE;

Answer: BC

Explanation:

Option A (RETURN_FAILED_ONLY) will only load files that previously failed to load. Since file5.csv already exists in the stage with the same name, it will not be considered a new file and will not be loaded.
Option D (FORCE) will overwrite any existing data in the table. This is not desired as we only want to load the data from file5.csv.
Option E (NEW_FILES_ONLY) will only load files that have been added to the stage since the last COPY command. This will not work because file5.csv was already in the stage before it was fixed. Option F (MERGE) is used to merge data from a stage into an existing table, creating new rows for any data not already present. This is not needed in this case as we simply

want to load the data from file5.csv.

Therefore, the architect can use either COPY INTO tablea FROM @%tablea or COPY INTO tablea FROM @%tablea FILES = ('file5.csv') to load only file5.csv from the stage. Both options will load the data from the specified file without overwriting any existing data or requiring additional configuration

Question: 5

A large manufacturing company runs a dozen individual Snowflake accounts across its business divisions. The company wants to increase the level of data sharing to support supply chain optimizations and increase its purchasing leverage with multiple vendors.

The company's Snowflake Architects need to design a solution that would allow the business divisions to decide what to share, while minimizing the level of effort spent on configuration and management. Most of the company divisions use Snowflake accounts in the same cloud deployments with a few exceptions for European-based divisions.

According to Snowflake recommended best practice, how should these requirements be met?

A. Migrate the European accounts in the global region and manage shares in a connected graph architecture. Deploy a Data Exchange.

B. Deploy a Private Data Exchange in combination with data shares for the European accounts.

C. Deploy to the Snowflake Marketplace making sure that invoker_share() is used in all secure views. D. Deploy a Private Data Exchange and use replication to allow European data shares in the Exchange.

Answer: B

Explanation:

According to Snowflake recommended best practice, the requirements of the large manufacturing company should be met by deploying a Private Data Exchange in combination with data shares for the European accounts. A Private Data Exchange is a feature of the Snowflake Data Cloud platform

that enables secure and governed sharing of data between organizations. It allows Snowflake customers to create their own data hub and invite other parts of their organization or external partners to access and contribute data sets. A Private Data Exchange provides centralized management, granular access control, and data usage metrics for the data shared in the exchange1. A data share is a secure and direct way of sharing data between Snowflake accounts without having to copy or move the data. A data share allows the data provider to grant privileges on selected objects in their account to one or more data consumers in other accounts2. By using a Private Data Exchange in combination with data shares, the company can achieve the following benefits: The business divisions can decide what data to share and publish it to the Private Data Exchange, where it can be discovered and accessed by other members of the exchange. This reduces the effort and complexity of managing multiple data sharing relationships and configurations.

The company can leverage the existing Snowflake accounts in the same cloud deployments to create the Private Data Exchange and invite the members to join. This minimizes the migration and setup costs and leverages the existing Snowflake features and security.

The company can use data shares to share data with the European accounts that are in different regions or cloud platforms. This allows the company to comply with the regional and regulatory requirements for data sovereignty and privacy, while still enabling data collaboration across the organization.

The company can use the Snowflake Data Cloud platform to perform data analysis and transformation on the shared data, as well as integrate with other data sources and applications. This enables the company to optimize its supply chain and increase its purchasing leverage with multiple vendors.

The other options are incorrect because they do not meet the requirements or follow the best practices. Option A is incorrect because migrating the European accounts to the global region may violate the data sovereignty and privacy regulations, and deploying a Data Exchange may not provide the level of control and management that the company

needs. Option C is incorrect because deploying to the Snowflake Marketplace may expose the company's data to unwanted consumers, and using invoker_share() in secure views may not provide the desired level of security and governance. Option D is incorrect because using replication to allow European data shares in the Exchange may incur additional costs and complexity, and may not be necessary if data shares can be used instead. Reference: Private Data Exchange | Snowflake Documentation, Introduction to Secure Data Sharing | Snowflake Documentation

Question: 6

A user has the appropriate privilege to see unmasked data in a column.

If the user loads this column data into another column that does not have a masking policy, what will occur?

A. Unmasked data will be loaded in the new column.
B. Masked data will be loaded into the new column.
C. Unmasked data will be loaded into the new column but only users with the appropriate privileges will be able to see the unmasked data.
D. Unmasked data will be loaded into the new column and no users will be able to see the unmasked data.

Answer: A

Explanation:

According to the SnowPro Advanced: Architect documents and learning resources, column masking policies are applied at query time based on the privileges of the user who runs the query. Therefore, if a user has the privilege to see unmasked data in a column, they will see the original data when they query that column. If they load this column data into another column that does not have a masking policy, the unmasked data will be loaded in the new column, and any user who can query the new column will see the unmasked data as well. The masking policy does not affect the underlying data in the column, only the query results.

Reference:

Snowflake Documentation: Column Masking

Snowflake Learning: Column Masking

Question: 7

How can an Architect enable optimal clustering to enhance performance for different access paths on a given table?

A. Create multiple clustering keys for a table.
B. Create multiple materialized views with different cluster keys.
C. Create super projections that will automatically create clustering.
D. Create a clustering key that contains all columns used in the access paths.

Answer: B

Explanation:

According to the SnowPro Advanced: Architect documents and learning resources, the best way to enable optimal clustering to enhance performance for different access paths on a given table is to create multiple materialized views with different cluster keys. A materialized view is a pre-computed result set that is derived from a query on one or more base tables. A materialized view can be clustered by specifying a clustering key, which is a subset of columns or expressions that determines how the data in the materialized view is co-located in micro-partitions. By creating multiple materialized views with different cluster keys, an Architect can optimize the performance of queries that use different access paths on the same base table. For example, if a base table has columns A, B, C, and D, and there are queries that filter on A and B, or on

C and D, or on A and C, the Architect can create three materialized views, each with a different cluster key: (A, B), (C, D), and (A, C). This way, each query can leverage the optimal clustering of the corresponding materialized view and achieve faster scan efficiency and better compression.

Reference:

Snowflake Documentation: Materialized Views

Snowflake Learning: Materialized Views

https://www.snowflake.com/blog/using-materialized-views-to-solve-multi-clustering-performance- problems/

Question: 8

Company A would like to share data in Snowflake with Company B. Company B is not on the same cloud platform as Company A.

What is required to allow data sharing between these two companies?

A. Create a pipeline to write shared data to a cloud storage location in the target cloud provider.

B. Ensure that all views are persisted, as views cannot be shared across cloud platforms.

C. Setup data replication to the region and cloud platform where the consumer resides.

D. Company A and Company B must agree to use a single cloud platform: Data sharing is only possible if the companies share the same cloud provider.

Answer: C

Explanation:

According to the SnowPro Advanced: Architect documents and learning resources, the requirement to allow data sharing between two companies that are not on the same cloud platform is to set up data replication to the region and cloud platform where the consumer resides. Data replication is a feature of Snowflake that enables copying databases across accounts in different regions and cloud platforms. Data replication allows data providers to securely share data with data consumers across different regions and cloud platforms by creating a replica database in the consumer's account. The replica database is read-only and automatically synchronized with the primary database in the provider's account. Data replication is useful for scenarios where data sharing is not possible or desirable due to latency, compliance, or security reasons1. The other options are incorrect because they are not required or feasible to allow data sharing between two companies that are not on the same cloud platform. Option A is incorrect because creating a pipeline to write shared data to a cloud storage location in the target cloud provider is not a secure or efficient way of sharing data. It would require additional steps to load the data from the cloud storage to the consumer's account, and it would not leverage the benefits of Snowflake's data sharing features. Option B is incorrect because ensuring that all views are persisted is not relevant for data sharing across cloud platforms. Views can be shared across cloud platforms as long as they reference objects in the same database. Persisting views is an option to improve the performance of querying views, but it is not required for data sharing2. Option D is incorrect because Company A and Company B do not need to agree to use a single cloud platform. Data sharing is possible across different cloud platforms using data replication or other methods, such as listings or auto-fulfillment3. Reference: Replicating Databases Across Multiple Accounts | Snowflake Documentation, Persisting Views | Snowflake Documentation, Sharing Data Across Regions and Cloud Platforms | Snowflake Documentation

Question: 9

What are some of the characteristics of result set caches? (Choose three.)

A. Time Travel queries can be executed against the result set cache.

B. Snowflake persists the data results for 24 hours.
C. Each time persisted results for a query are used, a 24-hour retention period is reset.
D. The data stored in the result cache will contribute to storage costs.
E. The retention period can be reset for a maximum of 31 days.
F. The result set cache is not shared between warehouses.

Answer: B, C, E

Explanation:
Comprehensive and Detailed Explanation: According to the SnowPro Advanced: Architect documents and learning resources, some of the characteristics of result set caches are:
Snowflake persists the data results for 24 hours. This means that the result set cache holds the results of every query executed in the past 24 hours, and can be reused if the same query is submitted again and the underlying data has not changed1.
Each time persisted results for a query are used, a 24-hour retention period is reset. This means that the result set cache extends the lifetime of the results every time they are reused, up to a maximum of 31 days from the date and time that the query was first executed1.
The retention period can be reset for a maximum of 31 days. This means that the result set cache will purge the results after 31 days, regardless of whether they are reused or not. After 31 days, the next time the query is submitted, a new result is generated and persisted1.
The other options are incorrect because they are not characteristics of result set caches. Option A is incorrect because Time Travel queries cannot be executed against the result set cache. Time Travel queries use the AS OF clause to access historical data that is stored in the storage layer, not the result set cache2. Option D is incorrect because the data stored in the result set cache does not contribute to storage costs. The result set cache is maintained by the service layer, and does not incur any additional charges1. Option F is incorrect because the result set cache is shared between warehouses. The result set cache is available across virtual warehouses, so query results returned to one user are available to any other user on the system who executes the same query, provided the underlying data has not changed1. Reference: Using Persisted Query Results | Snowflake Documentation, Time Travel | Snowflake Documentation

Question: 10

Which organization-related tasks can be performed by the ORGADMIN role? (Choose three.)

A. Changing the name of the organization
B. Creating an account
C. Viewing a list of organization accounts
D. Changing the name of an account
E. Deleting an account
F. Enabling the replication of a database

Answer: B, C, F

Explanation:

According to the SnowPro Advanced: Architect documents and learning resources, the organization- related tasks that can be performed by the ORGADMIN role are:
Creating an account in the organization. A user with the ORGADMIN role can use the CREATE ACCOUNT command to create a new account that belongs to the same organization as the current account1.
Viewing a list of organization accounts. A user with the ORGADMIN role can use the SHOW ORGANIZATION ACCOUNTS command to view the names and properties of all accounts in the organization2. Alternatively, the user can use the Admin » Accounts page in the web interface to view the organization name and account names3.
Enabling the replication of a database. A user with the ORGADMIN role can use the

SYSTEM$GLOBAL_ACCOUNT_SET_PARAMETER function to enable database replication for an account in the organization. This allows the user to replicate databases across accounts in different regions and cloud platforms for data availability and durability4.

The other options are incorrect because they are not organization-related tasks that can be performed by the ORGADMIN role. Option A is incorrect because changing the name of the organization is not a task that can be performed by the ORGADMIN role. To change the name of an organization, the user must contact Snowflake Support3. Option D is incorrect because changing the name of an account is not a task that can be performed by the ORGADMIN role. To change the name of an account, the user must contact Snowflake Support5. Option E is incorrect because deleting an account is not a task that can be performed by the ORGADMIN role. To delete an account, the user must contact Snowflake Support. Reference: CREATE ACCOUNT | Snowflake Documentation, SHOW ORGANIZATION ACCOUNTS | Snowflake Documentation, Getting Started with Organizations | Snowflake Documentation, SYSTEM$GLOBAL_ACCOUNT_SET_PARAMETER | Snowflake Documentation, ALTER ACCOUNT | Snowflake Documentation, [DROP ACCOUNT | Snowflake Documentation]

Question: 11

A Data Engineer is designing a near real-time ingestion pipeline for a retail company to ingest event logs into Snowflake to derive insights. A Snowflake Architect is asked to define security best practices to configure access control privileges for the data load for auto-ingest to Snowpipe.

What are the MINIMUM object privileges required for the Snowpipe user to execute Snowpipe?

A. OWNERSHIP on the named pipe, USAGE on the named stage, target database, and schema, and INSERT and SELECT on the target table

B. OWNERSHIP on the named pipe, USAGE and READ on the named stage, USAGE on the target database and schema, and INSERT end SELECT on the target table

C. CREATE on the named pipe, USAGE and READ on the named stage, USAGE on the target database and schema, and INSERT end SELECT on the target table

D. USAGE on the named pipe, named stage, target database, and schema, and INSERT and SELECT on the target table

Answer: B

Explanation:

According to the SnowPro Advanced: Architect documents and learning resources, the minimum object privileges required for the Snowpipe user to execute Snowpipe are:

OWNERSHIP on the named pipe. This privilege allows the Snowpipe user to create, modify, and drop the pipe object that defines the COPY statement for loading data from the stage to the table1.

USAGE and READ on the named stage. These privileges allow the Snowpipe user to access and read the data files from the stage that are loaded by Snowpipe2.

USAGE on the target database and schema. These privileges allow the Snowpipe user to access the database and schema that contain the target table3.

INSERT and SELECT on the target table. These privileges allow the Snowpipe user to insert data into the table and select data from the table4.

The other options are incorrect because they do not specify the minimum object privileges required for the Snowpipe user to execute Snowpipe. Option A is incorrect because it does not include the READ privilege on the named stage, which is required for the Snowpipe user to read the data files from the stage. Option C is incorrect because it does not include the OWNERSHIP privilege on the named pipe, which is required for the Snowpipe user to create, modify, and drop the pipe object. Option D is incorrect because it does not include the OWNERSHIP privilege on the named pipe or the READ privilege on the named stage, which are both required for the Snowpipe user to execute Snowpipe. Reference: CREATE PIPE | Snowflake Documentation, CREATE STAGE | Snowflake Documentation, CREATE DATABASE | Snowflake Documentation, CREATE TABLE | Snowflake

Documentation

Question: 12

The IT Security team has identified that there is an ongoing credential stuffing attack on many of their organization's system.
What is the BEST way to find recent and ongoing login attempts to Snowflake?

A. Call the LOGIN_HISTORY Information Schema table function.
B. Query the LOGIN_HISTORY view in the ACCOUNT_USAGE schema in the SNOWFLAKE database.
C. View the History tab in the Snowflake UI and set up a filter for SQL text that contains the text "LOGIN".
D. View the Users section in the Account tab in the Snowflake UI and review the last login column.

Answer: B

Explanation:

This view can be used to query login attempts by Snowflake users within the last 365 days (1 year). It provides information such as the event timestamp, the user name, the client IP, the authentication method, the success or failure status, and the error code or message if the login attempt was unsuccessful. By querying this view, the IT Security team can identify any suspicious or malicious login attempts to Snowflake and take appropriate actions to prevent credential stuffing attacks1. The other options are not the best ways to find recent and ongoing login attempts to Snowflake. Option A is incorrect because the LOGIN_HISTORY Information Schema table function only returns login events within the last 7 days, which may not be sufficient to detect credential stuffing attacks that span a longer period of time2. Option C is incorrect because the History tab in the Snowflake UI only shows the queries executed by the current user or role, not the login events of other users or roles3. Option D is incorrect because the Users section in the Account tab in the Snowflake UI only shows the last login time for each user, not the details of the login attempts or the failures.

Question: 13

An Architect has a VPN_ACCESS_LOGS table in the SECURITY_LOGS schema containing timestamps of the connection and disconnection, username of the user, and summary statistics.
What should the Architect do to enable the Snowflake search optimization service on this table?

A. Assume role with OWNERSHIP on future tables and ADD SEARCH OPTIMIZATION on the SECURITY_LOGS schema.
B. Assume role with ALL PRIVILEGES including ADD SEARCH OPTIMIZATION in the SECURITY LOGS schema.
C. Assume role with OWNERSHIP on VPN_ACCESS_LOGS and ADD SEARCH OPTIMIZATION in the SECURITY_LOGS schema.
D. Assume role with ALL PRIVILEGES on VPN_ACCESS_LOGS and ADD SEARCH OPTIMIZATION in the SECURITY_LOGS schema.

Answer: C

Explanation:

According to the SnowPro Advanced: Architect Exam Study Guide, to enable the search optimization service on a table, the user must have the ADD SEARCH OPTIMIZATION privilege on the table and the schema. The privilege can be granted explicitly or inherited from a higher-level object, such as a database or a role. The OWNERSHIP privilege on a table implies

the ADD SEARCH OPTIMIZATION privilege, so the user who owns the table can enable the search optimization service on it. Therefore, the correct answer is to assume a role with OWNERSHIP on VPN_ACCESS_LOGS and ADD SEARCH OPTIMIZATION in the SECURITY_LOGS schema. This will allow the user to enable the search optimization service on the VPN_ACCESS_LOGS table and any future tables created in the SECURITY_LOGS schema. The other options are incorrect because they either grant excessive privileges or do not grant the required privileges on the table or the schema.
Reference:
SnowPro Advanced: Architect Exam Study Guide, page 11, section 2.3.1 Snowflake Documentation: Enabling the Search Optimization Service

Question: 14

A table contains five columns and it has millions of records. The cardinality distribution of the columns is shown below:

Column	Number of Distinct Values
C1	10,790
C2	108
C3	302,605
C4	1.117,736
C5	2.205,400

Column C4 and C5 are mostly used by SELECT queries in the GROUP BY and ORDER BY clauses. Whereas columns C1, C2 and C3 are heavily used in filter and join conditions of SELECT queries. The Architect must design a clustering key for this table to improve the query performance.
Based on Snowflake recommendations, how should the clustering key columns be ordered while defining the multi-column clustering key?

A. C5, C4, C2
B. C3, C4, C5
C. C1, C3, C2
D. C2, C1, C3

Answer: C

Explanation:

According to the Snowflake documentation, the following are some considerations for choosing clustering for a table1:

Clustering is optimal when either:
You require the fastest possible response times, regardless of cost.
Your improved query performance offsets the credits required to cluster and maintain the table. Clustering is most effective when the clustering key is used in the following types of query predicates:
Filter predicates (e.g. WHERE clauses)
Join predicates (e.g. ON clauses)
Grouping predicates (e.g. GROUP BY clauses)
Sorting predicates (e.g. ORDER BY clauses)
Clustering is less effective when the clustering key is not used in any of the above query predicates, or when the clustering key is used in a predicate that requires a function or expression to be applied to the key (e.g. DATE_TRUNC, TO_CHAR, etc.).

For most tables, Snowflake recommends a maximum of 3 or 4 columns (or expressions) per key.
Adding more than 3-4 columns tends to increase costs more than benefits.
Based on these considerations, the best option for the clustering key columns is C. C1, C3, C2, because:
These columns are heavily used in filter and join conditions of SELECT queries, which are the most effective types of predicates for clustering.
These columns have high cardinality, which means they have many distinct values and can help reduce the clustering skew and improve the compression ratio.
These columns are likely to be correlated with each other, which means they can help co-locate similar rows in the same micro-partitions and improve the scan efficiency.
These columns do not require any functions or expressions to be applied to them, which means they can be directly used in the predicates without affecting the clustering.
Reference:: 1: Considerations for Choosing Clustering for a Table | Snowflake Documentation

Question: 15

Which security, governance, and data protection features require, at a MINIMUM, the Business Critical edition of Snowflake? (Choose two.)

A. Extended Time Travel (up to 90 days)
B. Customer-managed encryption keys through Tri-Secret Secure
C. Periodic rekeying of encrypted data
D. AWS, Azure, or Google Cloud private connectivity to Snowflake
E. Federated authentication and SSO

Answer: B, D

Explanation:

According to the SnowPro Advanced: Architect documents and learning resources, the security, governance, and data protection features that require, at a minimum, the Business Critical edition of Snowflake are:
Customer-managed encryption keys through Tri-Secret Secure. This feature allows customers to manage their own encryption keys for data at rest in Snowflake, using a combination of three secrets: a master key, a service key, and a security password. This provides an additional layer of security and control over the data encryption and decryption process1.
Periodic rekeying of encrypted data. This feature allows customers to periodically rotate the encryption keys for data at rest in Snowflake, using either Snowflake-managed keys or customermanaged keys. This enhances the security and protection of the data by reducing the risk of key compromise or exposure2.
The other options are incorrect because they do not require the Business Critical edition of Snowflake. Option A is incorrect because extended Time Travel (up to 90 days) is available with the Enterprise edition of Snowflake3. Option D is incorrect because AWS, Azure, or Google Cloud private connectivity to Snowflake is available with the Standard edition of Snowflake4. Option E is incorrect because federated authentication and SSO are available with the Standard edition of Snowflake5. Reference: Tri-Secret Secure | Snowflake Documentation, Periodic Rekeying of Encrypted Data | Snowflake Documentation, Snowflake Editions | Snowflake Documentation, Snowflake Network Policies | Snowflake Documentation, Configuring Federated Authentication and SSO | Snowflake Documentation

Question: 16

A company wants to deploy its Snowflake accounts inside its corporate network with no visibility on the internet. The company is using a VPN infrastructure and Virtual Desktop Infrastructure (VDI) for its Snowflake users. The company also

wants to re-use the login credentials set up for the VDI to eliminate redundancy when managing logins.
What Snowflake functionality should be used to meet these requirements? (Choose two.)

A. Set up replication to allow users to connect from outside the company VPN.
B. Provision a unique company Tri-Secret Secure key.
C. Use private connectivity from a cloud provider.
D. Set up SSO for federated authentication.
E. Use a proxy Snowflake account outside the VPN, enabling client redirect for user logins.

Answer: C, D

Explanation:

According to the SnowPro Advanced: Architect documents and learning resources, the Snowflake functionality that should be used to meet these requirements are:
Use private connectivity from a cloud provider. This feature allows customers to connect to Snowflake from their own private network without exposing their data to the public Internet. Snowflake integrates with AWS PrivateLink, Azure Private Link, and Google Cloud Private Service Connect to offer private connectivity from customers' VPCs or VNets to Snowflake endpoints. Customers can control how traffic reaches the Snowflake endpoint and avoid the need for proxies or public IP addresses123.
Set up SSO for federated authentication. This feature allows customers to use their existing identity provider (IdP) to authenticate users for SSO access to Snowflake. Snowflake supports most SAML 2.0- compliant vendors as an IdP, including Okta, Microsoft AD FS, Google G Suite, Microsoft Azure Active Directory, OneLogin, Ping Identity, and PingOne. By setting up SSO for federated authentication, customers can leverage their existing user credentials and profile information, and provide stronger security than username/password authentication4.
The other options are incorrect because they do not meet the requirements or are not feasible. Option A is incorrect because setting up replication does not allow users to connect from outside the company VPN. Replication is a feature of Snowflake that enables copying databases across accounts in different regions and cloud platforms. Replication does not affect the connectivity or visibility of

the accounts5. Option B is incorrect because provisioning a unique company Tri-Secret Secure key does not affect the network or authentication requirements. Tri-Secret Secure is a feature of Snowflake that allows customers to manage their own encryption keys for data at rest in Snowflake, using a combination of three secrets: a master key, a service key, and a security password. Tri-Secret Secure provides an additional layer of security and control over the data encryption and decryption process, but it does not enable private connectivity or SSO6. Option E is incorrect because using a proxy Snowflake account outside the VPN, enabling client redirect for user logins, is not a supported or recommended way of meeting the requirements. Client redirect is a feature of Snowflake that allows customers to connect to a different Snowflake account than the one specified in the connection string. This feature is useful for scenarios such as cross-region failover, data sharing, and account migration, but it does not provide private connectivity or SSO7. Reference: AWS PrivateLink & Snowflake | Snowflake Documentation, Azure Private Link & Snowflake | Snowflake Documentation, Google Cloud Private Service Connect & Snowflake | Snowflake Documentation, Overview of Federated Authentication and SSO | Snowflake
Documentation, Replicating Databases Across Multiple Accounts | Snowflake Documentation, TriSecret Secure | Snowflake Documentation, Redirecting Client Connections | Snowflake Documentation

Question: 17

How do Snowflake databases that are created from shares differ from standard databases that are not created from shares? (Choose three.)

A. Shared databases are read-only.
B. Shared databases must be refreshed in order for new data to be visible.
C. Shared databases cannot be cloned.
D. Shared databases are not supported by Time Travel.
E. Shared databases will have the PUBLIC or INFORMATION_SCHEMA schemas without explicitly granting these schemas to the share.
F. Shared databases can also be created as transient databases.

Answer: A, C, D

Explanation:

According to the SnowPro Advanced: Architect documents and learning resources, the ways that Snowflake databases that are created from shares differ from standard databases that are not created from shares are:
Shared databases are read-only. This means that the data consumers who access the shared databases cannot modify or delete the data or the objects in the databases. The data providers who share the databases have full control over the data and the objects, and can grant or revoke privileges on them1.
Shared databases cannot be cloned. This means that the data consumers who access the shared databases cannot create a copy of the databases or the objects in the databases. The data providers who share the databases can clone the databases or the objects, but the clones are not automatically shared2.
Shared databases are not supported by Time Travel. This means that the data consumers who access the shared databases cannot use the AS OF clause to query historical data or restore deleted data. The data providers who share the databases can use Time Travel on the databases or the objects, but the historical data is not visible to the data consumers3.
The other options are incorrect because they are not ways that Snowflake databases that are created from shares differ from standard databases that are not created from shares. Option B is incorrect because shared databases do not need to be refreshed in order for new data to be visible. The data consumers who access the shared databases can see the latest data as soon as the data providers update the data1. Option E is incorrect because shared databases will not have the PUBLIC or INFORMATION_SCHEMA schemas without explicitly granting these schemas to the share. The data consumers who access the shared databases can only see the objects that the data providers grant to the share, and the PUBLIC and INFORMATION_SCHEMA schemas are not granted by default4. Option F is incorrect because shared databases cannot be created as transient databases. Transient databases are databases that do not support Time Travel or Fail-safe, and can be dropped without affecting the retention period of the data. Shared databases are always created as permanent databases, regardless of the type of the source database5. Reference: Introduction to Secure Data Sharing | Snowflake Documentation, Cloning Objects | Snowflake Documentation, Time Travel | Snowflake Documentation, Working with Shares | Snowflake Documentation, CREATE DATABASE | Snowflake Documentation

Question: 18

What integration object should be used to place restrictions on where data may be exported?

A. Stage integration
B. Security integration
C. Storage integration
D. API integration

Answer: B

Explanation:

According to the SnowPro Advanced: Architect documents and learning resources, the integration object that should be used to place restrictions on where data may be exported is the security integration. A security integration is a Snowflake object that provides an interface between Snowflake and third-party security services, such as Okta, Duo, or Google Authenticator. A security integration can be used to enforce policies on data export, such as requiring multi-factor authentication (MFA) or restricting the export destination to a specific network or domain. A security integration can also be used to enable single sign-on (SSO) or federated authentication for Snowflake users1.

The other options are incorrect because they are not integration objects that can be used to place restrictions on where data may be exported. Option A is incorrect because a stage integration is not a valid type of integration object in Snowflake. A stage is a Snowflake object that references a location where data files are stored, such as an internal stage, an external stage, or a named stage. A stage is not an integration object that provides an interface between Snowflake and third-party services2. Option C is incorrect because a storage integration is a Snowflake object that provides an interface between Snowflake and external cloud storage, such as Amazon S3, Azure Blob Storage, or Google Cloud Storage. A storage integration can be used to securely access data files from external cloud storage without exposing the credentials, but it cannot be used to place restrictions on where data may be exported3. Option D is incorrect because an API integration is a Snowflake object that provides an interface between Snowflake and third-party services that use REST APIs, such as

Salesforce, Slack, or Twilio. An API integration can be used to securely call external REST APIs from Snowflake using the CALL_EXTERNAL_API function, but it cannot be used to place restrictions on where data may be exported4. Reference: CREATE SECURITY INTEGRATION | Snowflake Documentation, CREATE STAGE | Snowflake Documentation, CREATE STORAGE INTEGRATION | Snowflake Documentation, CREATE API INTEGRATION | Snowflake Documentation

Question: 19

The following DDL command was used to create a task based on a stream:

```
CREATE TASK ts insert new customers
    WAREHOUSE = MYWH
    Schedule = '5 minute'
WHEN
    System$STREAM_HAS_DATA('MYSTREAM')
AS
    INSERT INTO new customers(id, name) SELECT id, name
FROM mystream WHERE METADATA?ACTION = 'INSERT';
```

Assuming MY_WH is set to auto_suspend – 60 and used exclusively for this task, which statement is true?

A. The warehouse MY_WH will be made active every five minutes to check the stream.
B. The warehouse MY_WH will only be active when there are results in the stream.
C. The warehouse MY_WH will never suspend.
D. The warehouse MY_WH will automatically resize to accommodate the size of the stream.

Answer: B

Explanation:

The warehouse MY_WH will only be active when there are results in the stream. This is because the task is created based on a stream, which means that the task will only be executed when there are new data in the stream. Additionally, the warehouse is set to auto_suspend - 60, which means that the warehouse will automatically suspend after 60 seconds of inactivity. Therefore, the warehouse will only be active when there are results in the stream. Reference:
[CREATE TASK | Snowflake Documentation]

[Using Streams and Tasks | Snowflake Documentation]
[CREATE WAREHOUSE | Snowflake Documentation]

Question: 20

What is a characteristic of loading data into Snowflake using the Snowflake Connector for Kafka?

A. The Connector only works in Snowflake regions that use AWS infrastructure.
B. The Connector works with all file formats, including text, JSON, Avro, Ore, Parquet, and XML.
C. The Connector creates and manages its own stage, file format, and pipe objects.
D. Loads using the Connector will have lower latency than Snowpipe and will ingest data in real time.

Answer: C

Explanation:

According to the SnowPro Advanced: Architect documents and learning resources, a characteristic of loading data into Snowflake using the Snowflake Connector for Kafka is that the Connector creates and manages its own stage, file format, and pipe objects. The stage is an internal stage that is used to store the data files from the Kafka topics. The file format is a JSON or Avro file format that is used to parse the data files. The pipe is a Snowpipe object that is used to load the data files into the Snowflake table. The Connector automatically creates and configures these objects based on the Kafka configuration properties, and handles the cleanup and maintenance of these objects1.
The other options are incorrect because they are not characteristics of loading data into Snowflake using the Snowflake Connector for Kafka. Option A is incorrect because the Connector works in Snowflake regions that use any cloud infrastructure, not just AWS. The Connector supports AWS, Azure, and Google Cloud platforms, and can load data across different regions and cloud platforms using data replication2. Option B is incorrect because the Connector does not work with all file formats, only JSON and Avro. The Connector expects the data in the Kafka topics to be in JSON or Avro format, and parses the data accordingly. Other file formats, such as text, ORC, Parquet, or XML, are not supported by the Connector3. Option D is incorrect because loads using the Connector do not have lower latency than Snowpipe, and do not ingest data in real time. The Connector uses Snowpipe to load data into Snowflake, and inherits the same latency and performance characteristics of Snowpipe. The Connector does not provide real-time ingestion, but near real-time ingestion, depending on the frequency and size of the data files4. Reference: Installing and Configuring the Kafka Connector | Snowflake Documentation, Sharing Data Across Regions and Cloud Platforms | Snowflake Documentation, Overview of the Kafka Connector | Snowflake Documentation, Using Snowflake Connector for Kafka With Snowpipe Streaming | Snowflake Documentation

Question: 21

A healthcare company wants to share data with a medical institute. The institute is running a Standard edition of Snowflake; the healthcare company is running a Business Critical edition. How can this data be shared?

A. The healthcare company will need to change the institute's Snowflake edition in the accounts panel.
B. By default, sharing is supported from a Business Critical Snowflake edition to a Standard edition.
C. Contact Snowflake and they will execute the share request for the healthcare company.
D. Set the share_restriction parameter on the shared object to false.

Answer: D

Explanation:

By default, Snowflake does not allow sharing data from a Business Critical edition to a non-Business Critical edition. This is

because Business Critical edition provides enhanced security and data protection features that are not available in lower editions. However, this restriction can be overridden by setting the share_restriction parameter on the shared object (database, schema, or table) to false. This parameter allows the data provider to explicitly allow sharing data with lower edition accounts. Note that this parameter can only be set by the data provider, not the data consumer. Also, setting this parameter to false may reduce the level of security and data protection for the shared data.

Reference:

Enable Data Share:Business Critical Account to Lower Edition

Sharing Is Not Allowed From An Account on BUSINESS CRITICAL Edition to an Account On A Lower Edition

SQL Execution Error: Sharing is Not Allowed from an Account on BUSINESS CRITICAL Edition to an Account on a Lower Edition

Snowflake Editions | Snowflake Documentation

Question: 22

An Architect is designing a pipeline to stream event data into Snowflake using the Snowflake Kafka connector. The Architect's highest priority is to configure the connector to stream data in the MOST cost-effective manner.

Which of the following is recommended for optimizing the cost associated with the Snowflake Kafka connector?

A. Utilize a higher Buffer.flush.time in the connector configuration.
B. Utilize a higher Buffer.size.bytes in the connector configuration.
C. Utilize a lower Buffer.size.bytes in the connector configuration.
D. Utilize a lower Buffer.count.records in the connector configuration.

Answer: A

Explanation:

The minimum value supported for the buffer.flush.time property is 1 (in seconds). For higher average data flow rates, we suggest that you decrease the default value for improved latency. If cost is a greater concern than latency, you could increase the buffer flush time. Be careful to flush the Kafka memory buffer before it becomes full to avoid out of memory exceptions.

https://docs.snowflake.com/en/user-guide/data-load-snowpipe-streaming-kafka

Question: 23

An Architect has chosen to separate their Snowflake Production and QA environments using two separate Snowflake accounts.

The QA account is intended to run and test changes on data and database objects before pushing those changes to the Production account. It is a requirement that all database objects and data in the QA account need to be an exact copy of the database objects, including privileges and data in the Production account on at least a nightly basis.

Which is the LEAST complex approach to use to populate the QA account with the Production account's data and database objects on a nightly basis?

A.

1) Create a share in the Production account for each database
2) Share access to the QA account as a Consumer
3) The QA account creates a database directly from each share
4) Create clones of those databases on a nightly basis

5) Run tests directly on those cloned databases

B.

1) Create a stage in the Production account
2) Create a stage in the QA account that points to the same external object-storage location
3) Create a task that runs nightly to unload each table in the Production account into the stage
4) Use Snowpipe to populate the QA account

C.

1) Enable replication for each database in the Production account
2) Create replica databases in the QA account
3) Create clones of the replica databases on a nightly basis
4) Run tests directly on those cloned databases

D.

1) In the Production account, create an external function that connects into the QA account and returns all the data for one specific table
2) Run the external function as part of a stored procedure that loops through each table in the Production account and populates each table in the QA account

Answer: C

Explanation:

This approach is the least complex because it uses Snowflake's built-in replication feature to copy the data and database objects from the Production account to the QA account. Replication is a fast and efficient way to synchronize data across accounts, regions, and cloud platforms. It also preserves the privileges and metadata of the replicated objects. By creating clones of the replica databases, the QA account can run tests on the cloned data without affecting the original dat a. Clones are also zero-copy, meaning they do not consume any additional storage space unless the data is modified. This approach does not require any external stages, tasks, Snowpipe, or external functions, which can add complexity and overhead to the data transfer process.

Reference:

Introduction to Replication and Failover

Replicating Databases Across Multiple Accounts

Cloning Considerations

Question: 24

A user can change object parameters using which of the following roles?

A. ACCOUNTADMIN, SECURITYADMIN
B. SYSADMIN, SECURITYADMIN
C. ACCOUNTADMIN, USER with PRIVILEGE
D. SECURITYADMIN, USER with PRIVILEGE

Answer: C

Explanation:

According to the Snowflake documentation, object parameters are parameters that can be set on individual objects such as databases, schemas, tables, and stages. Object parameters can be set by users with the appropriate privileges on the objects. For example, to set the object parameter AUTO_REFRESH on a table, the user must have the MODIFY privilege on the table. The
ACCOUNTADMIN role has the highest level of privileges on all objects in the account, so it can set any object parameter on any object. However, other roles, such as SECURITYADMIN or SYSADMIN, do not have the same level of privileges on all objects, so they cannot set object parameters on objects they do not own or have the required privileges on. Therefore, the correct answer is C. ACCOUNTADMIN, USER with PRIVILEGE.

Reference:

Parameters | Snowflake Documentation

Object Parameters | Snowflake Documentation
Object Privileges | Snowflake Documentation

Question: 25

A media company needs a data pipeline that will ingest customer review data into a Snowflake table, and apply some transformations. The company also needs to use Amazon Comprehend to do sentiment analysis and make the de-identified final data set available publicly for advertising companies who use different cloud providers in different regions.
The data pipeline needs to run continuously ang efficiently as new records arrive in the object storage leveraging event notifications. Also, the operational complexity, maintenance of the infrastructure, including platform upgrades and security, and the development effort should be minimal.
Which design will meet these requirements?

A. Ingest the data using COPY INTO and use streams and tasks to orchestrate transformations. Export the data into Amazon S3 to do model inference with Amazon Comprehend and ingest the data back into a Snowflake table. Then create a listing in the Snowflake Marketplace to make the data available to other companies.
B. Ingest the data using Snowpipe and use streams and tasks to orchestrate transformations. Create an external function to do model inference with Amazon Comprehend and write the final records to a Snowflake table. Then create a listing in the Snowflake Marketplace to make the data available to other companies.
C. Ingest the data into Snowflake using Amazon EMR and PySpark using the Snowflake Spark connector. Apply transformations using another Spark job. Develop a python program to do model inference by leveraging the Amazon Comprehend text analysis API. Then write the results to a Snowflake table and create a listing in the Snowflake Marketplace to make the data available to other companies.
D. Ingest the data using Snowpipe and use streams and tasks to orchestrate transformations. Export the data into Amazon S3 to do model inference with Amazon Comprehend and ingest the data back into a Snowflake table. Then create a listing in the Snowflake Marketplace to make the data available to other companies.

Answer: B

Explanation:

This design meets all the requirements for the data pipeline. Snowpipe is a feature that enables continuous data loading into Snowflake from object storage using event notifications. It is efficient, scalable, and serverless, meaning it does not require any infrastructure or maintenance from the user. Streams and tasks are features that enable automated data pipelines within Snowflake, using change data capture and scheduled execution. They are also efficient, scalable, and serverless, and they simplify the data transformation process. External functions are functions that can invoke

external services or APIs from within Snowflake. They can be used to integrate with Amazon Comprehend and perform sentiment analysis on the data. The results can be written back to a Snowflake table using standard SQL commands. Snowflake Marketplace is a platform that allows data providers to share data with data consumers across different accounts, regions, and cloud platforms. It is a secure and easy way to make data publicly available to other companies.

Reference:
Snowpipe Overview | Snowflake Documentation
Introduction to Data Pipelines | Snowflake Documentation
External Functions Overview | Snowflake Documentation
Snowflake Data Marketplace Overview | Snowflake Documentation

Question: 26

A Snowflake Architect is designing an application and tenancy strategy for an organization where strong legal isolation rules as well as multi-tenancy are requirements.

Which approach will meet these requirements if Role-Based Access Policies (RBAC) is a viable option for isolating tenants?

A. Create accounts for each tenant in the Snowflake organization.
B. Create an object for each tenant strategy if row level security is viable for isolating tenants.
C. Create an object for each tenant strategy if row level security is not viable for isolating tenants.
D. Create a multi-tenant table strategy if row level security is not viable for isolating tenants.

Answer: A

Explanation:

This approach meets the requirements of strong legal isolation and multi-tenancy. By creating separate accounts for each tenant, the application can ensure that each tenant has its own dedicated storage, compute, and metadata resources, as well as its own encryption keys and security policies. This provides the highest level of isolation and data protection among the tenancy models. Furthermore, by creating the accounts within the same Snowflake organization, the application can leverage the features of Snowflake Organizations, such as centralized billing, account management, and cross-account data sharing.

Reference:

Snowflake Organizations Overview | Snowflake Documentation
Design Patterns for Building Multi-Tenant Applications on Snowflake

Question: 27

Which statements describe characteristics of the use of materialized views in Snowflake? (Choose two.)

A. They can include ORDER BY clauses.
B. They cannot include nested subqueries.
C. They can include context functions, such as CURRENT_TIME().
D. They can support MIN and MAX aggregates.
E. They can support inner joins, but not outer joins.

Answer: B, D

Explanation:

According to the Snowflake documentation, materialized views have some limitations on the query specification that defines them. One of these limitations is that they cannot include nested subqueries, such as subqueries in the FROM clause or scalar subqueries in the SELECT list. Another limitation is that they cannot include ORDER BY clauses, context functions (such as CURRENT_TIME()), or outer joins. However, materialized views can support MIN and MAX aggregates, as well as other aggregate functions, such as SUM, COUNT, and AVG.

Reference:

Limitations on Creating Materialized Views | Snowflake Documentation
Working with Materialized Views | Snowflake Documentation

Question: 28

The Data Engineering team at a large manufacturing company needs to engineer data coming from many sources to support a wide variety of use cases and data consumer requirements which include: 1) Finance and Vendor Management team members who require reporting and visualization 2) Data Science team members who require access to raw data for ML model development 3) Sales team members who require engineered and protected data for data monetization What Snowflake data modeling approaches will meet these requirements? (Choose two.)

A. Consolidate data in the company's data lake and use EXTERNAL TABLES.
B. Create a raw database for landing and persisting raw data entering the data pipelines.
C. Create a set of profile-specific databases that aligns data with usage patterns.
D. Create a single star schema in a single database to support all consumers' requirements.
E. Create a Data Vault as the sole data pipeline endpoint and have all consumers directly access the Vault.

Answer: B, C

Explanation:

These two approaches are recommended by Snowflake for data modeling in a data lake scenario. Creating a raw database allows the data engineering team to ingest data from various sources without any transformation or cleansing, preserving the original data quality and format. This enables the data science team to access the raw data for ML model development. Creating a set of profile-specific databases allows the data engineering team to apply different transformations and optimizations for different use cases and data consumer requirements. For example, the finance and vendor management team can access a dimensional database that supports reporting and visualization, while the sales team can access a secure database that supports data monetization. Reference:
Snowflake Data Lake Architecture | Snowflake Documentation
Snowflake Data Lake Best Practices | Snowflake Documentation

Question: 29

An Architect on a new project has been asked to design an architecture that meets Snowflake security, compliance, and governance requirements as follows:
1) Use Tri-Secret Secure in Snowflake
2) Share some information stored in a view with another Snowflake customer
3) Hide portions of sensitive information from some columns
4) Use zero-copy cloning to refresh the non-production environment from the production environment
To meet these requirements, which design elements must be implemented? (Choose three.)

A. Define row access policies.
B. Use the Business-Critical edition of Snowflake.
C. Create a secure view.
D. Use the Enterprise edition of Snowflake.
E. Use Dynamic Data Masking.
F. Create a materialized view.

Answer: B, C, E

Explanation:

These three design elements are required to meet the security, compliance, and governance requirements for the project.

To use Tri-Secret Secure in Snowflake, the Business Critical edition of Snowflake is required. This edition provides enhanced data protection features, such as customer-managed encryption keys, that are not available in lower editions. Tri-Secret Secure is a feature that combines a Snowflake- maintained key and a customer-managed key to create a composite master key to encrypt the data in Snowflake1.

To share some information stored in a view with another Snowflake customer, a secure view is recommended. A secure view is a view that hides the underlying data and the view definition from unauthorized users. Only the owner of the view and the users who are granted the owner's role can see the view definition and the data in the base tables of the view2. A secure view can be shared with another Snowflake account using a data share3.

To hide portions of sensitive information from some columns, Dynamic Data Masking can be used. Dynamic Data Masking is a feature that allows applying masking policies to columns to selectively mask plain-text data at query time. Depending on the masking policy conditions and the user's role, the data can be fully or partially masked, or shown as plain-text4.

Question: 30

Which of the following are characteristics of how row access policies can be applied to external tables? (Choose three.)

A. An external table can be created with a row access policy, and the policy can be applied to the VALUE column.
B. A row access policy can be applied to the VALUE column of an existing external table.
C. A row access policy cannot be directly added to a virtual column of an external table.
D. External tables are supported as mapping tables in a row access policy.
E. While cloning a database, both the row access policy and the external table will be cloned.
F. A row access policy cannot be applied to a view created on top of an external table.

Answer: A, B, C

Explanation:

These three statements are true according to the Snowflake documentation and the web search results. A row access policy is a feature that allows filtering rows based on user-defined conditions. A row access policy can be applied to an external table, which is a table that reads data from external files in a stage. However, there are some limitations and considerations for using row access policies with external tables.

An external table can be created with a row access policy by using the WITH ROW ACCESS POLICY clause in the CREATE EXTERNAL TABLE statement. The policy can be applied to the VALUE column, which is the column that contains the raw data from the external files in a VARIANT data type1.

A row access policy can also be applied to the VALUE column of an existing external table by using the ALTER TABLE statement with the SET ROW ACCESS POLICY clause2.

A row access policy cannot be directly added to a virtual column of an external table. A virtual column is a column that is derived from the VALUE column using an expression. To apply a row access policy to a virtual column, the policy must be applied to the VALUE column and the expression must be repeated in the policy definition3.

External tables are not supported as mapping tables in a row access policy. A mapping table is a table that is used to determine the access rights of users or roles based on some criteria. Snowflake does not support using an external table as a mapping table because it may cause performance issues or errors4.

While cloning a database, Snowflake clones the row access policy, but not the external table. Therefore, the policy in the cloned database refers to a table that is not present in the cloned database. To avoid this issue, the external table must be manually cloned or recreated in the cloned database4.

A row access policy can be applied to a view created on top of an external table. The policy can be applied to the view itself or to the underlying external table. However, if the policy is applied to the view, the view must be a secure view, which is a view that hides the underlying data and the view definition from unauthorized users5.

Reference:

CREATE EXTERNAL TABLE | Snowflake Documentation
ALTER EXTERNAL TABLE | Snowflake Documentation
Understanding Row Access Policies | Snowflake Documentation
Snowflake Data Governance: Row Access Policy Overview
Secure Views | Snowflake Documentation

Question: 31

An Architect needs to allow a user to create a database from an inbound share.
To meet this requirement, the user's role must have which privileges? (Choose two.)

A. IMPORT SHARE;
B. IMPORT PRIVILEGES;
C. CREATE DATABASE;
D. CREATE SHARE;
E. IMPORT DATABASE;

Answer: C, E

Explanation:

According to the Snowflake documentation, to create a database from an inbound share, the user's role must have the following privileges:
The CREATE DATABASE privilege on the current account. This privilege allows the user to create a new database in the account1.
The IMPORT DATABASE privilege on the share. This privilege allows the user to import a database from the share into the account2. The other privileges listed are not relevant for this requirement. The IMPORT SHARE privilege is used to import a share into the account, not a database3. The IMPORT PRIVILEGES privilege is used to import the privileges granted on the shared objects, not the objects themselves2. The CREATE SHARE privilege is used to create a share to provide data to other accounts, not to consume data from other accounts4.
Reference:
CREATE DATABASE | Snowflake Documentation
Importing Data from a Share | Snowflake Documentation
Importing a Share | Snowflake Documentation
CREATE SHARE | Snowflake Documentation

Question: 32

Files arrive in an external stage every 10 seconds from a proprietary system. The files range in size from 500 K to 3 MB. The data must be accessible by dashboards as soon as it arrives.
How can a Snowflake Architect meet this requirement with the LEAST amount of coding? (Choose two.)

A. Use Snowpipe with auto-ingest.
B. Use a COPY command with a task.
C. Use a materialized view on an external table.
D. Use the COPY INTO command.
E. Use a combination of a task and a stream.

Answer: A, C

Explanation:

These two options are the best ways to meet the requirement of loading data from an external stage and making it accessible by dashboards with the least amount of coding.

Snowpipe with auto-ingest is a feature that enables continuous and automated data loading from an external stage into a Snowflake table. Snowpipe uses event notifications from the cloud storage service to detect new or modified files in the stage and triggers a COPY INTO command to load the data into the table. Snowpipe is efficient, scalable, and serverless, meaning it does not require any infrastructure or maintenance from the user. Snowpipe also supports loading data from files of any size, as long as they are in a supported format1.

A materialized view on an external table is a feature that enables creating a pre-computed result set from an external table and storing it in Snowflake. A materialized view can improve the performance and efficiency of querying data from an external table, especially for complex queries or dashboards. A materialized view can also support aggregations, joins, and filters on the external table data. A materialized view on an external table is automatically refreshed when the underlying data in the external stage changes, as long as the AUTO_REFRESH parameter is set to true2.

Reference:

Snowpipe Overview | Snowflake Documentation

Materialized Views on External Tables | Snowflake Documentation

Question: 33

When loading data into a table that captures the load time in a column with a default value of either CURRENT_TIME () or CURRENT_TIMESTAMP() what will occur?

A. All rows loaded using a specific COPY statement will have varying timestamps based on when the rows were inserted.

B. Any rows loaded using a specific COPY statement will have varying timestamps based on when the rows were read from the source.

C. Any rows loaded using a specific COPY statement will have varying timestamps based on when the rows were created in the source.

D. All rows loaded using a specific COPY statement will have the same timestamp value.

Answer: D

Explanation:

According to the Snowflake documentation, when loading data into a table that captures the load time in a column with a default value of either CURRENT_TIME () or CURRENT_TIMESTAMP(), the default value is evaluated once per COPY statement, not once per row. Therefore, all rows loaded using a specific COPY statement will have the same timestamp value. This behavior ensures that the timestamp value reflects the time when the data was loaded into the table, not when the data was read from the source or created in the source. Reference:

Snowflake Documentation: Loading Data into Tables with Default Values

Snowflake Documentation: COPY INTO table

Question: 34

How does a standard virtual warehouse policy work in Snowflake?

A. It conserves credits by keeping running clusters fully loaded rather than starting additional clusters.

B. It starts only if the system estimates that there is a query load that will keep the cluster busy for at least 6 minutes.

C. It starts only f the system estimates that there is a query load that will keep the cluster busy for at least 2 minutes.

D. It prevents or minimizes queuing by starting additional clusters instead of conserving credits.

Answer: D

Explanation:

A standard virtual warehouse policy is one of the two scaling policies available for multi-cluster warehouses in Snowflake. The other policy is economic. A standard policy aims to prevent or minimize queuing by starting additional clusters as soon as the current cluster is fully loaded, regardless of the number of queries in the queue. This policy can improve query performance and concurrency, but it may also consume more credits than an economic policy, which tries to conserve credits by keeping the running clusters fully loaded before starting additional clusters. The scaling policy can be set when creating or modifying a warehouse, and it can be changed at any time. Reference:
Snowflake Documentation: Multi-cluster Warehouses
Snowflake Documentation: Scaling Policy for Multi-cluster Warehouses

Question: 35

Which feature provides the capability to define an alternate cluster key for a table with an existing cluster key?

A. External table
B. Materialized view
C. Search optimization
D. Result cache

Answer: B

Explanation:

A materialized view is a feature that provides the capability to define an alternate cluster key for a table with an existing cluster key. A materialized view is a pre-computed result set that is stored in Snowflake and can be queried like a regular table. A materialized view can have a different cluster key than the base table, which can improve the performance and efficiency of queries on the materialized view. A materialized view can also support aggregations, joins, and filters on the base table data. A materialized view is automatically refreshed when the underlying data in the base table changes, as long as the AUTO_REFRESH parameter is set to true1.
Reference:
Materialized Views | Snowflake Documentation

Question: 36

An Architect would like to save quarter-end financial results for the previous six years.
Which Snowflake feature can the Architect use to accomplish this?

A. Search optimization service
B. Materialized view
C. Time Travel
D. Zero-copy cloning
E. Secure views

Answer: D

Explanation:

Zero-copy cloning is a Snowflake feature that can be used to save quarter-end financial results for the previous six years. Zero-copy cloning allows creating a copy of a database, schema, table, or view without duplicating the data or metadata. The clone shares the same data files as the original object, but tracks any changes made to the clone or the original separately. Zero-copy cloning can be used to create snapshots of data at different points in time, such as quarter-end financial results, and preserve them for future analysis or comparison. Zero-copy cloning is fast, efficient, and does not consume any additional storage space unless the data is modified1.

Reference:

Zero-Copy Cloning | Snowflake Documentation

Question: 37

A company is using a Snowflake account in Azure. The account has SAML SSO set up using ADFS as a SCIM identity provider. To validate Private Link connectivity, an Architect performed the following steps:

* Confirmed Private Link URLs are working by logging in with a username/password account
* Verified DNS resolution by running nslookups against Private Link URLs
* Validated connectivity using SnowCD
* Disabled public access using a network policy set to use the company's IP address range However, the following error message is received when using SSO to log into the company account: IP XX.XXX.XX.XX is not allowed to access snowflake. Contact your local security administrator.

What steps should the Architect take to resolve this error and ensure that the account is accessed using only Private Link? (Choose two.)

A. Alter the Azure security integration to use the Private Link URLs.
B. Add the IP address in the error message to the allowed list in the network policy.
C. Generate a new SCIM access token using system$generate_scim_access_token and save it to Azure AD.
D. Update the configuration of the Azure AD SSO to use the Private Link URLs.
E. Open a case with Snowflake Support to authorize the Private Link URLs' access to the account.

Answer: B, D

Explanation:

The error message indicates that the IP address in the error message is not allowed to access Snowflake because it is not in the allowed list of the network policy. The network policy is a feature that allows restricting access to Snowflake based on IP addresses or ranges. To resolve this error, the Architect should take the following steps:

Add the IP address in the error message to the allowed list in the network policy. This will allow the IP address to access Snowflake using the Private Link URLs. Alternatively, the Architect can disable the network policy if it is not required for security reasons.

Update the configuration of the Azure AD SSO to use the Private Link URLs. This will ensure that the SSO authentication process uses the Private Link URLs instead of the public URLs. The configuration can be updated by following the steps in the Azure documentation1.

These two steps should resolve the error and ensure that the account is accessed using only Private Link. The other options are not necessary or relevant for this scenario. Altering the Azure security integration to use the Private Link URLs is not required because the security integration is used for SCIM provisioning, not for SSO authentication.

Generating a new SCIM access token using system$generate_scim_access_token and saving it to Azure AD is not required because the SCIM access token is used for SCIM provisioning, not for SSO authentication. Opening a case with Snowflake Support to authorize the Private Link URLs' access to the account is not required because the authorization can be done by the account administrator using the SYSTEM$AUTHORIZE_PRIVATELINK function2.

Question: 38

Which steps are recommended best practices for prioritizing cluster keys in Snowflake? (Choose two.)

A. Choose columns that are frequently used in join predicates.
B. Choose lower cardinality columns to support clustering keys and cost effectiveness.
C. Choose TIMESTAMP columns with nanoseconds for the highest number of unique rows.
D. Choose cluster columns that are most actively used in selective filters.
E. Choose cluster columns that are actively used in the GROUP BY clauses.

Answer: A, D

Explanation:

According to the Snowflake documentation, the best practices for choosing clustering keys are: Choose columns that are frequently used in join predicates. This can improve the join performance by reducing the number of micro-partitions that need to be scanned and joined.
Choose columns that are most actively used in selective filters. This can improve the scan efficiency by skipping micro-partitions that do not match the filter predicates.
Avoid using low cardinality columns, such as gender or country, as clustering keys. This can result in poor clustering and high maintenance costs.
Avoid using TIMESTAMP columns with nanoseconds, as they tend to have very high cardinality and low correlation with other columns. This can also result in poor clustering and high maintenance costs.
Avoid using columns with duplicate values or NULLs, as they can cause skew in the clustering and reduce the benefits of pruning.
Cluster on multiple columns if the queries use multiple filters or join predicates. This can increase the chances of pruning more micro-partitions and improve the compression ratio.
Clustering is not always useful, especially for small or medium-sized tables, or tables that are not frequently queried or updated. Clustering can incur additional costs for initially clustering the data and maintaining the clustering over time.
Reference:
Clustering Keys & Clustered Tables | Snowflake Documentation
[Considerations for Choosing Clustering for a Table | Snowflake Documentation]

Question: 39

Which Snowflake data modeling approach is designed for BI queries?

A. 3 NF
B. Star schema
C. Data Vault
D. Snowflake schema

Answer: B

Explanation:

A star schema is a Snowflake data modeling approach that is designed for BI queries. A star schema is a type of dimensional modeling that organizes data into fact tables and dimension tables. A fact table contains the measures or metrics of the business process, such as sales amount, order quantity, or profit margin. A dimension table contains the

attributes or descriptors of the business process, such as product name, customer name, or order date. A star schema is called so because it resembles a star, with one fact table in the center and multiple dimension tables radiating from it. A star schema can improve the performance and simplicity of BI queries by reducing the number of joins, providing fast access to aggregated data, and enabling intuitive query syntax. A star schema can also support various types of analysis, such as trend analysis, slice and dice, drill down, and roll up12.

Reference:

Snowflake Documentation: Dimensional Modeling

Snowflake Documentation: Star Schema

Question: 40

How is the change of local time due to daylight savings time handled in Snowflake tasks? (Choose two.)

A. A task scheduled in a UTC-based schedule will have no issues with the time changes.

B. Task schedules can be designed to follow specified or local time zones to accommodate the time changes.

C. A task will move to a suspended state during the daylight savings time change.

D. A frequent task execution schedule like minutes may not cause a problem, but will affect the task history.

E. A task schedule will follow only the specified time and will fail to handle lost or duplicated hours.

Answer: A, B

Explanation:

According to the Snowflake documentation1 and the web search results2, these two statements are true about how the change of local time due to daylight savings time is handled in Snowflake tasks. A task is a feature that allows scheduling and executing SQL statements or stored procedures in Snowflake. A task can be scheduled using a cron expression that specifies the frequency and time zone of the task execution.

A task scheduled in a UTC-based schedule will have no issues with the time changes. UTC is a universal time standard that does not observe daylight savings time. Therefore, a task that uses UTC as the time zone will run at the same time throughout the year, regardless of the local time changes1.

Task schedules can be designed to follow specified or local time zones to accommodate the time changes. Snowflake supports using any valid IANA time zone identifier in the cron expression for a task. This allows the task to run according to the local time of the specified time zone, which may

include daylight savings time adjustments. For example, a task that uses Europe/London as the time zone will run one hour earlier or later when the local time switches between GMT and BST12.

Reference:

Snowflake Documentation: Scheduling Tasks

Snowflake Community: Do the timezones used in scheduling tasks in Snowflake adhere to daylight savings?

Question: 41

An Architect needs to grant a group of ORDER_ADMIN users the ability to clean old data in an ORDERS table (deleting all records older than 5 years), without granting any privileges on the table. The group's manager (ORDER_MANAGER) has full DELETE privileges on the table.

How can the ORDER_ADMIN role be enabled to perform this data cleanup, without needing the DELETE privilege held by the ORDER_MANAGER role?

A. Create a stored procedure that runs with caller's rights, including the appropriate "> 5 years" business logic, and

grant USAGE on this procedure to ORDER_ADMIN. The ORDER_MANAGER role owns the procedure.

B. Create a stored procedure that can be run using both caller's and owner's rights (allowing the user to specify which rights are used during execution), and grant USAGE on this procedure to ORDER_ADMIN. The ORDER_MANAGER role owns the procedure.

C. Create a stored procedure that runs with owner's rights, including the appropriate "> 5 years" business logic, and grant USAGE on this procedure to ORDER_ADMIN. The ORDER_MANAGER role owns the procedure.

D. This scenario would actually not be possible in Snowflake – any user performing a DELETE on a table requires the DELETE privilege to be granted to the role they are using.

Answer: C

Explanation:

This is the correct answer because it allows the ORDER_ADMIN role to perform the data cleanup without needing the DELETE privilege on the ORDERS table. A stored procedure is a feature that allows scheduling and executing SQL statements or stored procedures in Snowflake. A stored procedure can run with either the caller's rights or the owner's rights. A caller's rights stored procedure runs with the privileges of the role that called the stored procedure, while an owner's rights stored procedure runs with the privileges of the role that created the stored procedure. By creating a stored procedure that runs with owner's rights, the ORDER_MANAGER role can delegate the specific task of deleting old data to the ORDER_ADMIN role, without granting the ORDER_ADMIN role more general privileges on the ORDERS table. The stored procedure must include the appropriate business logic to delete only the records older than 5 years, and the ORDER_MANAGER role must grant the USAGE privilege on the stored procedure to the ORDER_ADMIN role. The ORDER_ADMIN role can then execute the stored procedure to perform the data cleanup12.

Reference:

Snowflake Documentation: Stored Procedures

Snowflake Documentation: Understanding Caller's Rights and Owner's Rights Stored Procedures

Question: 42

A company's daily Snowflake workload consists of a huge number of concurrent queries triggered between 9pm and 11pm. At the individual level, these queries are smaller statements that get completed within a short time period.

What configuration can the company's Architect implement to enhance the performance of this workload? (Choose two.)

A. Enable a multi-clustered virtual warehouse in maximized mode during the workload duration.

B. Set the MAX_CONCURRENCY_LEVEL to a higher value than its default value of 8 at the virtual warehouse level.

C. Increase the size of the virtual warehouse to size X-Large.

D. Reduce the amount of data that is being processed through this workload.

E. Set the connection timeout to a higher value than its default.

Answer: A, B

Explanation:

These two configuration options can enhance the performance of the workload that consists of a huge number of concurrent queries that are smaller and faster.

Enabling a multi-clustered virtual warehouse in maximized mode allows the warehouse to scale out automatically by adding more clusters as soon as the current cluster is fully loaded, regardless of the number of queries in the queue. This can improve the concurrency and throughput of the workload by minimizing or preventing queuing. The maximized mode is suitable for workloads that require high performance and low latency, and are less sensitive to credit

consumption1.

Setting the MAX_CONCURRENCY_LEVEL to a higher value than its default value of 8 at the virtual warehouse level allows the warehouse to run more queries concurrently on each cluster. This can improve the utilization and efficiency of the warehouse resources, especially for smaller and faster queries that do not require a lot of processing power. The MAX_CONCURRENCY_LEVEL parameter can be set when creating or modifying a warehouse, and it can be changed at any time2. Reference:

Snowflake Documentation: Scaling Policy for Multi-cluster Warehouses

Snowflake Documentation: MAX_CONCURRENCY_LEVEL

Question: 43

Which system functions does Snowflake provide to monitor clustering information within a table (Choose two.)

A. SYSTEM$CLUSTERING_INFORMATION
B. SYSTEM$CLUSTERING_USAGE
C. SYSTEM$CLUSTERING_DEPTH
D. SYSTEM$CLUSTERING_KEYS
E. SYSTEM$CLUSTERING_PERCENT

Answer: A, C

Explanation:

According to the Snowflake documentation, these two system functions are provided by Snowflake to monitor clustering information within a table. A system function is a type of function that allows executing actions or returning information about the system. A clustering key is a feature that allows organizing data across micro-partitions based on one or more columns in the table. Clustering can improve query performance by reducing the number of files to scan.

SYSTEM$CLUSTERING_INFORMATION is a system function that returns clustering information, including average clustering depth, for a table based on one or more columns in the table. The function takes a table name and an optional column name or expression as arguments, and returns a JSON string with the clustering information. The clustering information includes the cluster by keys, the total partition count, the total constant partition count, the average overlaps, and the average depth1.

SYSTEM$CLUSTERING_DEPTH is a system function that returns the clustering depth for a table based on one or more columns in the table. The function takes a table name and an optional column name or expression as arguments, and returns an integer value with the clustering depth. The clustering depth is the maximum number of overlapping micro-partitions for any micro-partition in the table. A lower clustering depth indicates a better clustering2.

Reference:

SYSTEM$CLUSTERING_INFORMATION | Snowflake Documentation

SYSTEM$CLUSTERING_DEPTH | Snowflake Documentation

Question: 44

A company has a table with that has corrupted data, named Dat

a. The company wants to recover the data as it was 5 minutes ago using cloning and Time Travel. What command will accomplish this?

A. CREATE CLONE TABLE Recover_Data FROM Data AT(OFFSET => -60*5);
B. CREATE CLONE Recover_Data FROM Data AT(OFFSET => -60*5);
C. CREATE TABLE Recover_Data CLONE Data AT(OFFSET => -60*5);
D. CREATE TABLE Recover Data CLONE Data AT(TIME => -60*5);

Answer: C

Explanation:

This is the correct command to create a clone of the table Data as it was 5 minutes ago using cloning and Time Travel. Cloning is a feature that allows creating a copy of a database, schema, table, or view without duplicating the data or metadata. Time Travel is a feature that enables accessing historical data (i.e. data that has been changed or deleted) at any point within a defined period. To create a clone of a table at a point in time in the past, the syntax is:
CREATE TABLE <clone_name> CLONE <source_table> AT (OFFSET => <offset_in_seconds>);
The OFFSET parameter specifies the time difference in seconds from the present time. A negative value indicates a point in the past. For example, -60*5 means 5 minutes ago. Alternatively, the TIMESTAMP parameter can be used to specify an exact timestamp in the past. The clone will contain the data as it existed in the source table at the specified point in time12.

Reference:
Snowflake Documentation: Cloning Objects
Snowflake Documentation: Cloning Objects at a Point in Time in the Past

Question: 45

A company has an inbound share set up with eight tables and five secure views. The company plans to make the share part of its production data pipelines.
Which actions can the company take with the inbound share? (Choose two.)

A. Clone a table from a share.
B. Grant modify permissions on the share.
C. Create a table from the shared database.
D. Create additional views inside the shared database.
E. Create a table stream on the shared table.

Answer: A, D

Explanation:

These two actions are possible with an inbound share, according to the Snowflake documentation and the web search results. An inbound share is a share that is created by another Snowflake account (the provider) and imported into your account (the consumer). An inbound share allows you to access the data shared by the provider, but not to modify or delete it. However, you can perform some actions with the inbound share, such as:
Clone a table from a share. You can create a copy of a table from an inbound share using the CREATE TABLE … CLONE statement. The clone will contain the same data and metadata as the original table, but it will be independent of the share. You can modify or delete the clone as you wish, but it will not reflect any changes made to the original table by the provider1.
Create additional views inside the shared database. You can create views on the tables or views from an inbound share using the CREATE VIEW statement. The views will be stored in the shared database, but they will be owned by your account. You can query the views as you would query any other view in your account, but you cannot modify or delete the underlying objects from the share2.
The other actions listed are not possible with an inbound share, because they would require modifying the share or the shared objects, which are read-only for the consumer. You cannot grant modify permissions on the share, create a table from the shared database, or create a table stream on the shared table34.

Reference:
Cloning Objects from a Share | Snowflake Documentation
Creating Views on Shared Data | Snowflake Documentation

Importing Data from a Share | Snowflake Documentation
Streams on Shared Tables | Snowflake Documentation

Question: 46

A company has several sites in different regions from which the company wants to ingest data. Which of the following will enable this type of data ingestion?

A. The company must have a Snowflake account in each cloud region to be able to ingest data to that account.
B. The company must replicate data between Snowflake accounts.
C. The company should provision a reader account to each site and ingest the data through the reader accounts.
D. The company should use a storage integration for the external stage.

Answer: D

Explanation:

This is the correct answer because it allows the company to ingest data from different regions using a storage integration for the external stage. A storage integration is a feature that enables secure and easy access to files in external cloud storage from Snowflake. A storage integration can be used to create an external stage, which is a named location that references the files in the external storage. An external stage can be used to load data into Snowflake tables using the COPY INTO command, or to unload data from Snowflake tables using the COPY INTO LOCATION command. A storage integration can support multiple regions and cloud platforms, as long as the external storage service is compatible with Snowflake12.

Reference:
Snowflake Documentation: Storage Integrations
Snowflake Documentation: External Stages

Question: 47

What Snowflake features should be leveraged when modeling using Data Vault?

A. Snowflake's support of multi-table inserts into the data model's Data Vault tables
B. Data needs to be pre-partitioned to obtain a superior data access performance
C. Scaling up the virtual warehouses will support parallel processing of new source loads D. Snowflake's ability to hash keys so that hash key joins can run faster than integer joins

Answer: A, C

Explanation:

These two features are relevant for modeling using Data Vault on Snowflake. Data Vault is a data modeling approach that organizes data into hubs, links, and satellites. Data Vault is designed to enable high scalability, flexibility, and performance for data integration and analytics. Snowflake is a cloud data platform that supports various data modeling techniques, including Data Vault. Snowflake provides some features that can enhance the Data Vault modeling, such as: Snowflake's support of multi-table inserts into the data model's Data Vault tables. Multi-table inserts (MTI) are a feature that allows inserting data from a single query into multiple tables in a single DML statement. MTI can improve the performance and efficiency of loading data into Data Vault tables, especially for real-time or near-real-time data integration. MTI can also reduce the complexity and maintenance of the loading code, as well as the data duplication

and latency12.

Scaling up the virtual warehouses will support parallel processing of new source loads. Virtual warehouses are a feature that allows provisioning compute resources on demand for data processing. Virtual warehouses can be scaled up or down by changing the size of the warehouse, which determines the number of servers in the warehouse. Scaling up the virtual warehouses can improve the performance and concurrency of processing new source loads into Data Vault tables, especially for large or complex data sets. Scaling up the virtual warehouses can also leverage the parallelism and distribution of Snowflake's architecture, which can optimize the data loading and querying34.

Reference:

Snowflake Documentation: Multi-table Inserts
Snowflake Blog: Tips for Optimizing the Data Vault Architecture on Snowflake
Snowflake Documentation: Virtual Warehouses
Snowflake Blog: Building a Real-Time Data Vault in Snowflake

Question: 48

A company's client application supports multiple authentication methods, and is using Okta.

What is the best practice recommendation for the order of priority when applications authenticate to Snowflake?

A.

1) OAuth (either Snowflake OAuth or External OAuth)
2) External browser
3) Okta native authentication
4) Key Pair Authentication, mostly used for service account users
5) Password

B.

1) External browser, SSO
2) Key Pair Authentication, mostly used for development environment users
3) Okta native authentication
4) OAuth (ether Snowflake OAuth or External OAuth)
5) Password

C.

1) Okta native authentication
2) Key Pair Authentication, mostly used for production environment users
3) Password
4) OAuth (either Snowflake OAuth or External OAuth)
5) External browser, SSO

D.

1) Password
2) Key Pair Authentication, mostly used for production environment users
3) Okta native authentication
4) OAuth (either Snowflake OAuth or External OAuth)
5) External browser, SSO

Answer: A

Explanation:

This is the best practice recommendation for the order of priority when applications authenticate to Snowflake, according to the Snowflake documentation and the web search results. Authentication is the process of verifying the identity of a user or application that connects to Snowflake. Snowflake supports multiple authentication methods, each with different advantages and disadvantages. The recommended order of priority is based on the following factors:

Security: The authentication method should provide a high level of security and protection against unauthorized access

or data breaches. The authentication method should also support multi-factor authentication (MFA) or single sign-on (SSO) for additional security.

Convenience: The authentication method should provide a smooth and easy user experience, without requiring complex or manual steps. The authentication method should also support seamless integration with external identity providers or applications.

Flexibility: The authentication method should provide a range of options and features to suit different use cases and scenarios. The authentication method should also support customization and configuration to meet specific requirements.

Based on these factors, the recommended order of priority is:

OAuth (either Snowflake OAuth or External OAuth): OAuth is an open standard for authorization that allows applications to access Snowflake resources on behalf of a user, without exposing the user's credentials. OAuth provides a high level of security, convenience, and flexibility, as it supports MFA, SSO, token-based authentication, and various grant types and scopes. OAuth can be implemented using either Snowflake OAuth or External OAuth, depending on the identity provider and the application12.

External browser: External browser is an authentication method that allows users to log in to Snowflake using a web browser and an external identity provider, such as Okta, Azure AD, or Ping Identity. External browser provides a high level of security and convenience, as it supports MFA, SSO, and federated authentication. External browser also provides a consistent user interface and experience across different platforms and devices34.

Okta native authentication: Okta native authentication is an authentication method that allows users to log in to Snowflake using Okta as the identity provider, without using a web browser. Okta native authentication provides a high level of security and convenience, as it supports MFA, SSO, and federated authentication. Okta native authentication also provides a native user interface and experience for Okta users, and supports various Okta features, such as password policies and user management56.

Key Pair Authentication: Key Pair Authentication is an authentication method that allows users to log in to Snowflake using a public-private key pair, without using a password. Key Pair Authentication provides a high level of security, as it relies on asymmetric encryption and digital signatures. Key Pair Authentication also provides a flexible and customizable authentication option, as it supports various key formats, algorithms, and expiration times. Key Pair Authentication is mostly used for service account users, such as applications or scripts that connect to Snowflake programmatically7 .

Password: Password is the simplest and most basic authentication method that allows users to log in to Snowflake using a username and password. Password provides a low level of security, as it relies on symmetric encryption and is vulnerable to brute force attacks or phishing. Password also provides a low level of convenience and flexibility, as it requires manual input and management, and does not support MFA or SSO. Password is the least recommended authentication method, and should be used only as a last resort or for testing purposes .

Reference:

Snowflake Documentation: Snowflake OAuth

Snowflake Documentation: External OAuth

Snowflake Documentation: External Browser Authentication

Snowflake Blog: How to Use External Browser Authentication with Snowflake

Snowflake Documentation: Okta Native Authentication

Snowflake Blog: How to Use Okta Native Authentication with Snowflake

Snowflake Documentation: Key Pair Authentication

[Snowflake Blog: How to Use Key Pair Authentication with Snowflake]

[Snowflake Documentation: Password Authentication]

[Snowflake Blog: How to Use Password Authentication with Snowflake]

Question: 49

What is a valid object hierarchy when building a Snowflake environment?

A. Account --> Database --> Schema --> Warehouse

B. Organization --> Account --> Database --> Schema --> Stage

C. Account --> Schema > Table --> Stage
D. Organization --> Account --> Stage --> Table --> View

Answer: B

Explanation:

This is the valid object hierarchy when building a Snowflake environment, according to the Snowflake documentation and the web search results. Snowflake is a cloud data platform that supports various types of objects, such as databases, schemas, tables, views, stages, warehouses, and more. These objects are organized in a hierarchical structure, as follows:

Organization: An organization is the top-level entity that represents a group of Snowflake accounts that are related by business needs or ownership. An organization can have one or more accounts, and can enable features such as cross-account data sharing, billing and usage reporting, and single sign-on across accounts12.

Account: An account is the primary entity that represents a Snowflake customer. An account can have one or more databases, schemas, stages, warehouses, and other objects. An account can also have one or more users, roles, and security integrations. An account is associated with a specific cloud platform, region, and Snowflake edition34.

Database: A database is a logical grouping of schemas. A database can have one or more schemas, and can store structured, semi-structured, or unstructured data. A database can also have properties such as retention time, encryption, and ownership56.

Schema: A schema is a logical grouping of tables, views, stages, and other objects. A schema can have one or more objects, and can define the namespace and access control for the objects. A schema can also have properties such as ownership and default warehouse .

Stage: A stage is a named location that references the files in external or internal storage. A stage can be used to load data into Snowflake tables using the COPY INTO command, or to unload data from Snowflake tables using the COPY INTO LOCATION command. A stage can be created at the account, database, or schema level, and can have properties such as file format, encryption, and credentials . The other options listed are not valid object hierarchies, because they either omit or misplace some objects in the structure. For example, option A omits the organization level and places the warehouse under the schema level, which is incorrect. Option C omits the organization, account, and stage levels, and places the table under the schema level, which is incorrect. Option D omits the database level and places the stage and table under the account level, which is incorrect.

Reference:

Snowflake Documentation: Organizations
Snowflake Blog: Introducing Organizations in Snowflake
Snowflake Documentation: Accounts
Snowflake Blog: Understanding Snowflake Account Structures
Snowflake Documentation: Databases
Snowflake Blog: How to Create a Database in Snowflake [Snowflake Documentation: Schemas]
[Snowflake Blog: How to Create a Schema in Snowflake]
[Snowflake Documentation: Stages]
[Snowflake Blog: How to Use Stages in Snowflake]

Question: 50

Which of the following are characteristics of Snowflake's parameter hierarchy?

A. Session parameters override virtual warehouse parameters.
B. Virtual warehouse parameters override user parameters.
C. Table parameters override virtual warehouse parameters.
D. Schema parameters override account parameters.

Answer: D

Explanation:

This is the correct answer because it reflects the characteristics of Snowflake's parameter hierarchy. Snowflake provides three types of parameters that can be set for an account: account parameters, session parameters, and object parameters. All parameters have default values, which can be set and then overridden at different levels depending on the parameter type. The following diagram illustrates the hierarchical relationship between the different parameter types and how individual parameters can be overridden at each level1:

As shown in the diagram, schema parameters are a type of object parameters that can be set for schemas. Schema parameters can override the account parameters that are set at the account level. For example, the LOG_LEVEL parameter can be set at the account level to control the logging level for all objects in the account, but it can also be overridden at the schema level to control the logging level for specific stored procedures and UDFs in that schema2.

The other options listed are not correct because they do not reflect the characteristics of Snowflake's parameter hierarchy. Session parameters do not override virtual warehouse parameters, because virtual warehouse parameters are a type of session parameters that can be set for virtual warehouses. Virtual warehouse parameters do not override user parameters, because user parameters are a type of session parameters that can be set for users. Table parameters do not override virtual warehouse parameters, because table parameters are a type of object parameters that can be set for tables, and object parameters do not affect session parameters1.

Reference:

Snowflake Documentation: Parameters

Snowflake Documentation: Setting Log Level

Question: 51

A Snowflake Architect is designing a multi-tenant application strategy for an organization in the Snowflake Data Cloud and is considering using an Account Per Tenant strategy.

Which requirements will be addressed with this approach? (Choose two.)

A. There needs to be fewer objects per tenant.
B. Security and Role-Based Access Control (RBAC) policies must be simple to configure.
C. Compute costs must be optimized.
D. Tenant data shape may be unique per tenant.
E. Storage costs must be optimized.

Answer: D, E

Explanation:

An Account Per Tenant strategy means creating a separate Snowflake account for each tenant (customer or business unit) of the multi-tenant application.

This approach has some advantages and disadvantages compared to other strategies, such as Database Per Tenant or Schema Per Tenant.

One advantage is that each tenant can have a unique data shape, meaning they can define their own tables, views, and other objects without affecting other tenants. This allows for more flexibility and customization for each tenant. Therefore, option D is correct.

Another advantage is that storage costs can be optimized, because each tenant can use their own storage credits and manage their own data retention policies. This also reduces the risk of data spillover or cross-tenant access. Therefore, option E is correct.

However, this approach also has some drawbacks, such as:

It requires more administrative overhead and complexity to manage multiple accounts and their resources.
It may not optimize compute costs, because each tenant has to provision their own warehouses and pay for their own compute credits. This may result in underutilization or overprovisioning of compute resources. Therefore, option C is incorrect.
It may not simplify security and RBAC policies, because each account has to define its own roles, users, and privileges. This may increase the risk of human errors or inconsistencies in security configurations. Therefore, option B is incorrect.
It may not reduce the number of objects per tenant, because each tenant still has to create their own databases, schemas, and other objects within their account. This may affect the performance and scalability of the application. Therefore, option A is incorrect.
Reference: : Multi-Tenant Application Strategies

Question: 52

An Architect has been asked to clone schema STAGING as it looked one week ago, Tuesday June 1st at 8:00 AM, to recover some objects.
The STAGING schema has 50 days of retention.
The Architect runs the following statement:
CREATE SCHEMA STAGING_CLONE CLONE STAGING at (timestamp => '2021-06-01 08:00:00');
The Architect receives the following error: Time travel data is not available for schema STAGING. The requested time is either beyond the allowed time travel period or before the object creation time. The Architect then checks the schema history and sees the following: CREATED_ON|NAME|DROPPED_ON 2021-06-02 23:00:00 | STAGING | NULL 2021-05-01 10:00:00 | STAGING | 2021-06-02 23:00:00 How can cloning the STAGING schema be achieved?

A. Undrop the STAGING schema and then rerun the CLONE statement.
B. Modify the statement: CREATE SCHEMA STAGING_CLONE CLONE STAGING at (timestamp => '202105-01 10:00:00');
C. Rename the STAGING schema and perform an UNDROP to retrieve the previous STAGING schema version, then run the CLONE statement.
D. Cloning cannot be accomplished because the STAGING schema version was not active during the proposed Time Travel time period.

Answer: C

Explanation:

The error message indicates that the schema STAGING does not have time travel data available for the requested timestamp, because the current version of the schema was created on 2021-06-02 23:00:00, which is after the timestamp of 2021-06-01 08:00:00. Therefore, the CLONE statement cannot access the historical data of the schema at that point in time.
Option A is incorrect, because undropping the STAGING schema will not restore the previous version of the schema that was active on 2021-06-01 08:00:00. Instead, it will create a new version of the schema with the same name and no data or objects.
Option B is incorrect, because modifying the timestamp to 2021-05-01 10:00:00 will not clone the schema as it looked one week ago, but as it looked when it was first created. This may not reflect the desired state of the schema and its objects.
Option C is correct, because renaming the STAGING schema and performing an UNDROP to retrieve the previous STAGING schema version will restore the schema that was dropped on 2021-06-02 23:00:00. This schema has time travel data available for the requested timestamp of 2021-06-01 08:00:00, and can be cloned using the CLONE

statement.

Option D is incorrect, because cloning can be accomplished by using the UNDROP command to access the previous version of the schema that was active during the proposed time travel period. Reference: : Cloning Considerations : Understanding & Using Time Travel : CREATE <object> ... CLONE

Question: 53

What are purposes for creating a storage integration? (Choose three.)

A. Control access to Snowflake data using a master encryption key that is maintained in the cloud provider's key management service.

B. Store a generated identity and access management (IAM) entity for an external cloud provider regardless of the cloud provider that hosts the Snowflake account.

C. Support multiple external stages using one single Snowflake object.

D. Avoid supplying credentials when creating a stage or when loading or unloading data.

E. Create private VPC endpoints that allow direct, secure connectivity between VPCs without traversing the public internet.

F. Manage credentials from multiple cloud providers in one single Snowflake object.

Answer: B, C, D

Explanation:

A storage integration is a Snowflake object that stores a generated identity and access management (IAM) entity for an external cloud provider, such as Amazon S3, Google Cloud Storage, or Microsoft Azure Blob Storage. This integration allows Snowflake to read data from and write data to an external storage location referenced in an external stage1.

One purpose of creating a storage integration is to support multiple external stages using one single Snowflake object. An integration can list buckets (and optional paths) that limit the locations users can specify when creating external stages that use the integration. Note that many external stage objects can reference different buckets and paths and use the same storage integration for authentication1. Therefore, option C is correct.

Another purpose of creating a storage integration is to avoid supplying credentials when creating a stage or when loading or unloading data. Integrations are named, first-class Snowflake objects that avoid the need for passing explicit cloud provider credentials such as secret keys or access tokens. Integration objects store an IAM user ID, and an administrator in your organization grants the IAM user permissions in the cloud provider account1. Therefore, option D is correct.

A third purpose of creating a storage integration is to store a generated IAM entity for an external cloud provider regardless of the cloud provider that hosts the Snowflake account. For example, you can create a storage integration for Amazon S3 even if your Snowflake account is hosted on Azure or Google Cloud Platform. This allows you to access data across different cloud platforms using Snowflake1. Therefore, option B is correct.

Option A is incorrect, because creating a storage integration does not control access to Snowflake data using a master encryption key. Snowflake encrypts all data using a hierarchical key model, and the master encryption key is managed by Snowflake or by the customer using a cloud provider's key management service. This is independent of the storage integration feature2.

Option E is incorrect, because creating a storage integration does not create private VPC endpoints. Private VPC endpoints are a network configuration option that allow direct, secure connectivity between VPCs without traversing the public internet. This is also independent of the storage integration feature3.

Option F is incorrect, because creating a storage integration does not manage credentials from multiple cloud providers

in one single Snowflake object. A storage integration is specific to one cloud provider, and you need to create separate integrations for each cloud provider you want to access4. Reference: : Encryption and Decryption : Private Link for Snowflake : CREATE STORAGE INTEGRATION : Option 1: Configuring a Snowflake Storage Integration to Access Amazon S3

Question: 54

A healthcare company is deploying a Snowflake account that may include Personal Health Information (PHI). The company must ensure compliance with all relevant privacy standards. Which best practice recommendations will meet data protection and compliance requirements? (Choose three.)

A. Use, at minimum, the Business Critical edition of Snowflake.
B. Create Dynamic Data Masking policies and apply them to columns that contain PHI.
C. Use the Internal Tokenization feature to obfuscate sensitive data.
D. Use the External Tokenization feature to obfuscate sensitive data.
E. Rewrite SQL queries to eliminate projections of PHI data based on current_role().
F. Avoid sharing data with partner organizations.

Answer: A, B, D

Explanation:

A healthcare company that handles PHI data must ensure compliance with relevant privacy standards, such as HIPAA, HITRUST, and GDPR. Snowflake provides several features and best practices to help customers meet their data protection and compliance requirements1.
One best practice recommendation is to use, at minimum, the Business Critical edition of Snowflake. This edition provides the highest level of data protection and security, including end-to- end encryption with customer-managed keys, enhanced object-level security, and HIPAA and HITRUST compliance2. Therefore, option A is correct.
Another best practice recommendation is to create Dynamic Data Masking policies and apply them to columns that contain PHI. Dynamic Data Masking is a feature that allows masking or redacting sensitive data based on the current user's role. This way, only authorized users can view the unmasked data, while others will see masked values, such as NULL, asterisks, or random characters3. Therefore, option B is correct.
A third best practice recommendation is to use the External Tokenization feature to obfuscate sensitive data. External Tokenization is a feature that allows replacing sensitive data with tokens that are generated and stored by an external service, such as Protegrity. This way, the original data is never stored or processed by Snowflake, and only authorized users can access the tokenized data through the external service4. Therefore, option D is correct.
Option C is incorrect, because the Internal Tokenization feature is not available in Snowflake. Snowflake does not provide any native tokenization functionality, but only supports integration with external tokenization services4.
Option E is incorrect, because rewriting SQL queries to eliminate projections of PHI data based on current_role() is not a best practice. This approach is error-prone, inefficient, and hard to maintain. A better alternative is to use Dynamic Data Masking policies, which can automatically mask data based on the user's role without modifying the queries3.
Option F is incorrect, because avoiding sharing data with partner organizations is not a best practice. Snowflake enables secure and governed data sharing with internal and external consumers, such as business units, customers, or partners. Data sharing does not involve copying or moving data, but only granting access privileges to the shared objects. Data sharing can also leverage Dynamic Data Masking and External Tokenization features to protect sensitive data5.
Reference: : Snowflake's Security & Compliance Reports : Snowflake Editions : Dynamic Data Masking : External Tokenization : Secure Data Sharing

Question: 55

There are two databases in an account, named fin_db and hr_db which contain payroll and employee data, respectively. Accountants and Analysts in the company require different permissions on the objects in these databases to perform their jobs. Accountants need read-write access to fin_db but only require read-only access to hr_db because the database is maintained by human resources personnel.

An Architect needs to create a read-only role for certain employees working in the human resources department. Which permission sets must be granted to this role?

A. USAGE on database hr_db, USAGE on all schemas in database hr_db, SELECT on all tables in database hr_db
B. USAGE on database hr_db, SELECT on all schemas in database hr_db, SELECT on all tables in database hr_db
C. MODIFY on database hr_db, USAGE on all schemas in database hr_db, USAGE on all tables in database hr_db
D. USAGE on database hr_db, USAGE on all schemas in database hr_db, REFERENCES on all tables in database hr_db

Answer: A

Explanation:
To create a read-only role for certain employees working in the human resources department, the role needs to have the following permissions on the hr_db database:
USAGE on the database: This allows the role to access the database and see its schemas and objects. USAGE on all schemas in the database: This allows the role to access the schemas and see their objects.
SELECT on all tables in the database: This allows the role to query the data in the tables.
Option A is the correct answer because it grants the minimum permissions required for a read-only role on the hr_db database.
Option B is incorrect because SELECT on schemas is not a valid permission. Schemas only support USAGE and CREATE permissions.
Option C is incorrect because MODIFY on the database is not a valid permission. Databases only support USAGE, CREATE, MONITOR, and OWNERSHIP permissions. Moreover, USAGE on tables is not sufficient for querying the data. Tables support SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, and OWNERSHIP permissions.
Option D is incorrect because REFERENCES on tables is not relevant for querying the data. REFERENCES permission allows the role to create foreign key constraints on the tables. Reference:
: https://docs.snowflake.com/en/user-guide/security-access-control-privileges.html#database- privileges
: https://docs.snowflake.com/en/user-guide/security-access-control-privileges.html#schema- privileges
: https://docs.snowflake.com/en/user-guide/security-access-control-privileges.html#table-privileges

Question: 56

An Architect runs the following SQL query:

```
SELECT
      METADATA$FILENAME,
      METADATA$FILE_ROW_NUMBER
FROM @FILEROWS/Food_Reviews.csv
            (file_format=CSV_N)
```

How can this query be interpreted?

A. FILEROWS is a stage. FILE_ROW_NUMBER is line number in file.
B. FILEROWS is the table. FILE_ROW_NUMBER is the line number in the table.
C. FILEROWS is a file. FILE_ROW_NUMBER is the file format location.

D. FILERONS is the file format location. FILE_ROW_NUMBER is a stage.

Answer: A

Explanation:

A stage is a named location in Snowflake that can store files for data loading and unloading. A stage can be internal or external, depending on where the files are stored.

The query in the question uses the LIST function to list the files in a stage named FILEROWS. The function returns a table with various columns, including FILE_ROW_NUMBER, which is the line number of the file in the stage.

Therefore, the query can be interpreted as listing the files in a stage named FILEROWS and showing the line number of each file in the stage.

Reference:

: Stages

: LIST Function

Question: 57

An Architect entered the following commands in sequence:

CREATE DATABASE SANDBOX;

CREATE ROLE INTERN;

CREATE TABLE SANDBOX.PUBLIC.AGENDA (ID INT, ITEMS STRING);

GRANT SELECT ON ALL TABLES IN SCHEMA SANDBOX. PUBLIC TO ROLE INTERNEGRANT ROLE INTERN TO USER USERI;

USER1 cannot find the table.

Which of the following commands does the Architect need to run for USER1 to find the tables using the Principle of Least Privilege? (Choose two.)

A. GRANT ROLE PUBLIC TO ROLE INTERN;

B. GRANT USAGE ON DATABASE SANDBOX TO ROLE INTERN;

C. GRANT USAGE ON SCHEMA SANDBOX.PUBLIC TO ROLE INTERN;

D. GRANT OWNERSHIP ON DATABASE SANDBOX TO USER INTERN;

E. GRANT ALL PRIVILEGES ON DATABASE SANDBOX TO ROLE INTERN;

Answer: B, C

Explanation:

According to the Principle of Least Privilege, the Architect should grant the minimum privileges necessary for the USER1 to find the tables in the SANDBOX database.

The USER1 needs to have USAGE privilege on the SANDBOX database and the SANDBOX.PUBLIC schema to be able to access the tables in the PUBLIC schema. Therefore, the commands B and C are the correct ones to run.

The command A is not correct because the PUBLIC role is automatically granted to every user and role in the account, and it does not have any privileges on the SANDBOX database by default.

The command D is not correct because it would transfer the ownership of the SANDBOX database from the Architect to the USER1, which is not necessary and violates the Principle of Least Privilege. The command E is not correct because it would grant all the possible privileges on the SANDBOX database to the USER1, which is also not necessary and violates the Principle of Least Privilege. Reference: : Snowflake - Principle of Least Privilege : Snowflake - Access Control Privileges : Snowflake - Public Role : Snowflake - Ownership and Grants

Question: 58

A DevOps team has a requirement for recovery of staging tables used in a complex set of data pipelines. The staging tables are all located in the same staging schem

a. One of the requirements is to have online recovery of data on a rolling 7-day basis.

After setting up the DATA_RETENTION_TIME_IN_DAYS at the database level, certain tables remain unrecoverable past 1 day.

What would cause this to occur? (Choose two.)

A. The staging schema has not been setup for MANAGED ACCESS.
B. The DATA_RETENTION_TIME_IN_DAYS for the staging schema has been set to 1 day.
C. The tables exceed the 1 TB limit for data recovery.
D. The staging tables are of the TRANSIENT type.
E. The DevOps role should be granted ALLOW_RECOVERY privilege on the staging schema.

Answer: B, D

Explanation:

The DATA_RETENTION_TIME_IN_DAYS parameter controls the Time Travel retention period for an object (database, schema, or table) in Snowflake. This parameter specifies the number of days for which historical data is preserved and can be accessed using Time Travel operations (SELECT, CREATE ... CLONE, UNDROP)1.

The requirement for recovery of staging tables on a rolling 7-day basis means that the DATA_RETENTION_TIME_IN_DAYS parameter should be set to 7 at the database level. However, this parameter can be overridden at the lower levels (schema or table) if they have a different value1. Therefore, one possible cause for certain tables to remain unrecoverable past 1 day is that the DATA_RETENTION_TIME_IN_DAYS for the staging schema has been set to 1 day. This would override the database level setting and limit the Time Travel retention period for all the tables in the schema to 1 day. To fix this, the parameter should be unset or set to 7 at the schema level1. Therefore, option B is correct.

Another possible cause for certain tables to remain unrecoverable past 1 day is that the staging tables are of the TRANSIENT type. Transient tables are tables that do not have a Fail-safe period and can have a Time Travel retention period of either 0 or 1 day. Transient tables are suitable for temporary or intermediate data that can be easily reproduced or replicated2. To fix this, the tables should be created as permanent tables, which can have a Time Travel retention period of up to 90 days1. Therefore, option D is correct.

Option A is incorrect because the MANAGED ACCESS feature is not related to the data recovery requirement. MANAGED ACCESS is a feature that allows granting access privileges to objects without explicitly granting the privileges to roles. It does not affect the Time Travel retention period or the data availability3.

Option C is incorrect because there is no 1 TB limit for data recovery in Snowflake. The data storage size does not affect the Time Travel retention period or the data availability4.

Option E is incorrect because there is no ALLOW_RECOVERY privilege in Snowflake. The privilege required to perform Time Travel operations is SELECT, which allows querying historical data in tables5.

Reference: : Understanding & Using Time Travel : Transient Tables : Managed Access : Understanding Storage Cost : Table Privileges

Question: 59

Consider the following COPY command which is loading data with CSV format into a Snowflake table from an internal stage through a data transformation query.

```
copy into home_sales(city, zip, sale_date, price)
from (select t.$1, t.$2, t.$6, t.$7 from @mystage/sales.csv.qz t) file_fcrmat =
```

```
format_name = mycsvformat
empty_field_as_null = true
field_optionally_enclosed_by = '"'

validation mode - return all errors
```

This command results in the following error:
SQL compilation error: invalid parameter 'validation_mode'
Assuming the syntax is correct, what is the cause of this error?

A. The VALIDATION_MODE parameter supports COPY statements that load data from external stages only.
B. The VALIDATION_MODE parameter does not support COPY statements with CSV file formats.
C. The VALIDATION_MODE parameter does not support COPY statements that transform data during a load.
D. The value return_all_errors of the option VALIDATION_MODE is causing a compilation error.

Answer: C

Explanation:

The VALIDATION_MODE parameter is used to specify the behavior of the COPY statement when loading data into a table. It is used to specify whether the COPY statement should return an error if any of the rows in the file are invalid or if it should continue loading the valid rows. The VALIDATION_MODE parameter is only supported for COPY statements that load data from external stages1.
The query in the question uses a data transformation query to load data from an internal stage. A data transformation query is a query that transforms the data during the load process, such as parsing JSON or XML data, applying functions, or joining with other tables2.
According to the documentation, VALIDATION_MODE does not support COPY statements that transform data during a load. If the parameter is specified, the COPY statement returns an error1. Therefore, option C is the correct answer.
Reference: : COPY INTO <table> : Transforming Data During a Load

Question: 60

A company is storing large numbers of small JSON files (ranging from 1-4 bytes) that are received from IoT devices and sent to a cloud provider. In any given hour, 100,000 files are added to the cloud provider.
What is the MOST cost-effective way to bring this data into a Snowflake table?

A. An external table
B. A pipe
C. A stream
D. A copy command at regular intervals

Answer: B

Explanation:

A pipe is a Snowflake object that continuously loads data from files in a stage (internal or external) into a table. A pipe can be configured to use auto-ingest, which means that Snowflake automatically detects new or modified files in the stage and loads them into the table without any manual intervention1.
A pipe is the most cost-effective way to bring large numbers of small JSON files into a Snowflake table, because it minimizes the number of COPY commands executed and the number of micropartitions created. A pipe can use file

aggregation, which means that it can combine multiple small files into a single larger file before loading them into the table. This reduces the load time and the storage cost of the data2.

An external table is a Snowflake object that references data files stored in an external location, such as Amazon S3, Google Cloud Storage, or Microsoft Azure Blob Storage. An external table does not store the data in Snowflake, but only provides a view of the data for querying. An external table is not a cost-effective way to bring data into a Snowflake table, because it does not support file aggregation, and it requires additional network bandwidth and compute resources to query the external data3.

A stream is a Snowflake object that records the history of changes (inserts, updates, and deletes) made to a table. A stream can be used to consume the changes from a table and apply them to another table or a task. A stream is not a way to bring data into a Snowflake table, but a way to process the data after it is loaded into a table4.

A copy command is a Snowflake command that loads data from files in a stage into a table. A copy command can be executed manually or scheduled using a task. A copy command is not a costeffective way to bring large numbers of small JSON files into a Snowflake table, because it does not support file aggregation, and it may create many micro-partitions that increase the storage cost of the data5.

Reference: : Pipes : Loading Data Using Snowpipe : External Tables : Streams : COPY INTO <table>

Question: 61

A company has a Snowflake account named ACCOUNTA in AWS us-east-1 region. The company stores its marketing data in a Snowflake database named MARKET_DB. One of the company's business partners has an account named PARTNERB in Azure East US 2 region. For marketing purposes the company has agreed to share the database MARKET_DB with the partner account.

Which of the following steps MUST be performed for the account PARTNERB to consume data from the MARKET_DB database?

A. Create a new account (called AZABC123) in Azure East US 2 region. From account ACCOUNTA create a share of database MARKET_DB, create a new database out of this share locally in AWS us- east-1 region, and replicate this new database to AZABC123 account. Then set up data sharing to the PARTNERB account.

B. From account ACCOUNTA create a share of database MARKET_DB, and create a new database out of this share locally in AWS us-east-1 region. Then make this database the provider and share it with the PARTNERB account.

C. Create a new account (called AZABC123) in Azure East US 2 region. From account ACCOUNTA replicate the database MARKET_DB to AZABC123 and from this account set up the data sharing to

the PARTNERB account.

D. Create a share of database MARKET_DB, and create a new database out of this share locally in AWS us-east-1 region. Then replicate this database to the partner's account PARTNERB.

Answer: C

Explanation:

Snowflake supports data sharing across regions and cloud platforms using account replication and share replication features. Account replication enables the replication of objects from a source account to one or more target accounts in the same organization. Share replication enables the replication of shares from a source account to one or more target accounts in the same organization1.

To share data from the MARKET_DB database in the ACCOUNTA account in AWS us-east-1 region with the PARTNERB account in Azure East US 2 region, the following steps must be performed: Create a new account (called AZABC123) in Azure East US 2 region. This account will act as a bridge between the source and the target accounts. The new account must be linked to the ACCOUNTA account using an organization2.

From the ACCOUNTA account, replicate the MARKET_DB database to the AZABC123 account using the account replication feature. This will create a secondary database in the AZABC123 account that is a replica of the primary database in the ACCOUNTA account3.
From the AZABC123 account, set up the data sharing to the PARTNERB account using the share replication feature. This will create a share of the secondary database in the AZABC123 account and grant access to the PARTNERB account. The PARTNERB account can then create a database from the share and query the data4.
Therefore, option C is the correct answer.
Reference: : Replicating Shares Across Regions and Cloud Platforms : Working with Organizations and Accounts : Replicating Databases Across Multiple Accounts : Replicating Shares Across Multiple Accounts

Question: 62

You are a snowflake architect in an organization. The business team came to to deploy an use case which requires you to load some data which they can visualize through tableau. Everyday new data comes in and the old data is no longer required.

What type of table you will use in this case to optimize cost

A. TRANSIENT
B. TEMPORARY
C. PERMANENT

Answer: A

Explanation:

A transient table is a type of table in Snowflake that does not have a Fail-safe period and can have a Time Travel retention period of either 0 or 1 day. Transient tables are suitable for temporary or intermediate data that can be easily reproduced or replicated1.
A temporary table is a type of table in Snowflake that is automatically dropped when the session ends or the current user logs out. Temporary tables do not incur any storage costs, but they are not visible to other users or sessions2.
A permanent table is a type of table in Snowflake that has a Fail-safe period and a Time Travel retention period of up to 90 days. Permanent tables are suitable for persistent and durable data that needs to be protected from accidental or malicious deletion3.
In this case, the use case requires loading some data that can be visualized through Tableau. The data is updated every day and the old data is no longer required. Therefore, the best type of table to use in this case to optimize cost is a transient table, because it does not incur any Fail-safe costs and it can have a short Time Travel retention period of 0 or 1 day. This way, the data can be loaded and queried by Tableau, and then deleted or overwritten without incurring any unnecessary storage costs.
Reference: : Transient Tables : Temporary Tables : Understanding & Using Time Travel

Question: 63

Following objects can be cloned in snowflake

A. Permanent table
B. Transient table

C. Temporary table
D. External tables
E. Internal stages

Answer: A, B, D

Explanation:

Snowflake supports cloning of various objects, such as databases, schemas, tables, stages, file formats, sequences, streams, tasks, and roles. Cloning creates a copy of an existing object in the system without copying the data or metadata. Cloning is also known as zero-copy cloning1.
Among the objects listed in the question, the following ones can be cloned in Snowflake: Permanent table: A permanent table is a type of table that has a Fail-safe period and a Time Travel retention period of up to 90 days. A permanent table can be cloned using the CREATE TABLE ... CLONE command2. Therefore, option A is correct.
Transient table: A transient table is a type of table that does not have a Fail-safe period and can have a Time Travel retention period of either 0 or 1 day. A transient table can also be cloned using the CREATE TABLE ... CLONE command2. Therefore, option B is correct.
External table: An external table is a type of table that references data files stored in an external location, such as Amazon S3, Google Cloud Storage, or Microsoft Azure Blob Storage. An external table can be cloned using the CREATE EXTERNAL TABLE ... CLONE command3. Therefore, option D is correct.
The following objects listed in the question cannot be cloned in Snowflake:
Temporary table: A temporary table is a type of table that is automatically dropped when the session ends or the current user logs out. Temporary tables do not support cloning4. Therefore, option C is incorrect.
Internal stage: An internal stage is a type of stage that is managed by Snowflake and stores files in Snowflake's internal cloud storage. Internal stages do not support cloning5. Therefore, option E is incorrect.
Reference: : Cloning Considerations : CREATE TABLE ... CLONE : CREATE EXTERNAL TABLE ... CLONE : Temporary Tables : Internal Stages

Question: 64

When loading data from stage using COPY INTO, what options can you specify for the ON_ERROR clause?

A. CONTINUE
B. SKIP_FILE
C. ABORT_STATEMENT
D. FAIL

Answer: A, B, C

Explanation:

The ON_ERROR clause is an optional parameter for the COPY INTO command that specifies the behavior of the command when it encounters errors in the files. The ON_ERROR clause can have one of the following values1:
CONTINUE: This value instructs the command to continue loading the file and return an error message for a maximum of one error encountered per data file. The difference between the ROWS_PARSED and ROWS_LOADED column values represents the number of rows that include detected errors. To view all errors in the data files, use the VALIDATION_MODE parameter or query the VALIDATE function1.
SKIP_FILE: This value instructs the command to skip the file when it encounters a data error on any of the records in the file. The command moves on to the next file in the stage and continues loading. The skipped file is not loaded and no

error message is returned for the file1.
ABORT_STATEMENT: This value instructs the command to stop loading data when the first error is encountered. The command returns an error message for the file and aborts the load operation. This is the default value for the ON_ERROR clause1.
Therefore, options A, B, and C are correct.
Reference: : COPY INTO <table>

Question: 65

Which of the below commands will use warehouse credits?

A. SHOW TABLES LIKE 'SNOWFL%';
B. SELECT MAX(FLAKE_ID) FROM SNOWFLAKE;
C. SELECT COUNT(*) FROM SNOWFLAKE;
D. SELECT COUNT(FLAKE_ID) FROM SNOWFLAKE GROUP BY FLAKE_ID;

Answer: B, C, D

Explanation:

Warehouse credits are used to pay for the processing time used by each virtual warehouse in Snowflake. A virtual warehouse is a cluster of compute resources that enables executing queries,

loading data, and performing other DML operations. Warehouse credits are charged based on the number of virtual warehouses you use, how long they run, and their size1.
Among the commands listed in the question, the following ones will use warehouse credits:
SELECT MAX(FLAKE_ID) FROM SNOWFLAKE: This command will use warehouse credits because it is a query that requires a virtual warehouse to execute. The query will scan the SNOWFLAKE table and return the maximum value of the FLAKE_ID column2. Therefore, option B is correct.
SELECT COUNT(*) FROM SNOWFLAKE: This command will also use warehouse credits because it is a query that requires a virtual warehouse to execute. The query will scan the SNOWFLAKE table and return the number of rows in the table3. Therefore, option C is correct.
SELECT COUNT(FLAKE_ID) FROM SNOWFLAKE GROUP BY FLAKE_ID: This command will also use warehouse credits because it is a query that requires a virtual warehouse to execute. The query will scan the SNOWFLAKE table and return the number of rows for each distinct value of the FLAKE_ID column4. Therefore, option D is correct.
The command that will not use warehouse credits is:
SHOW TABLES LIKE 'SNOWFL%': This command will not use warehouse credits because it is a metadata operation that does not require a virtual warehouse to execute. The command will return the names of the tables that match the pattern 'SNOWFL%' in the current database and schema5. Therefore, option A is incorrect.
Reference: : Understanding Compute Cost : MAX Function : COUNT Function : GROUP BY Clause : SHOW TABLES

Question: 66

What does a Snowflake Architect need to consider when implementing a Snowflake Connector for Kafka?

A. Every Kafka message is in JSON or Avro format.
B. The default retention time for Kafka topics is 14 days.
C. The Kafka connector supports key pair authentication, OAUTH. and basic authentication (for example, username and password).

D. The Kafka connector will create one table and one pipe to ingest data for each topic. If the connector cannot create the table or the pipe it will result in an exception.

Answer: D

Explanation:

The Snowflake Connector for Kafka is a Kafka Connect sink connector that reads data from one or more Apache Kafka topics and loads the data into a Snowflake table. The connector supports different authentication methods to connect to Snowflake, such as key pair authentication, OAUTH, and basic authentication (for example, username and password). The connector also supports different encryption methods, such as HTTPS and SSL1. The connector does not require that every Kafka message is in JSON or Avro format, as it can handle other formats such as CSV, XML, and Parquet2. The default retention time for Kafka topics is not relevant for the connector, as it only consumes the messages that are available in the topics and does not store them in Kafka. The connector will create one table and one pipe to ingest data for each topic by default, but this behavior can be customized by using the snowflake.topic2table.map configuration property3. If the connector cannot create the table or the pipe, it will log an error and retry the operation until it succeeds or the connector is stopped4. Reference:
Installing and Configuring the Kafka Connector
Overview of the Kafka Connector
Managing the Kafka Connector
Troubleshooting the Kafka Connector

Question: 67

A group of Data Analysts have been granted the role analyst role. They need a Snowflake database where they can create and modify tables, views, and other objects to load with their own dat a. The Analysts should not have the ability to give other Snowflake users outside of their role access to this data.
How should these requirements be met?

A. Grant ANALYST_R0LE OWNERSHIP on the database, but make sure that ANALYST_ROLE does not have the MANAGE GRANTS privilege on the account.

B. Grant SYSADMIN ownership of the database, but grant the create schema privilege on the database to the ANALYST_ROLE.

C. Make every schema in the database a managed access schema, owned by SYSADMIN, and grant create privileges on each schema to the ANALYST_ROLE for each type of object that needs to be created.

D. Grant ANALYST_ROLE ownership on the database, but grant the ownership on future [object type] s in database privilege to SYSADMIN.

Answer: A

Explanation:

Granting ANALYST_ROLE OWNERSHIP on the database allows the analysts to create and modify tables, views, and other objects within the database. However, to prevent the analysts from giving other Snowflake users outside of their role access to this data, the ANALYST_ROLE should not have the MANAGE GRANTS privilege on the account. The MANAGE GRANTS privilege enables a role to grant or revoke privileges on any object in the account, regardless of the ownership of the object1. Therefore, by removing this privilege from the ANALYST_ROLE, the analysts can only grant or revoke privileges on the objects that they own within the database, and not on any other objects in the account2.
The other options are not correct because:
B) Granting SYSADMIN ownership of the database and granting the create schema privilege on the database to the ANALYST_ROLE would allow the analysts to create schemas within the database, but not to create or modify tables,

views, or other objects within those schemas. The analysts would need to have the create [object type] privilege on each schema to create or modify objects within the schema3.

C) Making every schema in the database a managed access schema, owned by SYSADMIN, and granting create privileges on each schema to the ANALYST_ROLE for each type of object that needs to be created would allow the analysts to create and modify objects within the schemas, but not to grant or revoke privileges on those objects. A managed access schema is a schema that requires explicit grants for any access to the objects within the schema, regardless of the ownership of the objects4. Therefore, the analysts would need to have the grant privilege on each schema to grant or revoke privileges on the objects within the schema.

D) Granting ANALYST_ROLE ownership on the database and granting the ownership on future [object type] s in database privilege to SYSADMIN would allow the analysts to create and modify objects within the database, but also to grant or revoke privileges on those objects. The ownership on future [object type] s in database privilege enables a role to automatically become the owner of any new object of the specified type that is created in the database. Therefore, by granting this privilege to SYSADMIN, the analysts would not be able to prevent SYSADMIN from accessing or modifying the objects that they create within the database.

Reference:

1: MANAGE GRANTS Privilege | Snowflake Documentation
2: Access Control Privileges | Snowflake Documentation
3: CREATE SCHEMA | Snowflake Documentation
4: Managed Access | Snowflake Documentation
: GRANT | Snowflake Documentation
: Ownership on Future Objects | Snowflake Documentation
: Ownership and Revoking Privileges | Snowflake Documentation

Question: 68

What considerations need to be taken when using database cloning as a tool for data lifecycle management in a development environment? (Select TWO).

A. Any pipes in the source are not cloned.
B. Any pipes in the source referring to internal stages are not cloned.
C. Any pipes in the source referring to external stages are not cloned.
D. The clone inherits all granted privileges of all child objects in the source object, including the database.
E. The clone inherits all granted privileges of all child objects in the source object, excluding the database.

Answer: A D

Explanation:

Database cloning is a feature of Snowflake that allows creating a copy of a database, schema, table, or view without consuming any additional storage space. Database cloning can be used as a tool for data lifecycle management in a development environment, where developers and testers can work on isolated copies of production data without affecting the original data or each other1.

However, there are some considerations that need to be taken when using database cloning in a development environment, such as:

Any pipes in the source are not cloned. Pipes are objects that load data from a stage into a table continuously. Pipes are not cloned because they are associated with a specific stage and table, and cloning them would create duplicate data loading and potential conflicts2.

The clone inherits all granted privileges of all child objects in the source object, including the database. Privileges are the permissions that control the access and actions that can be performed on an object. When a database is cloned, the

clone inherits all the privileges that were granted on the source database and its child objects, such as schemas, tables, and views. This means that the same roles that can access and modify the source database can also access and modify the clone, unless the privileges are explicitly revoked or modified3.

The other options are not correct because:

B) Any pipes in the source referring to internal stages are not cloned. This is a subset of option A, which states that any pipes in the source are not cloned, regardless of the type of stage they refer to.

C) Any pipes in the source referring to external stages are not cloned. This is also a subset of option A) which states that any pipes in the source are not cloned, regardless of the type of stage they refer to.

E) The clone inherits all granted privileges of all child objects in the source object, excluding the database. This is incorrect, as the clone inherits all granted privileges of the source object, including the database.

Reference:

1: Database Cloning | Snowflake Documentation

2: Pipes | Snowflake Documentation

3: Access Control Privileges | Snowflake Documentation

Question: 69

Which columns can be included in an external table schema? (Select THREE).

A. VALUE

B. METADATASROW_ID

C. METADATASISUPDATE

D. METADAT A$ FILENAME

E. METADATAS FILE_ROW_NUMBER

F. METADATASEXTERNAL TABLE PARTITION

Answer: A, D E

Explanation:

An external table schema defines the columns and data types of the data stored in an external stage.

All external tables include the following columns by default:

VALUE: A VARIANT type column that represents a single row in the external file.

METADATA$FILENAME: A pseudocolumn that identifies the name of each staged data file included in the external table, including its path in the stage.

METADATA$FILE_ROW_NUMBER: A pseudocolumn that shows the row number for each record in a staged data file.

You can also create additional virtual columns as expressions using the VALUE column and/or the pseudocolumns.

However, the following columns are not valid for external tables and cannot be included in the schema:

METADATASROW_ID: This column is only available for internal tables and shows the unique identifier for each row in the table.

METADATASISUPDATE: This column is only available for internal tables and shows whether the row was inserted or updated by a merge operation.

METADATASEXTERNAL TABLE PARTITION: This column is not a valid column name and does not exist in Snowflake.

Reference: Introduction to External Tables, CREATE EXTERNAL TABLE

Question: 70

Which SQL alter command will MAXIMIZE memory and compute resources for a Snowpark stored procedure when

executed on the snowpark_opt_wh warehouse?

A.

```
alter warehouse snowpark_opt_wh set max_concurrency_level = 1
```

B.

```
alter warehouse snowpark_opt_wh set max_concurrency_level = 2;
```

C.

```
alter warehouse snowpark_opt_wh set max_concurrency_level = 8;
```

D.

```
alter warehouse snowpark_opt_wh set max_concurrency_level = 16;
```

Answer:A

Explanation:

To maximize memory and compute resources for a Snowpark stored procedure, you need to set the MAX_CONCURRENCY_LEVEL parameter for the warehouse that executes the stored procedure. This parameter determines the maximum number of concurrent queries that can run on a single warehouse. By setting it to 16, you ensure that the warehouse can use all the available CPU cores and memory on a single node, which is the optimal configuration for Snowpark-optimized warehouses. This will improve the performance and efficiency of the stored procedure, as it will not have to share resources with other queries or nodes. The other options are incorrect because they either do not change the MAX_CONCURRENCY_LEVEL parameter, or they set it to a lower value than 16, which will reduce the memory and compute resources for the stored procedure. Reference: [Snowpark-optimized Warehouses] 1 [Training Machine Learning Models with Snowpark Python] 2 [Snowflake Shorts: Snowpark Optimized Warehouses] 3

Question: 71

An Architect clones a database and all of its objects, including tasks. After the cloning, the tasks stop running. Why is this occurring?

A. Tasks cannot be cloned.
B. The objects that the tasks reference are not fully qualified.
C. Cloned tasks are suspended by default and must be manually resumed.
D. The Architect has insufficient privileges to alter tasks on the cloned database.

Answer: C

Explanation:

When a database is cloned, all of its objects, including tasks, are also cloned. However, cloned tasks are suspended by

default and must be manually resumed by using the ALTER TASK command. This is to prevent the cloned tasks from running unexpectedly or interfering with the original tasks.
Therefore, the reason why the tasks stop running after the cloning is because they are suspended by default (Option C).
Options A, B, and D are not correct because tasks can be cloned, the objects that the tasks reference are also cloned and do not need to be fully qualified, and the Architect does not need to alter the tasks on the cloned database, only resume them. Reference: The answer can be verified from Snowflake's official documentation on cloning and tasks available on their website. Here are some relevant links:
Cloning Objects | Snowflake Documentation
Tasks | Snowflake Documentation
ALTER TASK | Snowflake Documentation

Question: 72

Is it possible for a data provider account with a Snowflake Business Critical edition to share data with an Enterprise edition data consumer account?

A. A Business Critical account cannot be a data sharing provider to an Enterprise consumer. Any consumer accounts must also be Business Critical.
B. If a user in the provider account with role authority to create or alter share adds an Enterprise account as a consumer, it can import the share.
C. If a user in the provider account with a share owning role sets share_restrictions to False when adding an Enterprise consumer account, it can import the share.
D. If a user in the provider account with a share owning role which also has override share restrictions privilege share_restrictions set to False when adding an Enterprise consumer account, it can import the share.

Answer: D

Explanation:

Data sharing is a feature that allows Snowflake accounts to share data with each other without the need for data movement or copying1. Data sharing is enabled by creating shares, which are collections of database objects (tables, views, secure views, and secure UDFs) that can be accessed by other accounts, called consumers2.
By default, Snowflake does not allow sharing data from a Business Critical edition account to a nonBusiness Critical edition account. This is because Business Critical edition offers higher levels of data protection and encryption than other editions, and sharing data with lower editions may compromise the security and compliance of the data3.
However, Snowflake provides the OVERRIDE SHARE RESTRICTIONS global privilege, which allows a user to override the default restriction and share data from a Business Critical edition account to a non-Business Critical edition account. This privilege is granted to the ACCOUNTADMIN role by default, and can be granted to other roles as well4.
To enable data sharing from a Business Critical edition account to an Enterprise edition account, the following steps are required34:
A user in the provider account with the OVERRIDE SHARE RESTRICTIONS privilege must create or alter a share and add the Enterprise edition account as a consumer. The user must also set the share_restrictions parameter to False when adding the consumer. This parameter indicates whether the share is restricted to Business Critical edition accounts only. Setting it to False allows the share to be imported by lower edition accounts.
A user in the consumer account with the IMPORT SHARE privilege must import the share and grant access to the share objects to other roles in the account. The user must also set the share_restrictions parameter to False when importing the share. This parameter indicates whether the consumer account accepts shares from Business Critical edition accounts only. Setting it to False allows the consumer account to import shares from lower edition accounts.

Reference:

1: Introduction to Secure Data Sharing | Snowflake Documentation
2: Creating Secure Data Shares | Snowflake Documentation
3: Enable Data Share:Business Critical Account to Lower Edition | Medium
4: Enabling sharing from a Business critical account to a non-business … | Snowflake Documentation

Question: 73

Which of the following ingestion methods can be used to load near real-time data by using the messaging services provided by a cloud provider?

A. Snowflake Connector for Kafka
B. Snowflake streams
C. Snowpipe
D. Spark

Answer: A C

Explanation:

Snowflake Connector for Kafka and Snowpipe are two ingestion methods that can be used to load near real-time data by using the messaging services provided by a cloud provider. Snowflake Connector for Kafka enables you to stream structured and semi-structured data from Apache Kafka topics into Snowflake tables. Snowpipe enables you to load data from files that are continuously added to a cloud storage location, such as Amazon S3 or Azure Blob Storage. Both methods leverage Snowflake's micro-partitioning and columnar storage to optimize data ingestion and query performance. Snowflake streams and Spark are not ingestion methods, but rather components of the Snowflake architecture. Snowflake streams provide change data capture (CDC) functionality by tracking data changes in a table. Spark is a distributed computing framework that can be used to process large-scale data and write it to Snowflake using the Snowflake Spark Connector. Reference: Snowflake Connector for Kafka
Snowpipe
Snowflake Streams
Snowflake Spark Connector

Question: 74

An Architect is designing a file ingestion recovery solution. The project will use an internal named stage for file storage. Currently, in the case of an ingestion failure, the Operations team must
manually download the failed file and check for errors.

Which downloading method should the Architect recommend that requires the LEAST amount of operational overhead?

A. Use the Snowflake Connector for Python, connect to remote storage and download the file.
B. Use the get command in SnowSQL to retrieve the file.
C. Use the get command in Snowsight to retrieve the file.
D. Use the Snowflake API endpoint and download the file.

Answer: B

Explanation:

The get command in SnowSQL is a convenient way to download files from an internal stage to a local directory. The get command can be used in interactive mode or in a script, and it supports wildcards and parallel downloads. The get command also allows specifying the overwrite option, which determines how to handle existing files with the same name2

The Snowflake Connector for Python, the Snowflake API endpoint, and the get command in Snowsight are not recommended methods for downloading files from an internal stage, because they require more operational overhead than the get command in SnowSQL. The Snowflake Connector for Python and the Snowflake API endpoint require writing and maintaining code to handle the connection, authentication, and file transfer. The get command in Snowsight requires using the web interface and manually selecting the files to download34 Reference:

1: SnowPro Advanced: Architect | Study Guide
2: Snowflake Documentation | Using the GET Command
3: Snowflake Documentation | Using the Snowflake Connector for Python
4: Snowflake Documentation | Using the Snowflake API
: Snowflake Documentation | Using the GET Command in Snowsight
: SnowPro Advanced: Architect | Study Guide
: Using the GET Command
: Using the Snowflake Connector for Python
: Using the Snowflake API
: [Using the GET Command in Snowsight]

Question: 75

A table for IOT devices that measures water usage is created. The table quickly becomes large and contains more than 2 billion rows.

```
create table water_iot (
Uniqeld number#
Deviceld varchar(20)#
DeviceManufacturer varchar(50)
CustomerId varchar(20),
IOT_time stamp timestamp_ntz, City varchar(80)# Location
varchar(50) )
```

The general query patterns for the table are:

1. Deviceld, lOT_timestamp and Customerld are frequently used in the filter predicate for the select statement
2. The columns City and DeviceManuf acturer are often retrieved
3. There is often a count on Uniqueld

Which field(s) should be used for the clustering key?

A. lOT_timestamp
B. City and DeviceManuf acturer
C. Deviceld and Customerld
D. Uniqueld

Answer: C

Explanation:

A clustering key is a subset of columns or expressions that are used to co-locate the data in the same micro-partitions, which are the units of storage in Snowflake. Clustering can improve the performance of queries that filter on the clustering key columns, as it reduces the amount of data that needs to be scanned. The best choice for a clustering key depends on the query patterns and the data distribution in the table. In this case, the columns DeviceId, IOT_timestamp, and CustomerId are frequently used in the filter predicate for the select statement, which means they are good candidates for the clustering key. The columns City and DeviceManufacturer are often retrieved, but not filtered on, so they are not as important for the clustering key. The column UniqueId is used for counting, but it is not a good choice for the clustering key, as it is likely to have a high cardinality and a uniform distribution, which means it will not help to co-locate the data. Therefore, the best option is to use DeviceId and CustomerId as the clustering key, as they can help to prune the micropartitions and speed up the queries. Reference: Clustering Keys & Clustered Tables, Micro-partitions & Data Clustering, A Complete Guide to Snowflake Clustering

Question: 76

Which Snowflake objects can be used in a data share? (Select TWO).

A. Standard view
B. Secure view
C. Stored procedure
D. External table
E. Stream

Answer:A B

Data sharing is a feature that allows you to share selected objects in a database in your account with other Snowflake accounts. You can share the following Snowflake database objects: external tables, dynamic tables, secure views, secure materialized views, secure UDFs, and tables. However, not all of these objects can be used in a data share. A data share is a named object that encapsulates the information required to share a database. You can grant privileges on objects to a share either via a database role or directly to a share. The objects that can be granted privileges directly to a share are: standard views, secure views, secure UDFs, and tables. Therefore, the correct answer is A and B. The other options are incorrect because they cannot be granted privileges directly to a share. External tables, dynamic tables, and streams can only be shared via a database role. Stored procedures cannot be shared at all. Reference:
[Introduction to Secure Data Sharing] 1
[Working with Shares] 2
[Choosing How to Share Database Objects] 3

Question: 77

A company has an external vendor who puts data into Google Cloud Storage. The company's Snowflake account is set up in Azure.
What would be the MOST efficient way to load data from the vendor into Snowflake?

A. Ask the vendor to create a Snowflake account, load the data into Snowflake and create a data share.
B. Create an external stage on Google Cloud Storage and use the external table to load the data into Snowflake.
C. Copy the data from Google Cloud Storage to Azure Blob storage using external tools and load data from Blob storage to Snowflake.
D. Create a Snowflake Account in the Google Cloud Platform (GCP), ingest data into this account and use data replication to move the data from GCP to Azure.

Answer: B

Explanation:

The most efficient way to load data from the vendor into Snowflake is to create an external stage on Google Cloud Storage and use the external table to load the data into Snowflake (Option B). This way, you can avoid copying or moving the data across different cloud platforms, which can incur additional costs and latency. You can also leverage the external table feature to query the data directly from Google Cloud Storage without loading it into Snowflake tables, which can save storage space and improve performance. Option A is not efficient because it requires the vendor to create a Snowflake account and a data share, which can be complicated and costly. Option C is not efficient because it involves copying the data from Google Cloud Storage to Azure Blob storage using external tools, which can be slow and expensive. Option D is not efficient because it requires creating a Snowflake account in the Google Cloud Platform (GCP), ingesting data into this account, and using data replication to move the data from GCP to Azure, which can be complex and timeconsuming. Reference: The answer can be verified from Snowflake's official documentation on external stages and external tables available on their website. Here are some relevant links: Using External Stages | Snowflake Documentation

Using External Tables | Snowflake Documentation

Loading Data from a Stage | Snowflake Documentation

Question: 78

How can the Snowpipe REST API be used to keep a log of data load history?

A. Call insertReport every 20 minutes, fetching the last 10,000 entries.
B. Call loadHistoryScan every minute for the maximum time range.
C. Call insertReport every 8 minutes for a 10-minute time range.
D. Call loadHistoryScan every 10 minutes for a 15-minute time range.

Answer:D

Snowpipe is a service that automates and optimizes the loading of data from external stages into Snowflake tables. Snowpipe uses a queue to ingest files as they become available in the stage. Snowpipe also provides REST endpoints to load data and retrieve load history reports1. The loadHistoryScan endpoint returns the history of files that have been ingested by Snowpipe within a specified time range. The endpoint accepts the following parameters2: pipe: The fully-qualified name of the pipe to query.

startTimeInclusive: The start of the time range to query, in ISO 8601 format. The value must be within the past 14 days.

endTimeExclusive: The end of the time range to query, in ISO 8601 format. The value must be later than the start time and within the past 14 days.

recentFirst: A boolean flag that indicates whether to return the most recent files first or last. The default value is false, which means the oldest files are returned first.

showSkippedFiles: A boolean flag that indicates whether to include files that were skipped by Snowpipe in the response. The default value is false, which means only files that were loaded are returned.

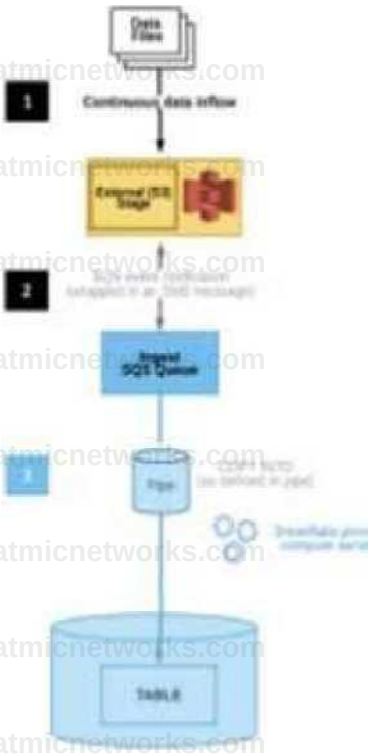
The loadHistoryScan endpoint can be used to keep a log of data load history by calling it periodically with a suitable time range. The best option among the choices is D, which is to call loadHistoryScan every 10 minutes for a 15-minute time range. This option ensures that the endpoint is called frequently enough to capture the latest files that have been ingested, and that the time range is wide enough to avoid missing any files that may have been delayed or retried by Snowpipe. The other options are either too infrequent, too narrow, or use the wrong endpoint3.

Reference:

1: Introduction to Snowpipe | Snowflake Documentation
2: loadHistoryScan | Snowflake Documentation
3: Monitoring Snowpipe Load History | Snowflake Documentation

Question: 79

The diagram shows the process flow for Snowpipe auto-ingest with Amazon Simple Notification Service (SNS) with the following steps:
Step 1: Data files are loaded in a stage.
Step 2: An Amazon S3 event notification, published by SNS, informs Snowpipe — by way of Amazon Simple Queue Service (SQS) - that files are ready to load. Snowpipe copies the files into a queue. Step 3: A Snowflake-provided virtual warehouse loads data from the queued files into the target table based on parameters defined in the specified pipe.



If an AWS Administrator accidentally deletes the SQS subscription to the SNS topic in Step 2, what will happen to the pipe that references the topic to receive event messages from Amazon S3?

A. The pipe will continue to receive the messages as Snowflake will automatically restore the subscription to the same SNS topic and will recreate the pipe by specifying the same SNS topic name in the pipe definition.
B. The pipe will no longer be able to receive the messages and the user must wait for 24 hours from the time when the SNS topic subscription was deleted. Pipe recreation is not required as the pipe will reuse the same subscription to the existing SNS topic after 24 hours.
C. The pipe will continue to receive the messages as Snowflake will automatically restore the subscription by creating a new SNS topic. Snowflake will then recreate the pipe by specifying the new SNS topic name in the pipe definition.
D. The pipe will no longer be able to receive the messages. To restore the system immediately, the user needs to manually create a new SNS topic with a different name and then recreate the pipe by specifying the new SNS topic name in the pipe definition.

Answer: D

Explanation:

If an AWS Administrator accidentally deletes the SQS subscription to the SNS topic in Step 2, the pipe that references the topic to receive event messages from Amazon S3 will no longer be able to receive the messages. This is because the SQS subscription is the link between the SNS topic and the Snowpipe notification channel. Without the subscription, the SNS topic will not be able to send notifications to the Snowpipe queue, and the pipe will not be triggered to load the new files. To restore the system immediately, the user needs to manually create a new SNS topic with a different name and then recreate the pipe by specifying the new SNS topic name in the pipe definition. This will create a new notification channel and a new SQS subscription for the pipe. Alternatively, the user can also recreate the SQS subscription to the existing SNS topic and then alter the pipe to use the same SNS topic name in the pipe definition. This will also restore the notification channel and the pipe functionality. Reference:
Automating Snowpipe for Amazon S3
Enabling Snowpipe Error Notifications for Amazon SNS
HowTo: Configuration steps for Snowpipe Auto-Ingest with AWS S3 Stages

Question: 80

An Architect needs to meet a company requirement to ingest files from the company's AWS storage accounts into the company's Snowflake Google Cloud Platform (GCP) account. How can the ingestion of these files into the company's Snowflake account be initiated? (Select TWO).

A. Configure the client application to call the Snowpipe REST endpoint when new files have arrived in Amazon S3 storage.
B. Configure the client application to call the Snowpipe REST endpoint when new files have arrived in Amazon S3 Glacier storage.
C. Create an AWS Lambda function to call the Snowpipe REST endpoint when new files have arrived in Amazon S3 storage.
D. Configure AWS Simple Notification Service (SNS) to notify Snowpipe when new files have arrived in Amazon S3 storage.
E. Configure the client application to issue a COPY INTO <TABLE> command to Snowflake when new files have arrived in Amazon S3 Glacier storage.

Answer: A, C

Explanation:

Snowpipe is a feature that enables continuous, near-real-time data ingestion from external sources into Snowflake tables. Snowpipe can ingest files from Amazon S3, Google Cloud Storage, or Azure Blob Storage into Snowflake tables on any cloud platform. Snowpipe can be triggered in two ways: by using the Snowpipe REST API or by using cloud notifications2
To ingest files from the company's AWS storage accounts into the company's Snowflake GCP account, the Architect can use either of these methods:
Configure the client application to call the Snowpipe REST endpoint when new files have arrived in Amazon S3 storage. This method requires the client application to monitor the S3 buckets for new files and send a request to the Snowpipe REST API with the list of files to ingest. The client application must also handle authentication, error handling, and retry logic3
Create an AWS Lambda function to call the Snowpipe REST endpoint when new files have arrived in Amazon S3 storage.

This method leverages the AWS Lambda service to execute a function that calls the Snowpipe REST API whenever an S3 event notification is received. The AWS Lambda function must be configured with the appropriate permissions, triggers, and code to invoke the Snowpipe REST API4

The other options are not valid methods for triggering Snowpipe:

Configure the client application to call the Snowpipe REST endpoint when new files have arrived in Amazon S3 Glacier storage. This option is not feasible because Snowpipe does not support ingesting files from Amazon S3 Glacier storage, which is a long-term archival storage service. Snowpipe only supports ingesting files from Amazon S3 standard storage classes5

Configure AWS Simple Notification Service (SNS) to notify Snowpipe when new files have arrived in Amazon S3 storage. This option is not applicable because Snowpipe does not support cloud notifications from AWS SNS. Snowpipe only supports cloud notifications from AWS SQS, Google Cloud Pub/Sub, or Azure Event Grid6

Configure the client application to issue a COPY INTO <TABLE> command to Snowflake when new files have arrived in Amazon S3 Glacier storage. This option is not relevant because it does not use Snowpipe, but rather the standard COPY command, which is a batch loading method. Moreover, the COPY command also does not support ingesting files from Amazon S3 Glacier storage7 Reference:

1: SnowPro Advanced: Architect | Study Guide 8
2: Snowflake Documentation | Snowpipe Overview 9
3: Snowflake Documentation | Using the Snowpipe REST API 10
4: Snowflake Documentation | Loading Data Using Snowpipe and AWS Lambda 11
5: Snowflake Documentation | Supported File Formats and Compression for Staged Data Files 12
6: Snowflake Documentation | Using Cloud Notifications to Trigger Snowpipe 13
7: Snowflake Documentation | Loading Data Using COPY into a Table
: SnowPro Advanced: Architect | Study Guide
: Snowpipe Overview
: Using the Snowpipe REST API
: Loading Data Using Snowpipe and AWS Lambda
: Supported File Formats and Compression for Staged Data Files
: Using Cloud Notifications to Trigger Snowpipe
: Loading Data Using COPY into a Table

Question: 81

When activating Tri-Secret Secure in a hierarchical encryption model in a Snowflake account, at what level is the customer-managed key used?

A. At the root level (HSM)
B. At the account level (AMK)
C. At the table level (TMK)
D. At the micro-partition level

Answer: B

Explanation:

Tri-Secret Secure is a feature that allows customers to use their own key, called the customermanaged key (CMK), in addition to the Snowflake-managed key, to create a composite master key that encrypts the data in Snowflake. The composite master key is also known as the account master key (AMK), as it is unique for each account and encrypts the table master keys (TMKs) that encrypt the file keys that encrypt the data files. The customer-managed key is used at the account level, not at the root level, the table level, or the micro-partition level. The root level is protected by a hardware security module (HSM), the table level is protected by the TMKs, and the micro-partition level is protected by the file keys12. Reference: Understanding Encryption Key Management in Snowflake
Tri-Secret Secure FAQ for Snowflake on AWS

Question: 82

What are characteristics of the use of transactions in Snowflake? (Select TWO).

A. Explicit transactions can contain DDL, DML, and query statements.
B. The autocommit setting can be changed inside a stored procedure.
C. A transaction can be started explicitly by executing a begin work statement and end explicitly by executing a commit work statement.
D. A transaction can be started explicitly by executing a begin transaction statement and end explicitly by executing an end transaction statement.
E. Explicit transactions should contain only DML statements and query statements. All DDL statements implicitly commit active transactions.

Answer: A, D

Explanation:

In Snowflake, a transaction is a sequence of SQL statements that are processed as an atomic unit. All statements in the transaction are either applied (i.e. committed) or undone (i.e. rolled back) together. Snowflake transactions guarantee ACID properties. A transaction can include both reads and writes1.
Explicit transactions are transactions that are started and ended explicitly by using the BEGIN TRANSACTION, COMMIT, and ROLLBACK statements. Snowflake supports the synonyms BEGIN WORK and BEGIN TRANSACTION, and COMMIT WORK and ROLLBACK WORK. Explicit transactions can contain DDL, DML, and query statements. However, explicit transactions should contain only DML statements and query statements, because DDL statements implicitly commit active transactions. This means that any changes made by the previous statements in the transaction are applied, and any changes made by the subsequent statements in the transaction are not part of the same transaction1.
The other options are not correct because:

B) The autocommit setting can be changed inside a stored procedure, but this does not affect the use of transactions in Snowflake. The autocommit setting determines whether each statement is executed in its own implicit transaction or not. If autocommit is enabled, each statement is committed automatically. If autocommit is disabled, each statement is executed in an implicit transaction until an explicit COMMIT or ROLLBACK is issued. Changing the autocommit setting inside a stored procedure only affects the statements within the stored procedure, and does not affect the

statements outside the stored procedure2.

C) A transaction can be started explicitly by executing a BEGIN WORK statement and end explicitly by executing a COMMIT WORK statement, but this is not a characteristic of the use of transactions in Snowflake. This is just one way of writing the statements that start and end an explicit transaction. Snowflake also supports the synonyms BEGIN TRANSACTION and COMMIT, which are recommended over BEGIN WORK and COMMIT WORK1.

D) A transaction can be started explicitly by executing a BEGIN TRANSACTION statement and end explicitly by executing an END TRANSACTION statement, but this is not a valid syntax in Snowflake. Snowflake does not support the END TRANSACTION statement. The correct way to end an explicit transaction is to use the COMMIT or ROLLBACK statement1.

Reference:

1: Transactions | Snowflake Documentation

2: AUTOCOMMIT | Snowflake Documentation

Question: 83

Which query will identify the specific days and virtual warehouses that would benefit from a multicluster warehouse to improve the performance of a particular workload?

A.

```
SELECT TO_DATE(START_TIME) AS DATE,
WAREHOUSE_NAME,
BYTES_SCANNED,
BYTES-SPILLED
FROM "SNOWFLAKE"."ACCOUNT_USAGE"."QUERY_HISTORY"
HAVING BYTES_SPILLED>BYTES_SCANNED;
```

B.

```
SELECT TO_DATE(START_TIME) AS DATE,
WAREHOUSE_NAME,
SUM(AVG_RUNNING) AS SUMRUNNING,
SUM(AVG_QUEUED_LOAD) AS SUM_QUEUED
FROM "SNOWFLAKE"."ACCOUNT-USAGE"."WAREHOUSE_LOAD_HISTORY"
GROUP BY 1,2
HAVING SUM(AVG_QUEUED_LOAD) > 0;
```

C.

```
SELECT TO-DATE (START_TIME) AS DATE,
WAPEHOUSE_NAME,
BYTES_S CANNED,
BYTES_SPILLED
FROM "SNOWFLAKE"."ACCOUNT_USAGE"."WAREHOUSE_LOAD_HISTORY" HAVING
BYTES_SPILLED>BYTES_SCANNED;
```

D.

```
SELECT TO_DATE(START_TIME) AS DATE,
```

```
WAREHOUSE_NAME,
BYTES_SPILLED_TO_LOCAL_STORAGE, SUM(AVG_QUEUED_LOAD) AS
SUM_QUEUED FROM ''SNOWFLAKE" . "ACCOUNT_USAGE" .
"QUERY_HISTORY" HAVING BYTES_SPILLED_TO_LOCAL_STORAGE >0;
```

Answer: C

Explanation:

A multi-cluster warehouse is a virtual warehouse that can scale compute resources by adding or removing clusters based on the workload demand. A multi-cluster warehouse can improve the performance of a particular workload by reducing the query queue time and the data spillage to local storage. To identify the specific days and virtual warehouses that would benefit from a multi-cluster warehouse, you need to analyze the query history and look for the following indicators:

High average queued load: This metric shows the average number of queries waiting in the queue for each warehouse cluster. A high value indicates that the warehouse is overloaded and cannot handle the concurrency demand.

High bytes spilled to local storage: This metric shows the amount of data that was spilled from memory to local disk during query processing. A high value indicates that the warehouse size is too small and cannot fit the data in memory.

High variation in workload: This metric shows the fluctuation in the number of queries submitted to the warehouse over time. A high variation indicates that the workload is unpredictable and dynamic, and requires a flexible scaling policy.

The query in option C is the best one to identify these indicators, as it selects the date, warehouse name, bytes spilled to local storage, and sum of average queued load from the query history table, and filters the results where bytes spilled to local storage is greater than zero. This query will show the days and warehouses that experienced data spillage and high queue time, and could benefit from a multi-cluster warehouse with auto-scale mode.

The query in option A is not correct, as it only selects the date and warehouse name, and does not include any metrics to measure the performance of the workload. The query in option B is not correct, as it selects the date, warehouse name, and average execution time, which is not a good indicator of the need for a multi-cluster warehouse. The query in option D is not correct, as it selects the date, warehouse name, and average credits used, which is not a good indicator of the need for a multi-cluster warehouse either.

Reference: Multi-cluster Warehouses, Query History View, Reducing Queues

Question: 84

A company is designing high availability and disaster recovery plans and needs to maximize redundancy and minimize recovery time objectives for their critical application processes. Cost is not a concern as long as the solution is the best available. The plan so far consists of the following steps: 1. Deployment of Snowflake accounts on two different cloud providers.

2. Selection of cloud provider regions that are geographically far apart.
3. The Snowflake deployment will replicate the databases and account data between both cloud provider accounts.
4. Implementation of Snowflake client redirect.

What is the MOST cost-effective way to provide the HIGHEST uptime and LEAST application disruption if there is a service event?

A. Connect the applications using the <organization_name>-<connection_name> URL. Use the Business Critical Snowflake edition.

B. Connect the applications using the <organization_name>-<connection_name> URL. Use the Virtual Private Snowflake (VPS) edition.

C. Connect the applications using the <organization_name>-<accountLocator> URL. Use the Enterprise Snowflake

edition.

D. Connect the applications using the <organization_name>-<accountLocator> URL. Use the Business Critical Snowflake edition.

Answer: D

Explanation:

To provide the highest uptime and least application disruption in case of a service event, the best option is to use the Business Critical Snowflake edition and connect the applications using the <organization_name>-<accountLocator> URL. The Business Critical Snowflake edition offers the highest level of security, performance, and availability for Snowflake accounts. It includes features such as customer-managed encryption keys, HIPAA compliance, and 4-hour RPO and RTO SLAs. It also supports account replication and failover across regions and cloud platforms, which enables business continuity and disaster recovery. By using the <organization_name>-<accountLocator> URL, the applications can leverage the Snowflake Client Redirect feature, which automatically redirects the client connections to the secondary account in case of a failover. This way, the applications can seamlessly switch to the backup account without any manual intervention or configuration changes. The other options are less cost-effective or less reliable because they either use a lower edition of Snowflake, which does not support account replication and failover, or they use the <organization_name>-<connection_name> URL, which does not support client redirect and requires manual updates to the connection string in case of a failover. Reference: [Snowflake Editions] 1 [Replication and Failover/Failback] 2 [Client Redirect] 3 [Snowflake Account Identifiers] 4

Question: 85

A company's Architect needs to find an efficient way to get data from an external partner, who is also a Snowflake user. The current solution is based on daily JSON extracts that are placed on an FTP server and uploaded to Snowflake manually. The files are changed several times each month, and the ingestion process needs to be adapted to accommodate these changes.
What would be the MOST efficient solution?

A. Ask the partner to create a share and add the company's account.

B. Ask the partner to use the data lake export feature and place the data into cloud storage where Snowflake can natively ingest it (schema-on-read).

C. Keep the current structure but request that the partner stop changing files, instead only appending new files.

D. Ask the partner to set up a Snowflake reader account and use that account to get the data for ingestion.

Answer: A

Explanation:

The most efficient solution is to ask the partner to create a share and add the company's account (Option A). This way, the company can access the live data from the partner without any data movement or manual intervention. Snowflake's secure data sharing feature allows data providers to share selected objects in a database with other Snowflake accounts. The shared data is read-only and does not incur any storage or compute costs for the data consumers. The data consumers can query the shared data directly or create local copies of the shared objects in their own databases. Option B is not efficient because it involves using the data lake export feature, which is intended for exporting data from Snowflake to an external data lake, not for importing data from another Snowflake account. The data lake export feature also requires the data provider to create an external stage on cloud storage and use the COPY INTO <location> command to export the data into parquet files. The data consumer then needs to create an external table or a file format to load the data from the cloud storage into Snowflake. This process can be complex and costly, especially if the

data changes frequently. Option C is not efficient because it does not solve the problem of manual data ingestion and adaptation. Keeping the current structure of daily JSON extracts on an FTP server and requesting the partner to stop changing files, instead only appending new files, does not improve the efficiency or reliability of the data ingestion process. The company still needs to upload the data to Snowflake manually and deal with any schema changes or data quality issues. Option D is not efficient because it requires the partner to set up a Snowflake reader account and use that account to get the data for ingestion. A reader account is a special type of account that can only consume data from the provider account that created it. It is intended for data consumers who are not Snowflake customers and do not have a licensing agreement with Snowflake. A reader account is not suitable for data ingestion from another Snowflake account, as it does not allow uploading, modifying, or unloading data. The company would need to use external tools or interfaces to access the data from the reader account and load it into their own account, which can be slow and expensive. Reference: The answer can be verified from Snowflake's official documentation on secure data sharing, data lake export, and reader accounts available on their website. Here are some relevant links:
Introduction to Secure Data Sharing | Snowflake Documentation
Data Lake Export Public Preview Is Now Available on Snowflake | Snowflake Blog Managing Reader Accounts | Snowflake Documentation

Question: 86

Refer to the exhibit.

Based on the architecture in the image, how can the data from DB1 be copied into TBL2? (Select TWO).

A.

```
use database DB1;
use schema SHI;
copy into DBi.SH2.TBL2
  from 0STAGE1
 file_format = (format_name = FF_PIPE_1);
```

B.

```
use database DB1;
use schema SH2;
copy into TBL2
  from 0STAGE1
 file_format = (format_name = FF_PIPE_1);
```

C.

```
use database DBi;
use schema SH2;
copy into DBi.SH2.TBL2
  from GSTAGEi
 file_format = (format_name = DB1.SH1.FF_PIPE_1);
```

D.

```
use database DB1;
use schema SH2;
copy into DB1.SH2.TBL2
  from GDB1.SH1.STAGE1
 file_format - (format_r.ame - FF_PIPE_1);
```

E.

```
use database DB1;
use schema SH2;
copy into TBL2
  from 0DB1.SH1.STAGE1
 file_format = (format_name = DB1.SH1.FF_PIPE_1);
```

Answer: B,E

Explanation:

The architecture in the image shows a Snowflake data platform with two databases, DB1 and DB2, and two schemas, SH1 and SH2. DB1 contains a table TBL1 and a stage STAGE1. DB2 contains a table TBL2. The image also shows a snippet of code written in SQL language that copies data from STAGE1 to TBL2 using a file format FF PIPE 1.

To copy data from DB1 to TBL2, there are two possible options among the choices given: Option B: Use a named external stage that references STAGE1. This option requires creating an external stage object in DB2.SH2 that points to the same location as STAGE1 in DB1.SH1. The external stage can be created using the CREATE STAGE command with the URL parameter specifying the location of STAGE11. For example:

SQLAI-generated code. Review and use carefully. More info on FAQ.

```
use database DB2;
use schema SH2;
create stage EXT_STAGE1 url = @DB1.SH1.STAGE1;
```

Then, the data can be copied from the external stage to TBL2 using the COPY INTO command with the FROM parameter specifying the external stage name and the FILE FORMAT parameter specifying the file format name2. For

example:

SQLAI-generated code. Review and use carefully. More info on FAQ.

```
copy into TBL2
from @EXT_STAGE1
file format = (format name = DB1.SH1.FF PIPE 1);
```

Option E: Use a cross-database query to select data from TBL1 and insert into TBL2. This option requires using the INSERT INTO command with the SELECT clause to query data from TBL1 in DB1.SH1 and insert it into TBL2 in DB2.SH2. The query must use the fully-qualified names of the tables, including the database and schema names3. For example:

SQLAI-generated code. Review and use carefully. More info on FAQ.

```
use database DB2;
use schema SH2;
insert into TBL2
select * from DB1.SH1.TBL1;
```

The other options are not valid because:

Option A: It uses an invalid syntax for the COPY INTO command. The FROM parameter cannot specify a table name, only a stage name or a file location2.

Option C: It uses an invalid syntax for the COPY INTO command. The FILE FORMAT parameter cannot specify a stage name, only a file format name or options2.

Option D: It uses an invalid syntax for the CREATE STAGE command. The URL parameter cannot specify a table name, only a file location1.

Reference:

1: CREATE STAGE | Snowflake Documentation
2: COPY INTO table | Snowflake Documentation
3: Cross-database Queries | Snowflake Documentation

Question: 87

Why might a Snowflake Architect use a star schema model rather than a 3NF model when designing a data architecture to run in Snowflake? (Select TWO).

A. Snowflake cannot handle the joins implied in a 3NF data model.
B. The Architect wants to remove data duplication from the data stored in Snowflake.
C. The Architect is designing a landing zone to receive raw data into Snowflake.
D. The Bl tool needs a data model that allows users to summarize facts across different dimensions, or to drill down from the summaries.
E. The Architect wants to present a simple flattened single view of the data to a particular group of end users.

Answer: D E

Explanation:

A star schema model is a type of dimensional data model that consists of a single fact table and multiple dimension tables. A 3NF model is a type of relational data model that follows the third normal form, which eliminates data redundancy and ensures referential integrity. A Snowflake Architect might use a star schema model rather than a 3NF model when designing a data architecture to run in Snowflake for the following reasons:

A star schema model is more suitable for analytical queries that require aggregating and slicing data across different dimensions, such as those performed by a BI tool. A 3NF model is more suitable for transactional queries that require inserting, updating, and deleting individual records.

A star schema model is simpler and faster to query than a 3NF model, as it involves fewer joins and less complex SQL statements. A 3NF model is more complex and slower to query, as it involves more joins and more complex SQL

statements.

A star schema model can provide a simple flattened single view of the data to a particular group of end users, such as business analysts or data scientists, who need to explore and visualize the data. A 3NF model can provide a more detailed and normalized view of the data to a different group of end users, such as application developers or data engineers, who need to maintain and update the data. The other options are not valid reasons for choosing a star schema model over a 3NF model in Snowflake:

Snowflake can handle the joins implied in a 3NF data model, as it supports ANSI SQL and has a powerful query engine that can optimize and execute complex queries efficiently.

The Architect can use both star schema and 3NF models to remove data duplication from the data stored in Snowflake, as both models can enforce data integrity and avoid data anomalies. However, the trade-off is that a star schema model may have more data redundancy than a 3NF model, as it denormalizes the data for faster query performance, while a 3NF model may have less data redundancy than a star schema model, as it normalizes the data for easier data maintenance.

The Architect can use both star schema and 3NF models to design a landing zone to receive raw data into Snowflake, as both models can accommodate different types of data sources and formats. However, the choice of the model may depend on the purpose and scope of the landing zone, such as whether it is a temporary or permanent storage, whether it is a staging area or a data lake, and whether it is a single source or a multi-source integration.

Reference:

Snowflake Architect Training

Data Modeling: Understanding the Star and Snowflake Schemas

Data Vault vs Star Schema vs Third Normal Form: Which Data Model to Use?

Star Schema vs Snowflake Schema: 5 Key Differences

Dimensional Data Modeling - Snowflake schema

Star schema vs Snowflake Schema

Question: 88

An Architect is troubleshooting a query with poor performance using the QUERY_HISTORY function. The Architect observes that the COMPILATIONJHME is greater than the EXECUTIONJTIME.

What is the reason for this?

A. The query is processing a very large dataset.
B. The query has overly complex logic.
C. The query is queued for execution.
D. The query is reading from remote storage.

Answer: B

Explanation:

Compilation time is the time it takes for the optimizer to create an optimal query plan for the efficient execution of the query. It also involves some pruning of partition files, making the query execution efficient2

If the compilation time is greater than the execution time, it means that the optimizer spent more time analyzing the query than actually running it. This could indicate that the query has overly complex logic, such as multiple joins, subqueries, aggregations, or expressions. The complexity of the query could also affect the size and quality of the query plan, which could impact the performance of the query3

To reduce the compilation time, the Architect can try to simplify the query logic, use views or common table expressions (CTEs) to break down the query into smaller parts, or use hints to guide the optimizer. The Architect can

also use the EXPLAIN command to examine the query plan and identify potential bottlenecks or inefficiencies4

Reference:

1: SnowPro Advanced: Architect | Study Guide 5

2: Snowflake Documentation | Query Profile Overview 6

3: Understanding Why Compilation Time in Snowflake Can Be Higher than Execution Time 7

4: Snowflake Documentation | Optimizing Query Performance 8

: SnowPro Advanced: Architect | Study Guide

: Query Profile Overview

: Understanding Why Compilation Time in Snowflake Can Be Higher than Execution Time

: Optimizing Query Performance

Question: 89

A Snowflake Architect is designing a multiple-account design strategy.
This strategy will be MOST cost-effective with which scenarios? (Select TWO).

A. The company wants to clone a production database that resides on AWS to a development database that resides on Azure.

B. The company needs to share data between two databases, where one must support Payment Card Industry Data Security Standard (PCI DSS) compliance but the other one does not.

C. The company needs to support different role-based access control features for the development, test, and production environments.

D. The company security policy mandates the use of different Active Directory instances for the development, test, and production environments.

E. The company must use a specific network policy for certain users to allow and block given IP addresses.

Answer: B, C

Explanation:

A multiple-account design strategy is a way of organizing Snowflake accounts into logical groups based on different criteria, such as cloud provider, region, environment, or business unit. A multipleaccount design strategy can help achieve various goals, such as cost optimization, performance isolation, security compliance, and data sharing1. In this question, the scenarios that would be most cost-effective with a multiple-account design strategy are:

The company wants to clone a production database that resides on AWS to a development database that resides on Azure. This scenario would benefit from a multiple-account design strategy because it would allow the company to leverage the cross-cloud replication feature of Snowflake, which enables replicating databases across different cloud platforms and regions. This feature can help reduce the data transfer costs and latency, as well as provide high availability and disaster recovery2. The company security policy mandates the use of different Active Directory instances for the development, test, and production environments. This scenario would benefit from a multiple account design strategy because it would allow the company to use different federated authentication methods for each environment, and integrate them with different Active Directory instances. This can help improve the security and governance of the access to the Snowflake accounts, as well as simplify the user management and provisioning3.

The other scenarios would not be most cost-effective with a multiple-account design strategy, because:

The company needs to share data between two databases, where one must support Payment Card Industry Data Security Standard (PCI DSS) compliance but the other one does not. This scenario can be handled within a single Snowflake account, by using secure views and secure UDFs to mask or filter the sensitive data, and applying the appropriate roles and privileges to the users who access the data. This can help achieve the PCI DSS compliance without incurring the additional costs of managing multiple accounts4.

The company needs to support different role-based access control features for the development, test, and production environments. This scenario can also be handled within a single Snowflake account, by using the native role-based access control (RBAC) features of Snowflake, such as roles, grants, and privileges, to define different access levels and permissions for each environment. This can help ensure the security and integrity of the data and the objects, as well as the separation of duties and responsibilities among the users.

The company must use a specific network policy for certain users to allow and block given IP addresses. This scenario can also be handled within a single Snowflake account, by using the network policy feature of Snowflake, which enables creating and applying network policies to restrict the IP addresses that can access the Snowflake account. This can help prevent unauthorized access and protect the data from malicious attacks.

Reference:

Designing Your Snowflake Topology

Cross-Cloud Replication

Configuring Federated Authentication and SSO

Using Secure Views and Secure UDFs to Comply with PCI DSS [Understanding Access Control in Snowflake] [Network Policies]

Question: 90

The following table exists in the production database:

A regulatory requirement states that the company must mask the username for events that are older than six months based on the current date when the data is queried.

How can the requirement be met without duplicating the event data and making sure it is applied when creating views using the table or cloning the table?

A. Use a masking policy on the username column using a entitlement table with valid dates.
B. Use a row level policy on the user_events table using a entitlement table with valid dates.
C. Use a masking policy on the username column with event_timestamp as a conditional column.
D. Use a secure view on the user_events table using a case statement on the username column.

Answer: C

Explanation:

A masking policy is a feature of Snowflake that allows masking sensitive data in query results based on the role of the user and the condition of the data. A masking policy can be applied to a column in a table or a view, and it can use another column in the same table or view as a conditional column. A conditional column is a column that determines whether the masking policy is applied or not based on its value1.

In this case, the requirement can be met by using a masking policy on the username column with event_timestamp as a conditional column. The masking policy can use a function that masks the username if the event_timestamp is older than six months based on the current date, and returns the original username otherwise. The masking policy can be applied to the user_events table, and it will also be applied when creating views using the table or cloning the table2.

The other options are not correct because:

A) Using a masking policy on the username column using an entitlement table with valid dates would require creating another table that stores the valid dates for each username, and joining it with the user_events table in the masking policy function. This would add complexity and overhead to the masking policy, and it would not use the event_timestamp column as the condition for masking. B) Using a row level policy on the user_events table using an entitlement table with valid dates would require creating another table that stores the valid dates for each username, and joining it with the user_events table in the row access policy function. This would filter out the rows that have event_timestamp older than six months based on the valid dates, instead of masking the username column. This would not meet the requirement of masking the username, and it would also reduce the

visibility of the event data.

D) Using a secure view on the user_events table using a case statement on the username column would require creating a view that uses a case expression to mask the username column based on the event_timestamp column. This would meet the requirement of masking the username, but it would not be applied when cloning the table. A secure view is a view that prevents the underlying data from being exposed by queries on the view. However, a secure view does not prevent the underlying data from being exposed by cloning the table3.

Reference:

1: Masking Policies | Snowflake Documentation

2: Using Conditional Columns in Masking Policies | Snowflake Documentation

3: Secure Views | Snowflake Documentation

Question: 91

Which data models can be used when modeling tables in a Snowflake environment? (Select THREE).

A. Graph model
B. Dimensional/Kimball
C. Data lake
D. Inmon/3NF
E. Bayesian hierarchical model
F. Data vault

Answer: B, D, F

Explanation:

Snowflake is a cloud data platform that supports various data models for modeling tables in a Snowflake environment. The data models can be classified into two categories: dimensional and normalized. Dimensional data models are designed to optimize query performance and ease of use for business intelligence and analytics. Normalized data models are designed to reduce data redundancy and ensure data integrity for transactional and operational systems.

The following are some of the data models that can be used in Snowflake:

Dimensional/Kimball: This is a popular dimensional data model that uses a star or snowflake schema to organize data into fact and dimension tables. Fact tables store quantitative measures and foreign keys to dimension tables. Dimension tables store descriptive attributes and hierarchies. A star schema has a single denormalized dimension table for each dimension, while a snowflake schema has multiple normalized dimension tables for each dimension. Snowflake supports both star and snowflake schemas, and allows users to create views and joins to simplify queries.

Inmon/3NF: This is a common normalized data model that uses a third normal form (3NF) schema to organize data into entities and relationships. 3NF schema eliminates data duplication and ensures data consistency by applying three rules: 1) every column in a table must depend on the primary key,

2) every column in a table must depend on the whole primary key, not a part of it, and 3) every column in a table must depend only on the primary key, not on other columns. Snowflake supports 3NF schema and allows users to create referential integrity constraints and foreign key relationships to enforce data quality.

Data vault: This is a hybrid data model that combines the best practices of dimensional and normalized data models to create a scalable, flexible, and resilient data warehouse. Data vault schema consists of three types of tables: hubs, links, and satellites. Hubs store business keys and metadata for each entity. Links store associations and relationships between entities. Satellites store descriptive attributes and historical changes for each entity or relationship. Snowflake supports data vault schema and allows users to leverage its features such as time travel, zero-copy cloning, and secure data sharing to implement data vault methodology.

Reference: What is Data Modeling? | Snowflake, Snowflake Schema in Data Warehouse Model - GeeksforGeeks, [Data Vault 2.0 Modeling with Snowflake]

Question: 92

A Snowflake Architect is setting up database replication to support a disaster recovery plan. The primary database has external tables.
How should the database be replicated?

A. Create a clone of the primary database then replicate the database.
B. Move the external tables to a database that is not replicated, then replicate the primary database.
C. Replicate the database ensuring the replicated database is in the same region as the external tables.
D. Share the primary database with an account in the same region that the database will be replicated to.

Answer: B

Explanation:

Database replication is a feature that allows you to create a copy of a database in another account, region, or cloud platform for disaster recovery or business continuity purposes. However, not all database objects can be replicated. External tables are one of the exceptions, as they reference data files stored in an external stage that is not part of Snowflake. Therefore, to replicate a database that contains external tables, you need to move the external tables to a separate database that is not replicated, and then replicate the primary database that contains the other objects. This way, you can avoid replication errors and ensure consistency between the primary and secondary databases. The other options are incorrect because they either do not address the issue of external tables, or they use an alternative method that is not supported by Snowflake. You cannot create a clone of the primary database and then replicate it, as replication only works on the original database, not on its clones. You also cannot share the primary database with another account, as sharing is a different feature that does not create a copy of the database, but rather grants access to the shared objects. Finally, you do not need to ensure that the replicated database is in the same region as the external tables, as external tables can access data files stored in any region or cloud platform, as long as the stage URL is valid and accessible. Reference: [Replication and Failover/Failback] 1 [Introduction to External Tables] 2 [Working with External Tables] 3

[Replication : How to migrate an account from One Cloud Platform or Region to another in Snowflake] 4

Question: 93

An Architect is integrating an application that needs to read and write data to Snowflake without installing any additional software on the application server.
How can this requirement be met?

A. Use SnowSQL.
B. Use the Snowpipe REST API.
C. Use the Snowflake SQL REST API.
D. Use the Snowflake ODBC driver.

Answer: C

Explanation:

The Snowflake SQL REST API is a REST API that you can use to access and update data in a Snowflake database. You can use this API to execute standard queries and most DDL and DML statements. This API can be used to develop custom

applications and integrations that can read and write data to Snowflake without installing any additional software on the application server. Option A is not correct because SnowSQL is a command-line client that requires installation and configuration on the application server. Option B is not correct because the Snowpipe REST API is used to load data from cloud storage into Snowflake tables, not to read or write data to Snowflake. Option D is not correct because the Snowflake ODBC driver is a software component that enables applications to connect to Snowflake using the ODBC protocol, which also requires installation and configuration on the application server. Reference: The answer can be verified from Snowflake's official documentation on the Snowflake SQL REST API available on their website. Here are some relevant links:

Snowflake SQL REST API | Snowflake Documentation

Introduction to the SQL API | Snowflake Documentation

Submitting a Request to Execute SQL Statements | Snowflake Documentation

Question: 94

What transformations are supported in the below SQL statement? (Select THREE).

CREATE PIPE ... AS COPY ... FROM (...)

A. Data can be filtered by an optional where clause.
B. Columns can be reordered.
C. Columns can be omitted.
D. Type casts are supported.
E. Incoming data can be joined with other tables.
F. The ON ERROR - ABORT statement command can be used.

Answer: A, B, C

Explanation:

The SQL statement is a command for creating a pipe in Snowflake, which is an object that defines the COPY INTO <table> statement used by Snowpipe to load data from an ingestion queue into tables1. The statement uses a subquery in the FROM clause to transform the data from the staged files before loading it into the table2.

The transformations supported in the subquery are as follows2:

Data can be filtered by an optional WHERE clause, which specifies a condition that must be satisfied by the rows returned by the subquery. For example:

SQLAI-generated code. Review and use carefully. More info on FAQ.

```
create pipe mypipe as
copy into mytable
from (
  select * from @mystage
  where col1 = 'A' and col2 > 10
);
```

Columns can be reordered, which means changing the order of the columns in the subquery to match the order of the columns in the target table. For example: SQLAI-generated code. Review and use carefully. More info on FAQ.

```
create pipe mypipe as
copy into mytable (col1, col2, col3)
from (
  select col3, col1, col2 from @mystage
);
```

Columns can be omitted, which means excluding some columns from the subquery that are not needed in the target table. For example:

SQLAI-generated code. Review and use carefully. More info on FAQ.

```
create pipe mypipe as
copy into mytable (col1, col2)
from (
  select col1, col2 from @mystage
);
```

The other options are not supported in the subquery because2:

Type casts are not supported, which means changing the data type of a column in the subquery. For example, the following statement will cause an error:

SQLAI-generated code. Review and use carefully. More info on FAQ.

```
create pipe mypipe as
copy into mytable (col1, col2)
from (
  select col1::date, col2 from @mystage
);
```

Incoming data can not be joined with other tables, which means combining the data from the staged files with the data from another table in the subquery. For example, the following statement will cause an error:

SQLAI-generated code. Review and use carefully. More info on FAQ.

```
create pipe mypipe as
copy into mytable (col1, col2, col3)
from (
  select s.col1, s.col2, t.col3 from @mystage s
  join othertable t on s.col1 = t.col1
);
```

The ON ERROR - ABORT statement command can not be used, which means aborting the entire load operation if any error occurs. This command can only be used in the COPY INTO <table> statement, not in the subquery. For example, the following statement will cause an error: SQLAI-generated code. Review and use carefully. More info on FAQ. create pipe mypipe as copy into mytable from (

```
  select * from @mystage on error abort
);
```

Reference:

1: CREATE PIPE | Snowflake Documentation

2: Transforming Data During a Load | Snowflake Documentation

Question: 95

Data is being imported and stored as JSON in a VARIANT column. Query performance was fine, but most recently, poor query performance has been reported.

What could be causing this?

A. There were JSON nulls in the recent data imports.

B. The order of the keys in the JSON was changed.

C. The recent data imports contained fewer fields than usual.

D. There were variations in string lengths for the JSON values in the recent data imports.

Answer: B D

Explanation:

Data is being imported and stored as JSON in a VARIANT column. Query performance was fine, but most recently, poor

query performance has been reported. This could be caused by the following factors:
The order of the keys in the JSON was changed. Snowflake stores semi-structured data internally in a column-like structure for the most common elements, and the remainder in a leftovers-like column. The order of the keys in the JSON affects how Snowflake determines the common elements and how it optimizes the query performance. If the order of the keys in the JSON was changed, Snowflake might have to re-parse the data and re-organize the internal storage, which could result in slower query performance.
There were variations in string lengths for the JSON values in the recent data imports. Non-native values, such as dates and timestamps, are stored as strings when loaded into a VARIANT column. Operations on these values could be slower and also consume more space than when stored in a relational column with the corresponding data type. If there were variations in string lengths for the JSON values in the recent data imports, Snowflake might have to allocate more space and perform more conversions, which could also result in slower query performance.
The other options are not valid causes for poor query performance:
There were JSON nulls in the recent data imports. Snowflake supports two types of null values in semi-structured data: SQL NULL and JSON null. SQL NULL means the value is missing or unknown, while JSON null means the value is explicitly set to null. Snowflake can distinguish between these
two types of null values and handle them accordingly. Having JSON nulls in the recent data imports should not affect the query performance significantly.
The recent data imports contained fewer fields than usual. Snowflake can handle semi-structured data with varying schemas and fields. Having fewer fields than usual in the recent data imports should not affect the query performance significantly, as Snowflake can still optimize the data ingestion and query execution based on the existing fields.
Reference:
Considerations for Semi-structured Data Stored in VARIANT
Snowflake Architect Training
Snowflake query performance on unique element in variant column
Snowflake variant performance

Question: 96

What step will improve the performance of queries executed against an external table?

A. Partition the external table.
B. Shorten the names of the source files.
C. Convert the source files' character encoding to UTF-8.
D. Use an internal stage instead of an external stage to store the source files.

Answer: A

Explanation:

Partitioning an external table is a technique that improves the performance of queries executed against the table by reducing the amount of data scanned. Partitioning an external table involves creating one or more partition columns that define how the table is logically divided into subsets of data based on the values in those columns. The partition columns can be derived from the file metadata (such as file name, path, size, or modification time) or from the file content (such as a column value or a JSON attribute). Partitioning an external table allows the query optimizer to prune the files that do not match the query predicates, thus avoiding unnecessary data scanning and processing2
The other options are not effective steps for improving the performance of queries executed against an external table:
Shorten the names of the source files. This option does not have any impact on the query performance, as the file

names are not used for query processing. The file names are only used for creating the external table and displaying the query results3

Convert the source files' character encoding to UTF-8. This option does not affect the query performance, as Snowflake supports various character encodings for external table files, such as UTF- 8, UTF-16, UTF-32, ISO-8859-1, and Windows-1252. Snowflake automatically detects the character encoding of the files and converts them to UTF-8 internally for query processing4 Use an internal stage instead of an external stage to store the source files. This option is not applicable, as external tables can only reference files stored in external stages, such as Amazon S3, Google Cloud Storage, or Azure Blob Storage. Internal stages are used for loading data into internal tables, not external tables5

Reference:

1: SnowPro Advanced: Architect | Study Guide

2: Snowflake Documentation | Partitioning External Tables

3: Snowflake Documentation | Creating External Tables

4: Snowflake Documentation | Supported File Formats and Compression for Staged Data Files

5: Snowflake Documentation | Overview of Stages

: SnowPro Advanced: Architect | Study Guide

: Partitioning External Tables

: Creating External Tables

: Supported File Formats and Compression for Staged Data Files

: Overview of Stages

Question: 97

The Business Intelligence team reports that when some team members run queries for their dashboards in parallel with others, the query response time is getting significantly slower What can a Snowflake Architect do to identify what is occurring and troubleshoot this issue?

A.

Use larger warehouses to speed up the queries running In parallel Identify the queries running in parallel using this query

```
SELECT QUERYJD
,USEf:_MAME
,WAREHOUSE-NAME
,WAREHOUSE-SIZE
.BYTES_SCANNED
.BYTES_SPILLED_TO_REMOTE_STORAGE
, BYTES_SPILLED_TO_REMOTE_STOFAGE / BYTES_SCANNED AS SPILLING_READ_RATIO
FROM "SNOWFLAKE". "ACCOUNT-USAGE". "QVEP.Y HISTOP.Y"
WHERE BYTES_SPILLED_TO_F.EMDTE_STORAGE > BYTES_SCANNED • 5
ORDER BY SETLLING_F£AL_FATIO DESC ;
```

B.

Increase the size of the warehouse cache to speed up concurrent queues Identify the concurrent queues using ths query

```
SELECT MAHEBCWWJBKE ,CCW7<*> A5 QUEHYJXMW aSlWiBYTSS.^ANWEri AS BYTEBBCamD
.SUM(BYTE5_5rANNEr • FEJOTTACT-SCAWKID_FE5M_^?KEI A3 BYTES.5CAJiNED_F>Ctt_C A--7BI , 'TH<BYTES.?.CAKBED*PER'UJTA^.3'AmD.FPnM.CA^KEt
/ SLWBrnJJKAHKEI I AS 5EP?£hT_L^!IH_FFC«_^^
EMM "3lMiffLAKE'. *ACOQUIT.USAGE*. "QUERY .HISTORY*
WHERE. STABS.EWE ~ LATEAIC Ixuith.-h cnxxejit.tiaejtaspO I AML BYTES,3'JL'ttJEL C GHOIE BY i ORCEB BY 5 r
```

C.

Introduce mufti-cluster warehouses to help with concurrent queues Identify the concurrent queries by running this query

```
SELECT TO_PATE(STAPT_TIME) AS DATE
,WAREHOUSE_NAME .SUMtAVGJWNNING) AS SUN-RUNNING ,SUM<AVG_QUEUED_LOAD) AS SIIM_QUEUED FROM
"SNOWFLAKE"."ACCOUNTJJSAGE"."WAREHOUSE_DOAD_HISTOEY" WHERE TO_DAT£(START TIME I >- DATEAIiD(i»jnth,-1,CURRENT—
TIMESTAMP ()) GROUP BY 1,2 HAVING SUM(AVG_QUEVED_LOAD| >0 ;
```

D.

Identfy etiicti queries are spiled to mmole slot age and change the watehouse pniarmiers to address Tua issue Identity Iha issue by running ths query

```
SELECT QUEM^IC
```

, SISFTr igCESr_TEJCI, 1, f > EA.rTUU._ ."JEH-TKT
, t>SEF_KAMt
,1»SEHOOSE_r«WE
.WUEKXtSE^SHE
, BnEE_SWLUEti_TO_M>m_JT»AUE
, STAFT TtiHr., EHOTWE
,TOTAL_ElAFSED_rM/1000 TOTAL_ELAFEEB_Tri« & *
FKK :;:HlilFLAEE.^:CWi_tifME.';nOT SHEET HETBr«KLLEIi_TO_RB»K_STOFA6E - 0 AKE 5TMir_nMEssMWE ■ MTBMIOI'DM1
•.-UWtEITT LATE I CMiEi B/SWES SEIIXED TO DEMOTE BTORME CCSC LIHIT'10 ;

Answer: A

Explanation:
The image shows a SQL query that can be used to identify which queries are spilled to remote storage and suggests changing the warehouse parameters to address this issue. Spilling to remote storage occurs when the memory allocated to a warehouse is insufficient to process a query, and Snowflake uses disk or cloud storage as a temporary cache. This can significantly slow down the query performance and increase the cost. To troubleshoot this issue, a Snowflake Architect can run the query shown in the image to find out which queries are spilling, how much data they are spilling, and which warehouses they are using. Then, the architect can adjust the warehouse size, type, or scaling policy to provide enough memory for the queries and avoid spilling12. Reference: Recognizing Disk Spilling Managing the Kafka Connector

Question: 98

What is a key consideration when setting up search optimization service for a table?

A. Search optimization service works best with a column that has a minimum of 100 K distinct values. B. Search optimization service can significantly improve query performance on partitioned external tables.
C. Search optimization service can help to optimize storage usage by compressing the data into a GZIP format.
D. The table must be clustered with a key having multiple columns for effective search optimization.

Answer: A

Explanation:

Search optimization service is a feature of Snowflake that can significantly improve the performance of certain types of lookup and analytical queries on tables. Search optimization service creates and maintains a persistent data structure called a search access path, which keeps track of which values of the table's columns might be found in each of its micro-partitions, allowing some micro-partitions to be skipped when scanning the table1.
Search optimization service can significantly improve query performance on partitioned external tables, which are tables that store data in external locations such as Amazon S3 or Google Cloud Storage. Partitioned external tables can leverage the search access path to prune the partitions that do not contain the relevant data, reducing the amount of data that needs to be scanned and transferred from the external location2.
The other options are not correct because:
A) Search optimization service works best with a column that has a high cardinality, which means that the column has many distinct values. However, there is no specific minimum number of distinct values required for search optimization service to work effectively. The actual performance improvement depends on the selectivity of the queries and the distribution of the data1.
C) Search optimization service does not help to optimize storage usage by compressing the data into a GZIP format. Search optimization service does not affect the storage format or compression of the data, which is determined by the file format options of the table. Search optimization service only creates an additional data structure that is stored separately from the table data1.
D) The table does not need to be clustered with a key having multiple columns for effective search optimization. Clustering is a feature of Snowflake that allows ordering the data in a table or a partitioned external table based on one or more clustering keys. Clustering can improve the performance of queries that filter on the clustering keys, as it reduces the number of micro-partitions that need to be scanned. However, clustering is not required for search

optimization service to work, as search optimization service can skip micro-partitions based on any column that has a search
access path, regardless of the clustering key3.

Reference:

1: Search Optimization Service | Snowflake Documentation
2: Partitioned External Tables | Snowflake Documentation
3: Clustering Keys | Snowflake Documentation

Question: 99

A retail company has 2000+ stores spread across the country. Store Managers report that they are having trouble running key reports related to inventory management, sales targets, payroll, and staffing during business hours. The Managers report that performance is poor and time-outs occur frequently.

Currently all reports share the same Snowflake virtual warehouse.

How should this situation be addressed? (Select TWO).

A. Use a Business Intelligence tool for in-memory computation to improve performance.
B. Configure a dedicated virtual warehouse for the Store Manager team.
C. Configure the virtual warehouse to be multi-clustered.
D. Configure the virtual warehouse to size 4-XL
E. Advise the Store Manager team to defer report execution to off-business hours.

Answer: B, C

Explanation:

The best way to address the performance issues and time-outs faced by the Store Manager team is to configure a dedicated virtual warehouse for them and make it multi-clustered. This will allow them to run their reports independently from other workloads and scale up or down the compute resources as needed. A dedicated virtual warehouse will also enable them to apply specific security and access policies for their data. A multi-clustered virtual warehouse will provide high availability and concurrency for their queries and avoid queuing or throttling.

Using a Business Intelligence tool for in-memory computation may improve performance, but it will not solve the underlying issue of insufficient compute resources in the shared virtual warehouse. It will also introduce additional costs and complexity for the data architecture.

Configuring the virtual warehouse to size 4-XL may increase the performance, but it will also increase the cost and may not be optimal for the workload. It will also not address the concurrency and availability issues that may arise from sharing the virtual warehouse with other workloads.

Advising the Store Manager team to defer report execution to off-business hours may reduce the load on the shared virtual warehouse, but it will also reduce the timeliness and usefulness of the reports for the business. It will also not guarantee that the performance issues and time-outs will not occur at other times.

Reference:

Snowflake Architect Training
Snowflake SnowPro Advanced Architect Certification - Preparation Guide
SnowPro Advanced: Architect Exam Study Guide

Question: 100

A company needs to have the following features available in its Snowflake account:

1. Support for Multi-Factor Authentication (MFA)
2. A minimum of 2 months of Time Travel availability
3. Database replication in between different regions

4. Native support for JDBC and ODBC
5. Customer-managed encryption keys using Tri-Secret Secure
6. Support for Payment Card Industry Data Security Standards (PCI DSS)

In order to provide all the listed services, what is the MINIMUM Snowflake edition that should be selected during account creation?

A. Standard
B. Enterprise
C. Business Critical
D. Virtual Private Snowflake (VPS)

Answer: C

Explanation:

According to the Snowflake documentation1, the Business Critical edition offers the following features that are relevant to the

question:

Support for

Multi-Factor Authentication (MFA): This is a standard feature available in all Snowflake editions1.

A minimum of 2 months of Time Travel availability: This is an enterprise feature that allows users to access historical data for up to 90 days1.

Database replication in between different regions: This is an enterprise feature that enables users to replicate databases across different regions or cloud platforms1.

Native support for JDBC and ODBC: This is a standard feature available in all Snowflake editions1. Customer-managed encryption keys using Tri-Secret Secure: This is a business critical feature that provides enhanced security and data protection by allowing customers to manage their own encryption keys1.

Support for Payment Card Industry Data Security Standards (PCI DSS): This is a business critical feature that ensures compliance with PCI DSS regulations for handling sensitive cardholder data1. Therefore, the minimum Snowflake edition that should be selected during account creation to provide all the listed services is the Business Critical edition.

Reference:

Snowflake Editions | Snowflake Documentation

Question: 101

A media company needs a data pipeline that will ingest customer review data into a Snowflake table, and apply some transformations. The company also needs to use Amazon Comprehend to do sentiment analysis and make the de-identified final data set available publicly for advertising companies who use different cloud providers in different regions.

The data pipeline needs to run continuously and efficiently as new records arrive in the object storage leveraging event notifications. Also, the operational complexity, maintenance of the infrastructure, including platform upgrades and security, and the development effort should be minimal.

Which design will meet these requirements?

A. Ingest the data using copy into and use streams and tasks to orchestrate transformations. Export the data into Amazon S3 to do model inference with Amazon Comprehend and ingest the data back into a Snowflake table. Then create a listing in the Snowflake Marketplace to make the data available to other companies.

B. Ingest the data using Snowpipe and use streams and tasks to orchestrate transformations. Create an external function to do model inference with Amazon Comprehend and write the final records to a Snowflake table. Then create

a listing in the Snowflake Marketplace to make the data available to other companies.

C. Ingest the data into Snowflake using Amazon EMR and PySpark using the Snowflake Spark connector. Apply transformations using another Spark job. Develop a python program to do model inference by leveraging the Amazon Comprehend text analysis API. Then write the results to a Snowflake table and create a listing in the Snowflake Marketplace to make the data available to other companies.

D. Ingest the data using Snowpipe and use streams and tasks to orchestrate transformations. Export the data into Amazon S3 to do model inference with Amazon Comprehend and ingest the data back into a Snowflake table. Then create a listing in the Snowflake Marketplace to make the data available to other companies.

Answer: B

Explanation:

Option B is the best design to meet the requirements because it uses Snowpipe to ingest the data continuously and efficiently as new records arrive in the object storage, leveraging event notifications. Snowpipe is a service that automates the loading of data from external sources into Snowflake tables1. It also uses streams and tasks to orchestrate transformations on the ingested data. Streams are objects that store the change history of a table, and tasks are objects that execute SQL statements on a schedule or when triggered by another task2. Option B also uses an external function to do model inference with Amazon Comprehend and write the final records to a Snowflake table. An external function is a user-defined function that calls an external API, such as Amazon Comprehend, to perform computations that are not natively supported by Snowflake3. Finally, option B uses the Snowflake Marketplace to make the de-identified final data set available publicly for advertising companies who use different cloud providers in different regions. The Snowflake Marketplace is a platform that enables data providers to list and share their data sets with data consumers, regardless of the cloud platform or region they use4.

Option A is not the best design because it uses copy into to ingest the data, which is not as efficient and continuous as Snowpipe. Copy into is a SQL command that loads data from files into a table in a single transaction. It also exports the data into Amazon S3 to do model inference with Amazon Comprehend, which adds an extra step and increases the operational complexity and maintenance of the infrastructure.

Option C is not the best design because it uses Amazon EMR and PySpark to ingest and transform the data, which also increases the operational complexity and maintenance of the infrastructure. Amazon EMR is a cloud service that provides a managed Hadoop framework to process and analyze large-scale data sets. PySpark is a Python API for Spark, a distributed computing framework that can run on Hadoop. Option C also develops a python program to do model inference by leveraging the Amazon Comprehend text analysis API, which increases the development effort.

Option D is not the best design because it is identical to option A, except for the ingestion method. It still exports the data into Amazon S3 to do model inference with Amazon Comprehend, which adds

an extra step and increases the operational complexity and maintenance of the infrastructure. Reference: 1: Snowpipe Overview 2: Using Streams and Tasks to Automate Data Pipelines 3: External Functions Overview 4: Snowflake Data Marketplace Overview : [Loading Data Using COPY INTO] : [What is Amazon EMR?] : [PySpark Overview]

Question: 102

When using the copy into <table> command with the CSV file format, how does the match_by_column_name parameter behave?

A. It expects a header to be present in the CSV file, which is matched to a case-sensitive table column name.

B. The parameter will be ignored.

C. The command will return an error.

D. The command will return a warning stating that the file has unmatched columns.

Answer: B

Explanation:

Option B is the best design to meet the requirements because it uses Snowpipe to ingest the data continuously and efficiently as new records arrive in the object storage, leveraging event notifications. Snowpipe is a service that automates the loading of data from external sources into Snowflake tables1. It also uses streams and tasks to orchestrate transformations on the ingested data. Streams are objects that store the change history of a table, and tasks are objects that execute SQL statements on a schedule or when triggered by another task2. Option B also uses an external function to do model inference with Amazon Comprehend and write the final records to a Snowflake table. An external function is a user-defined function that calls an external API, such as Amazon Comprehend, to perform computations that are not natively supported by Snowflake3. Finally, option B uses the Snowflake Marketplace to make the de-identified final data set available publicly for advertising companies who use different cloud providers in different regions. The Snowflake Marketplace is a platform that enables data providers to list and share their data sets with data consumers, regardless of the cloud platform or region they use4.

Option A is not the best design because it uses copy into to ingest the data, which is not as efficient and continuous as Snowpipe. Copy into is a SQL command that loads data from files into a table in a single transaction. It also exports the data into Amazon S3 to do model inference with Amazon Comprehend, which adds an extra step and increases the operational complexity and maintenance of the infrastructure.

Option C is not the best design because it uses Amazon EMR and PySpark to ingest and transform the data, which also increases the operational complexity and maintenance of the infrastructure. Amazon EMR is a cloud service that provides a managed Hadoop framework to process and analyze large-scale data sets. PySpark is a Python API for Spark, a distributed computing framework that can run on Hadoop. Option C also develops a python program to do model inference by leveraging the Amazon Comprehend text analysis API, which increases the development effort.

Option D is not the best design because it is identical to option A, except for the ingestion method. It still exports the data into Amazon S3 to do model inference with Amazon Comprehend, which adds an extra step and increases the operational complexity and maintenance of the infrastructure. Reference: 1: Snowpipe Overview 2: Using Streams and Tasks to Automate Data Pipelines 3: External Functions Overview 4: Snowflake Data Marketplace Overview : [Loading Data Using COPY INTO] :

[What is Amazon EMR?] : [PySpark Overview]

The copy into <table> command is used to load data from staged files into an existing table in Snowflake. The command supports various file formats, such as CSV, JSON, AVRO, ORC, PARQUET, and XML1.

The match_by_column_name parameter is a copy option that enables loading semi-structured data into separate columns in the target table that match corresponding columns represented in the source data. The parameter can have one of the following values2:

CASE_SENSITIVE: The column names in the source data must match the column names in the target table exactly, including the case. This is the default value.

CASE_INSENSITIVE: The column names in the source data must match the column names in the target table, but the case is ignored.

NONE: The column names in the source data are ignored, and the data is loaded based on the order of the columns in the target table.

The match_by_column_name parameter only applies to semi-structured data, such as JSON, AVRO, ORC, PARQUET, and XML. It does not apply to CSV data, which is considered structured data2.

When using the copy into <table> command with the CSV file format, the match_by_column_name parameter behaves as follows2:

It expects a header to be present in the CSV file, which is matched to a case-sensitive table column name. This means that the first row of the CSV file must contain the column names, and they must match the column names in the target table exactly, including the case. If the header is missing or does not match, the command will return an

error.

The parameter will not be ignored, even if it is set to NONE. The command will still try to match the column names in the CSV file with the column names in the target table, and will return an error if they do not match.

The command will not return a warning stating that the file has unmatched columns. It will either load the data successfully if the column names match, or return an error if they do not match. Reference:

1: COPY INTO <table> | Snowflake Documentation

2: MATCH_BY_COLUMN_NAME | Snowflake Documentation

Question: 103

How can the Snowflake context functions be used to help determine whether a user is authorized to see data that has column-level security enforced? (Select TWO).

A. Set masking policy conditions using current_role targeting the role in use for the current session.

B. Set masking policy conditions using is_role_in_session targeting the role in use for the current account.

C. Set masking policy conditions using invoker_role targeting the executing role in a SQL statement.

D. Determine if there are ownership privileges on the masking policy that would allow the use of any function.

E. Assign the accountadmin role to the user who is executing the object.

Answer: A, C

Explanation:

Snowflake context functions are functions that return information about the current session, user, role, warehouse, database, schema, or object. They can be used to help determine whether a user is authorized to see data that has column-level security enforced by setting masking policy conditions based on the context functions. The following context functions are relevant for column-level security:

current_role: This function returns the name of the role in use for the current session. It can be used to set masking policy conditions that target the current session and are not affected by the execution context of the SQL statement. For example, a masking policy condition using current_role can allow or deny access to a column based on the role that the user activated in the session.

invoker_role: This function returns the name of the executing role in a SQL statement. It can be used to set masking policy conditions that target the executing role and are affected by the execution context of the SQL statement. For example, a masking policy condition using invoker_role can allow or deny access to a column based on the role that the user specified in the SQL statement, such as using the AS ROLE clause or a stored procedure.

is_role_in_session: This function returns TRUE if the user's current role in the session (i.e. the role returned by current_role) inherits the privileges of the specified role. It can be used to set masking policy conditions that involve role hierarchy and privilege inheritance. For example, a masking policy condition using is_role_in_session can allow or deny access to a column based on whether the user's current role is a lower privilege role in the specified role hierarchy.

The other options are not valid ways to use the Snowflake context functions for column-level security:

Set masking policy conditions using is_role_in_session targeting the role in use for the current account. This option is incorrect because is_role_in_session does not target the role in use for the current account, but rather the role in use for the current session. Also, the current account is not a role, but rather a logical entity that contains users, roles, warehouses, databases, and other objects. Determine if there are ownership privileges on the masking policy that would allow the use of any function. This option is incorrect because ownership privileges on the masking policy do not affect the use of any function, but rather the ability to create, alter, or drop the masking policy. Also, this is not a way to use the Snowflake context functions, but rather a way to check the privileges on the masking policy object.

Assign the accountadmin role to the user who is executing the object. This option is incorrect because assigning the accountadmin role to the user who is executing the object does not involve using the Snowflake context functions, but

rather granting the highest-level role to the user. Also, this is not a recommended practice for column-level security, as it would give the user full access to all objects and data in the account, which could compromise data security and governance. Reference:

Context Functions

Advanced Column-level Security topics

Snowflake Data Governance: Column Level Security Overview

Data Security Snowflake Part 2 - Column Level Security

Question: 104

A retail company has over 3000 stores all using the same Point of Sale (POS) system. The company wants to deliver near real-time sales results to category managers. The stores operate in a variety of time zones and exhibit a dynamic range of transactions each minute, with some stores having higher sales volumes than others.

Sales results are provided in a uniform fashion using data engineered fields that will be calculated in a complex data pipeline. Calculations include exceptions, aggregations, and scoring using external functions interfaced to scoring algorithms. The source data for aggregations has over 100M rows. Every minute, the POS sends all sales transactions files to a cloud storage location with a naming convention that includes store numbers and timestamps to identify the set of transactions contained in the files. The files are typically less than 10MB in size.

How can the near real-time results be provided to the category managers? (Select TWO).

A. All files should be concatenated before ingestion into Snowflake to avoid micro-ingestion.

B. A Snowpipe should be created and configured with AUTO_INGEST = true. A stream should be created to process INSERTS into a single target table using the stream metadata to inform the store number and timestamps.

C. A stream should be created to accumulate the near real-time data and a task should be created that runs at a frequency that matches the real-time analytics needs.

D. An external scheduler should examine the contents of the cloud storage location and issue SnowSQL commands to process the data at a frequency that matches the real-time analytics needs. E. The copy into command with a task scheduled to run every second should be used to achieve the near-real time requirement.

Answer: B, C

Explanation:

To provide near real-time sales results to category managers, the Architect can use the following steps:

Create an external stage that references the cloud storage location where the POS sends the sales transactions files. The external stage should use the file format and encryption settings that match the source files2

Create a Snowpipe that loads the files from the external stage into a target table in Snowflake. The Snowpipe should be configured with AUTO_INGEST = true, which means that it will automatically detect and ingest new files as they arrive in the external stage. The Snowpipe should also use a copy option to purge the files from the external stage after loading, to avoid duplicate ingestion3 Create a stream on the target table that captures the INSERTS made by the Snowpipe. The stream should include the metadata columns that provide information about the file name, path, size, and last modified time. The stream should also have a retention period that matches the real-time analytics needs4

Create a task that runs a query on the stream to process the near real-time data. The query should use the stream metadata to extract the store number and timestamps from the file name and path, and perform the calculations for exceptions, aggregations, and scoring using external functions. The query should also output the results to another table or view that can be accessed by the category managers. The task should be scheduled to run at a frequency that matches the real-time analytics needs, such as every minute or every 5 minutes.

The other options are not optimal or feasible for providing near real-time results:

All files should be concatenated before ingestion into Snowflake to avoid micro-ingestion. This option is not recommended because it would introduce additional latency and complexity in the data pipeline. Concatenating files would require an external process or service that monitors the cloud storage location and performs the file merging operation. This would delay the ingestion of new files into Snowflake and increase the risk of data loss or corruption. Moreover, concatenating files would not avoid micro-ingestion, as Snowpipe would still ingest each concatenated file as a separate load. An external scheduler should examine the contents of the cloud storage location and issue SnowSQL commands to process the data at a frequency that matches the real-time analytics needs. This option is not necessary because Snowpipe can automatically ingest new files from the external stage without requiring an external trigger or scheduler. Using an external scheduler would add more overhead and dependency to the data pipeline, and it would not guarantee near real-time ingestion, as it would depend on the polling interval and the availability of the external scheduler.

The copy into command with a task scheduled to run every second should be used to achieve the near-real time requirement. This option is not feasible because tasks cannot be scheduled to run every second in Snowflake. The minimum interval for tasks is one minute, and even that is not guaranteed, as tasks are subject to scheduling delays and concurrency limits. Moreover, using the copy into command with a task would not leverage the benefits of Snowpipe, such as automatic file detection, load balancing, and micro-partition optimization. Reference:

1: SnowPro Advanced: Architect | Study Guide
2: Snowflake Documentation | Creating Stages
3: Snowflake Documentation | Loading Data Using Snowpipe
4: Snowflake Documentation | Using Streams and Tasks for ELT
: Snowflake Documentation | Creating Tasks
: Snowflake Documentation | Best Practices for Loading Data
: Snowflake Documentation | Using the Snowpipe REST API
: Snowflake Documentation | Scheduling Tasks
: SnowPro Advanced: Architect | Study Guide
: Creating Stages
: Loading Data Using Snowpipe
: Using Streams and Tasks for ELT
: [Creating Tasks]
: [Best Practices for Loading Data]
: [Using the Snowpipe REST API]
: [Scheduling Tasks]

Question: 105

A company needs to share its product catalog data with one of its partners. The product catalog data is stored in two database tables: product_category, and product_details. Both tables can be joined by the product_id column. Data access should be governed, and only the partner should have access to the records.

The partner is not a Snowflake customer. The partner uses Amazon S3 for cloud storage.

Which design will be the MOST cost-effective and secure, while using the required Snowflake features?

A. Use Secure Data Sharing with an S3 bucket as a destination.
B. Publish product_category and product_details data sets on the Snowflake Marketplace.
C. Create a database user for the partner and give them access to the required data sets.
D. Create a reader account for the partner and share the data sets as secure views.

Answer: D

Explanation:

A reader account is a type of Snowflake account that allows external users to access data shared by a provider account without being a Snowflake customer. A reader account can be created and managed by the provider account, and can use the Snowflake web interface or JDBC/ODBC drivers to query the shared data. A reader account is billed to the provider account based on the credits consumed by the queries1. A secure view is a type of view that applies row-level security filters to the underlying tables, and masks the data that is not accessible to the user. A secure view can be shared with a reader account to provide granular and governed access to the data2. In this scenario, creating a reader account for the partner and sharing the data sets as secure views would be the most cost-effective and secure design, while using the required Snowflake features, because:
It would avoid the data transfer and storage costs of using an S3 bucket as a destination, and the potential security risks of exposing the data to unauthorized access or modification.
It would avoid the complexity and overhead of publishing the data sets on the Snowflake Marketplace, and the potential loss of control over the data ownership and pricing.
It would avoid the need to create a database user for the partner and grant them access to the required data sets, which would require the partner to have a Snowflake account and consume the provider's resources.
Reference:
Reader Accounts
Secure Views

Question: 106

A company has a Snowflake environment running in AWS us-west-2 (Oregon). The company needs to share data privately with a customer who is running their Snowflake environment in Azure East US 2 (Virginia).
What is the recommended sequence of operations that must be followed to meet this requirement?

A. 1. Create a share and add the database privileges to the share
2. Create a new listing on the Snowflake Marketplace
3. Alter the listing and add the share
4. Instruct the customer to subscribe to the listing on the Snowflake Marketplace

B. 1. Ask the customer to create a new Snowflake account in Azure EAST US 2 (Virginia)
2. Create a share and add the database privileges to the share
3. Alter the share and add the customer's Snowflake account to the share

C. 1. Create a new Snowflake account in Azure East US 2 (Virginia)
2. Set up replication between AWS us-west-2 (Oregon) and Azure East US 2 (Virginia) for the database objects to be shared
3. Create a share and add the database privileges to the share
4. Alter the share and add the customer's Snowflake account to the share

D. 1. Create a reader account in Azure East US 2 (Virginia)
2. Create a share and add the database privileges to the share
3. Add the reader account to the share
4. Share the reader account's URL and credentials with the customer

Answer:C

Option C is the correct answer because it allows the company to share data privately with the customer across different cloud platforms and regions. The company can create a new Snowflake account in Azure East US 2 (Virginia) and set up replication between AWS us-west-2 (Oregon) and Azure East US 2 (Virginia) for the database objects to be shared. This way, the company can ensure that the data is always up to date and consistent in both accounts. The company can then create a

share and add the database privileges to the share, and alter the share and add the customer's Snowflake account to the share. The customer can then access the shared data from their own Snowflake account in Azure East US 2 (Virginia).

Option A is incorrect because the Snowflake Marketplace is not a private way of sharing data. The Snowflake Marketplace is a public data exchange platform that allows anyone to browse and subscribe to data sets from various providers. The company would not be able to control who can access their data if they use the Snowflake Marketplace.

Option B is incorrect because it requires the customer to create a new Snowflake account in Azure East US 2 (Virginia), which may not be feasible or desirable for the customer. The customer may already have an existing Snowflake account in a different cloud platform or region, and may not want to incur additional costs or complexity by creating a new account.

Option D is incorrect because it involves creating a reader account in Azure East US 2 (Virginia), which is a limited and temporary way of sharing data. A reader account is a special type of Snowflake account that can only access data from a single share, and has a fixed duration of 30 days. The company would have to manage the reader account's URL and credentials, and renew the account every 30 days. The customer would not be able to use their own Snowflake account to access the shared data, and would have to rely on the company's reader account.

Reference:

Snowflake Replication

Secure Data Sharing Overview

Snowflake Marketplace Overview

Reader Account Overview

Question: 107

Company A has recently acquired company B. The Snowflake deployment for company B is located in the Azure West Europe region.

As part of the integration process, an Architect has been asked to consolidate company B's sales data into company A's Snowflake account which is located in the AWS us-east-1 region.

How can this requirement be met?

A. Replicate the sales data from company B's Snowflake account into company A's Snowflake account using cross-region data replication within Snowflake. Configure a direct share from company B's account to company A's account.

B. Export the sales data from company B's Snowflake account as CSV files, and transfer the files to company A's Snowflake account. Import the data using Snowflake's data loading capabilities.

C. Migrate company B's Snowflake deployment to the same region as company A's Snowflake deployment, ensuring data locality. Then perform a direct database-to-database merge of the sales data.

D. Build a custom data pipeline using Azure Data Factory or a similar tool to extract the sales data from company B's Snowflake account. Transform the data, then load it into company A's Snowflake account.

Answer: A

Explanation:

The best way to meet the requirement of consolidating company B's sales data into company A's Snowflake account is to use cross-region data replication within Snowflake. This feature allows data providers to securely share data with data consumers across different regions and cloud platforms. By replicating the sales data from company B's account in Azure West Europe region to company A's account in AWS us-east-1 region, the data will be synchronized and available for consumption. To enable data replication, the accounts must be linked and

replication must be enabled by a user with the ORGADMIN role. Then, a replication group must be created and the sales database must be added to the group. Finally, a direct share must be configured from company B's account to company A's account to grant access to the replicated data. This option is more efficient and secure than exporting and importing data using CSV files or migrating the entire Snowflake deployment to another region or cloud platform. It also does not require building a custom data pipeline using external tools.

Reference:

Sharing data securely across regions and cloud platforms

Introduction to replication and failover

Replication considerations

Replicating account objects

Question: 108

A Snowflake Architect created a new data share and would like to verify that only specific records in secure views are visible within the data share by the consumers.

What is the recommended way to validate data accessibility by the consumers?

A. Create reader accounts as shown below and impersonate the consumers by logging in with their credentials.
create managed account reader_acctl admin_name = userl , adroin_password ■ 'Sdfed43da!44T , type = reader;

B. Create a row access policy as shown below and assign it to the data share.
create or replace row access policy rap_acct as (acct_id varchar) returns boolean -> case when 'acctl_role' = current_role() then true else false end;

C. Set the session parameter called SIMULATED_DATA_SHARING_C0NSUMER as shown below in order to impersonate the consumer accounts.
alter session set simulated_data_sharing_consumer - 'Consumer Acctl*

D. Alter the share settings as shown below, in order to impersonate a specific consumer account. alter share sales share set accounts = 'Consumerl' share restrictions = true

Answer: C

Explanation:

The SIMULATED_DATA_SHARING_CONSUMER session parameter allows a data provider to simulate the data access of a consumer account without creating a reader account or logging in with the consumer credentials. This parameter can be used to validate the data accessibility by the consumers in a data share, especially when using secure views or secure UDFs that filter data based on the current account or role. By setting this parameter to the name of a consumer account, the data provider can see the same data as the consumer would see when querying the shared database. This is a convenient and efficient way to test the data sharing functionality and ensure that only the intended data is visible to the consumers.

Reference:

Using the SIMULATED_DATA_SHARING_CONSUMER Session Parameter

SnowPro Advanced: Architect Exam Study Guide

Question: 109

A company is using Snowflake in Azure in the Netherlands. The company analyst team also has data in JSON format that is stored in an Amazon S3 bucket in the AWS Singapore region that the team wants to analyze.

The Architect has been given the following requirements:

1. Provide access to frequently changing data

2. Keep egress costs to a minimum
3. Maintain low latency

How can these requirements be met with the LEAST amount of operational overhead?

A. Use a materialized view on top of an external table against the S3 bucket in AWS Singapore.
B. Use an external table against the S3 bucket in AWS Singapore and copy the data into transient tables.
C. Copy the data between providers from S3 to Azure Blob storage to collocate, then use Snowpipe for data ingestion.
D. Use AWS Transfer Family to replicate data between the S3 bucket in AWS Singapore and an Azure Netherlands Blob storage, then use an external table against the Blob storage.

Answer: B

Explanation:

: Option A is the best design to meet the requirements because it uses a materialized view on top of an external table against the S3 bucket in AWS Singapore. A materialized view is a database object that contains the results of a query and can be refreshed periodically to reflect changes in the underlying data1. An external table is a table that references data files stored in a cloud storage service, such as Amazon S32. By using a materialized view on top of an external table, the company can provide access to frequently changing data, keep egress costs to a minimum, and maintain low latency. This is because the materialized view will cache the query results in Snowflake, reducing the need to access the external data files and incur network charges. The materialized view will also improve the query performance by avoiding scanning the external data files every time. The materialized view can be refreshed on a schedule or on demand to capture the changes in the external data files1.

Option B is not the best design because it uses an external table against the S3 bucket in AWS Singapore and copies the data into transient tables. A transient table is a table that is not subject to the Time Travel and Fail-safe features of Snowflake, and is automatically purged after a period of time3. By using an external table and copying the data into transient tables, the company will incur more egress costs and operational overhead than using a materialized view. This is because the external table will access the external data files every time a query is executed, and the copy operation will also transfer data from S3 to Snowflake. The transient tables will also consume more storage space in Snowflake and require manual maintenance to ensure they are up to date. Option C is not the best design because it copies the data between providers from S3 to Azure Blob storage to collocate, then uses Snowpipe for data ingestion. Snowpipe is a service that automates the loading of data from external sources into Snowflake tables4. By copying the data between providers, the company will incur high egress costs and latency, as well as operational complexity and maintenance of the infrastructure. Snowpipe will also add another layer of processing and storage in Snowflake, which may not be necessary if the external data files are already in a queryable format.

Option D is not the best design because it uses AWS Transfer Family to replicate data between the S3 bucket in AWS Singapore and an Azure Netherlands Blob storage, then uses an external table against the Blob storage. AWS Transfer Family is a service that enables secure and seamless transfer of files over SFTP, FTPS, and FTP to and from Amazon S3 or Amazon EFS5. By using AWS Transfer Family, the company will incur high egress costs and latency, as well as operational complexity and maintenance of the infrastructure. The external table will also access the external data files every time a query is executed, which may affect the query performance.

Reference: 1: Materialized Views 2: External Tables 3: Transient Tables 4: Snowpipe Overview 5: AWS Transfer Family

Question: 110

Based on the Snowflake object hierarchy, what securable objects belong directly to a Snowflake account? (Select THREE).

A. Database
B. Schema
C. Table
D. Stage
E. Role
F. Warehouse

Answer: A, E, F

Explanation:

A securable object is an entity to which access can be granted in Snowflake. Securable objects include databases, schemas, tables, views, stages, pipes, functions, procedures, sequences, tasks, streams, roles, warehouses, and shares1.
The Snowflake object hierarchy is a logical structure that organizes the securable objects in a nested manner. The top-most container is the account, which contains all the databases, roles, and warehouses for the customer organization. Each database contains schemas, which in turn contain tables, views, stages, pipes, functions, procedures, sequences, tasks, and streams. Each role can be granted privileges on other roles or securable objects. Each warehouse can be used to execute queries on securable objects2.
Based on the Snowflake object hierarchy, the securable objects that belong directly to a Snowflake account are databases, roles, and warehouses. These objects are created and managed at the account level, and do not depend on any other securable object. The other options are not correct because:
Schemas belong to databases, not to accounts. A schema must be created within an existing database3.
Tables belong to schemas, not to accounts. A table must be created within an existing schema4. Stages belong to schemas or tables, not to accounts. A stage must be created within an existing schema or table.
Reference:
1: Overview of Access Control | Snowflake Documentation
2: Securable Objects | Snowflake Documentation
3: CREATE SCHEMA | Snowflake Documentation
4: CREATE TABLE | Snowflake Documentation [5]: CREATE STAGE | Snowflake Documentation

Question: 111

What is a characteristic of Role-Based Access Control (RBAC) as used in Snowflake?

A. Privileges can be granted at the database level and can be inherited by all underlying objects.
B. A user can use a "super-user" access along with securityadmin to bypass authorization checks and access all databases, schemas, and underlying objects.
C. A user can create managed access schemas to support future grants and ensure only schema owners can grant privileges to other roles.
D. A user can create managed access schemas to support current and future grants and ensure only object owners can grant privileges to other roles.

Answer: A C

Explanation:

Role-Based Access Control (RBAC) is the Snowflake Access Control Framework that allows privileges to be granted by object owners to roles, and roles, in turn, can be assigned to users to restrict or allow actions to be performed on

objects. A characteristic of RBAC as used in Snowflake is: Privileges can be granted at the database level and can be inherited by all underlying objects. This means that a role that has a certain privilege on a database, such as CREATE SCHEMA or USAGE, can also perform the same action on any schema, table, view, or other object within that database, unless explicitly revoked. This simplifies the access control management and reduces the number of grants required.
A user can create managed access schemas to support future grants and ensure only schema owners can grant privileges to other roles. This means that a user can create a schema with the MANAGED ACCESS option, which changes the default behavior of object ownership and privilege granting within the schema. In a managed access schema, object owners lose the ability to grant privileges on their objects to other roles, and only the schema owner or a role with the MANAGE GRANTS privilege can do so. This enhances the security and governance of the schema and its objects. The other options are not characteristics of RBAC as used in Snowflake:
A user can use a "super-user" access along with securityadmin to bypass authorization checks and access all databases, schemas, and underlying objects. This is not true, as there is no such thing as a "super-user" access in Snowflake. The securityadmin role is a predefined role that can manage users and roles, but it does not have any privileges on any database objects by default. To access any object, the securityadmin role must be explicitly granted the appropriate privilege by the object owner or another role with the grant option.
A user can create managed access schemas to support current and future grants and ensure only object owners can grant privileges to other roles. This is not true, as this contradicts the definition of a managed access schema. In a managed access schema, object owners cannot grant privileges on their objects to other roles, and only the schema owner or a role with the MANAGE GRANTS privilege can do so.
Reference:
Overview of Access Control
A Functional Approach For Snowflake's Role-Based Access Controls
Snowflake Role-Based Access Control simplified
Snowflake RBAC security prefers role inheritance to role composition
Overview of Snowflake Role Based Access Control

Question: 112

Assuming all Snowflake accounts are using an Enterprise edition or higher, in which development and testing scenarios would be copying of data be required, and zero-copy cloning not be suitable? (Select TWO).

A. Developers create their own datasets to work against transformed versions of the live data.
B. Production and development run in different databases in the same account, and Developers need to see production-like data but with specific columns masked.
C. Data is in a production Snowflake account that needs to be provided to Developers in a separate development/testing Snowflake account in the same cloud region.
D. Developers create their own copies of a standard test database previously created for them in the development account, for their initial development and unit testing.
E. The release process requires pre-production testing of changes with data of production scale and complexity. For security reasons, pre-production also runs in the production account.

Answer: A C

Explanation:

Zero-copy cloning is a feature that allows creating a clone of a table, schema, or database without physically copying the data. Zero-copy cloning is suitable for scenarios where the cloned object needs to have the same data and metadata as the original object, and where the cloned object does not need to be modified or updated frequently. Zero-copy cloning is also suitable for scenarios where the cloned object needs to be shared within the same Snowflake account or across different accounts in the same cloud region2

However, zero-copy cloning is not suitable for scenarios where the cloned object needs to have different data or metadata than the original object, or where the cloned object needs to be modified or updated frequently. Zero-copy cloning is also not suitable for scenarios where the cloned object needs to be shared across different accounts in different cloud regions. In these scenarios, copying of data would be required, either by using the COPY INTO command or by using data sharing with secure views3

The following are examples of development and testing scenarios where copying of data would be required, and zero-copy cloning would not be suitable:

Developers create their own datasets to work against transformed versions of the live data. This scenario requires copying of data because the developers need to modify the data or metadata of the cloned object to perform transformations, such as adding, deleting, or updating columns, rows, or values. Zero-copy cloning would not be suitable because it would create a read-only clone that shares the same data and metadata as the original object, and any changes made to the clone would affect the original object as well4

Data is in a production Snowflake account that needs to be provided to Developers in a separate development/testing Snowflake account in the same cloud region. This scenario requires copying of data because the data needs to be shared across different accounts in the same cloud region. Zerocopy cloning would not be suitable because it would create a clone within the same account as the original object, and it would not allow sharing the clone with another account. To share data across different accounts in the same cloud region, data sharing with secure views or COPY INTO command can be used5

The following are examples of development and testing scenarios where zero-copy cloning would be suitable, and copying of data would not be required:

Production and development run in different databases in the same account, and Developers need to see production-like data but with specific columns masked. This scenario can use zero-copy cloning because the data needs to be shared within the same account, and the cloned object does not need to have different data or metadata than the original object. Zero-copy cloning can create a clone of the production database in the development database, and the clone can have the same data and metadata as the original database. To mask specific columns, secure views can be created on top of the clone, and the developers can access the secure views instead of the clone directly6 Developers create their own copies of a standard test database previously created for them in the development account, for their initial development and unit testing. This scenario can use zero-copy cloning because the data needs to be shared within the same account, and the cloned object does not need to have different data or metadata than the original object. Zero-copy cloning can create a clone of the standard test database for each developer, and the clone can have the same data and metadata as the original database. The developers can use the clone for their initial development and unit testing, and any changes made to the clone would not affect the original database or other clones7

The release process requires pre-production testing of changes with data of production scale and complexity. For security reasons, pre-production also runs in the production account. This scenario can use zero-copy cloning because the data needs to be shared within the same account, and the cloned object does not need to have different data or metadata than the original object. Zero-copy cloning can create a clone of the production database in the pre-production database, and the clone can have the same data and metadata as the original database. The pre-production testing can use the clone to test the changes with data of production scale and complexity, and any changes made to the clone would not affect the original database or the production environment8 Reference: 1: SnowPro Advanced: Architect | Study Guide 9

2: Snowflake Documentation | Cloning Overview

3: Snowflake Documentation | Loading Data Using COPY into a Table

4: Snowflake Documentation | Transforming Data During a Load

5: Snowflake Documentation | Data Sharing Overview

6: Snowflake Documentation | Secure Views

7: Snowflake Documentation | Cloning Databases, Schemas, and Tables

8: Snowflake Documentation | Cloning for Testing and Development : SnowPro Advanced: Architect | Study Guide

: Cloning Overview

: Loading Data Using COPY into a Table

: Transforming Data During a Load
: Data Sharing Overview
: Secure Views
: Cloning Databases, Schemas, and Tables
: Cloning for Testing and Development

Question: 113

A new user user_01 is created within Snowflake. The following two commands are executed:
Command 1-> show grants to user user_01;
Command 2 ~> show grants on user user 01;
What inferences can be made about these commands?

A. Command 1 defines which user owns user_01
Command 2 defines all the grants which have been given to user_01
B. Command 1 defines all the grants which are given to user_01 Command 2 defines which user owns user_01
C. Command 1 defines which role owns user_01
Command 2 defines all the grants which have been given to user_01
D. Command 1 defines all the grants which are given to user_01
Command 2 defines which role owns user 01

Answer: D

Explanation:

The SHOW GRANTS command in Snowflake can be used to list all the access control privileges that have been explicitly granted to roles, users, and shares. The syntax and the output of the command vary depending on the object type and the grantee type specified in the command1. In this question, the two commands have the following meanings:
Command 1: show grants to user user_01; This command lists all the roles granted to the user user_01. The output includes the role name, the grantee name, and the granted by role name for each grant. This command is equivalent to show grants to user current_user if user_01 is the current user1.
Command 2: show grants on user user_01; This command lists all the privileges that have been granted on the user object user_01. The output includes the privilege name, the grantee name, and the granted by role name for each grant. This command shows which role owns the user object user_01, as the owner role has the privilege to modify or drop the user object2.
Therefore, the correct inference is that command 1 defines all the grants which are given to user_01, and command 2 defines which role owns user_01.
Reference:
SHOW GRANTS
Understanding Access Control in Snowflake

Question: 114

Which technique will efficiently ingest and consume semi-structured data for Snowflake data lake workloads?

A. IDEF1X
B. Schema-on-write

C. Schema-on-read
D. Information schema

Answer: C

Explanation:
Option C is the correct answer because schema-on-read is a technique that allows Snowflake to ingest and consume semi-structured data without requiring a predefined schema. Snowflake supports various semi-structured data formats such as JSON, Avro, ORC, Parquet, and XML, and provides native data types (ARRAY, OBJECT, and VARIANT) for storing them. Snowflake also provides native support for querying semi-structured data using SQL and dot notation. Schema-on-read enables Snowflake to query semi-structured data at the same speed as performing relational queries while preserving the flexibility of schema-on-read. Snowflake's near-instant elasticity rightsizes compute resources, and consumption-based pricing ensures you only pay for what you use.
Option A is incorrect because IDEF1X is a data modeling technique that defines the structure and constraints of relational data using diagrams and notations. IDEF1X is not suitable for ingesting and consuming semi-structured data, which does not have a fixed schema or structure.
Option B is incorrect because schema-on-write is a technique that requires defining a schema before loading and processing data. Schema-on-write is not efficient for ingesting and consuming semistructured data, which may have varying or complex structures that are difficult to fit into a predefined schema. Schema-on-write also introduces additional overhead and complexity for data transformation and validation.
Option D is incorrect because information schema is a set of metadata views that provide information about the objects and privileges in a Snowflake database. Information schema is not a technique for ingesting and consuming semi-structured data, but rather a way of accessing metadata about the data.
Reference:
Semi-structured Data
Snowflake for Data Lake

Question: 115

What are characteristics of Dynamic Data Masking? (Select TWO).

A. A masking policy that Is currently set on a table can be dropped.
B. A single masking policy can be applied to columns in different tables.
C. A masking policy can be applied to the value column of an external table.
D. The role that creates the masking policy will always see unmasked data In query results E. A masking policy can be applied to a column with the GEOGRAPHY data type.

Answer: AB

Explanation:

Dynamic Data Masking is a feature that allows masking sensitive data in query results based on the role of the user who executes the query. A masking policy is a user-defined function that specifies the masking logic and can be applied to one or more columns in one or more tables. A masking policy that is currently set on a table can be dropped using the ALTER TABLE command. A single masking policy can be applied to columns in different tables using the ALTER TABLE command with the SET MASKING POLICY clause. The other options are either incorrect or not supported by Snowflake. A masking policy cannot be applied to the value column of an external table, as external tables do not support column-level security. The role that creates the masking policy will not always see unmasked data in query results, as the masking policy can be applied to the owner role as well. A masking policy cannot be applied to a column with the GEOGRAPHY data type, as Snowflake only supports masking policies for scalar data types. Reference: Snowflake Documentation: Dynamic Data

Masking, Snowflake Documentation: ALTER TABLE

Question: 116

What Snowflake system functions are used to view and or monitor the clustering metadata for a table? (Select TWO).

A. SYSTEMSCLUSTERING
B. SYSTEMSTABLE_CLUSTERING
C. SYSTEMSCLUSTERING_DEPTH
D. SYSTEMSCLUSTERING_RATIO
E. SYSTEMSCLUSTERINGINFORMATION

Answer: CE

Explanation:

The Snowflake system functions used to view and monitor the clustering metadata for a table are:
SYSTEM$CLUSTERING_INFORMATION
SYSTEM$CLUSTERING_DEPTH
Comprehensive But Short Explanation:
The SYSTEM$CLUSTERING_INFORMATION function in Snowflake returns a variety of clustering information for a specified table. This information includes the average clustering depth, total number of micro-partitions, total constant partition count, average overlaps, average depth, and a partition depth histogram. This function allows you to specify either one or multiple columns for which the clustering information is returned, and it returns this data in JSON format.
The SYSTEM$CLUSTERING_DEPTH function computes the average depth of a table based on specified columns or the clustering key defined for the table. A lower average depth indicates that the table is better clustered with respect to the specified columns. This function also allows specifying columns to calculate the depth, and the values need to be enclosed in single quotes. Reference:
SYSTEM$CLUSTERING_INFORMATION: Snowflake Documentation
SYSTEM$CLUSTERING_DEPTH: Snowflake Documentation

Question: 117

What is a characteristic of event notifications in Snowpipe?

A. The load history is stored In the metadata of the target table.
B. Notifications identify the cloud storage event and the actual data in the files.
C. Snowflake can process all older notifications when a paused pipe Is resumed.
D. When a pipe Is paused, event messages received for the pipe enter a limited retention period.

Answer: D

Explanation:

Event notifications in Snowpipe are messages sent by cloud storage providers to notify Snowflake of new or modified files in a stage. Snowpipe uses these notifications to trigger data loading from the

stage to the target table. When a pipe is paused, event messages received for the pipe enter a limited retention period, which varies depending on the cloud storage provider. If the pipe is not resumed within the retention period, the event messages will be discarded and the data will not be loaded automatically. To load the data, the pipe must be resumed

and the COPY command must be executed manually. This is a characteristic of event notifications in Snowpipe that distinguishes them from other options. Reference: Snowflake Documentation: Using Snowpipe, Snowflake Documentation: Pausing and Resuming a Pipe

Question: 118

An Architect needs to design a Snowflake account and database strategy to store and analyze large amounts of structured and semi-structured dat

a. There are many business units and departments within the company. The requirements are scalability, security, and cost efficiency.

What design should be used?

A. Create a single Snowflake account and database for all data storage and analysis needs, regardless of data volume or complexity.

B. Set up separate Snowflake accounts and databases for each department or business unit, to ensure data isolation and security.

C. Use Snowflake's data lake functionality to store and analyze all data in a central location, without the need for structured schemas or indexes

D. Use a centralized Snowflake database for core business data, and use separate databases for departmental or project-specific data.

Answer: D

Explanation:

The best design to store and analyze large amounts of structured and semi-structured data for different business units and departments is to use a centralized Snowflake database for core business data, and use separate databases for departmental or project-specific data. This design allows for scalability, security, and cost efficiency by leveraging Snowflake's features such as:

Database cloning: Cloning a database creates a zero-copy clone that shares the same data files as the original database, but can be modified independently. This reduces storage costs and enables fast and consistent data replication for different purposes.

Database sharing: Sharing a database allows granting secure and governed access to a subset of data in a database to other Snowflake accounts or consumers. This enables data collaboration and monetization across different business units or external partners.

Warehouse scaling: Scaling a warehouse allows adjusting the size and concurrency of a warehouse to match the performance and cost requirements of different workloads. This enables optimal resource utilization and flexibility for different data analysis needs. Reference: Snowflake Documentation: Database Cloning, Snowflake Documentation: Database Sharing, [Snowflake Documentation: Warehouse Scaling]

Question: 119

How can the Snowpipe REST API be used to keep a log of data load history?

A. Call insertReport every 20 minutes, fetching the last 10,000 entries.

B. Call loadHistoryScan every minute for the maximum time range.

C. Call insertReport every 8 minutes for a 10-minute time range.

D. Call loadHistoryScan every 10 minutes for a 15-minutes range.

Answer: D

Explanation:

The Snowpipe REST API provides two endpoints for retrieving the data load history: insertReport and loadHistoryScan. The insertReport endpoint returns the status of the files that were submitted to the insertFiles endpoint, while the loadHistoryScan endpoint returns the history of the files that were actually loaded into the table by Snowpipe. To keep a log of data load history, it is recommended to use the loadHistoryScan endpoint, which provides more accurate and complete information about the data ingestion process. The loadHistoryScan endpoint accepts a start time and an end time as parameters, and returns the files that were loaded within that time range. The maximum time range that can be specified is 15 minutes, and the maximum number of files that can be returned is 10,000. Therefore, to keep a log of data load history, the best option is to call the loadHistoryScan endpoint every 10 minutes for a 15-minute time range, and store the results in a log file or a table. This way, the log will capture all the files that were loaded by Snowpipe, and avoid any gaps or overlaps in the time range. The other options are incorrect because:

Calling insertReport every 20 minutes, fetching the last 10,000 entries, will not provide a complete log of data load history, as some files may be missed or duplicated due to the asynchronous nature of Snowpipe. Moreover, insertReport only returns the status of the files that were submitted, not the files that were loaded.

Calling loadHistoryScan every minute for the maximum time range will result in too many API calls and unnecessary overhead, as the same files will be returned multiple times. Moreover, the maximum time range is 15 minutes, not 1 minute.

Calling insertReport every 8 minutes for a 10-minute time range will suffer from the same problems as option A, and also create gaps or overlaps in the time range.

Reference:

Snowpipe REST API

Option 1: Loading Data Using the Snowpipe REST API PIPE_USAGE_HISTORY

Question: 120

Database DB1 has schema S1 which has one table, T1.

DB1 --> S1 --> T1

The retention period of EG1 is set to 10 days.

The retention period of s: is set to 20 days.

The retention period of t: Is set to 30 days.

The user runs the following command:

Drop Database DB1;

What will the Time Travel retention period be for T1?

A. 10 days
B. 20 days
C. 30 days
D. 37 days

Answer: C

Explanation:

The Time Travel retention period for T1 will be 30 days, which is the retention period set at the table level. The Time Travel retention period determines how long the historical data is preserved and accessible for an object after it is modified or dropped. The Time Travel retention period can be set at the account level, the database level, the schema level, or the table level. The retention period set at the lowest level of the hierarchy takes precedence over the higher levels. Therefore, the retention period set at the table level overrides the retention periods set at the schema level, the database level, or the account level. When the user drops the database DB1, the table T1 is also dropped, but the historical data is still preserved for 30 days, which is the retention period set at the table level. The user can use the

UNDROP command to restore the table T1 within the 30-day period. The other options are incorrect because:
10 days is the retention period set at the database level, which is overridden by the table level. 20 days is the retention period set at the schema level, which is also overridden by the table level. 37 days is not a valid option, as it is not the retention period set at any level.

Reference:
Understanding & Using Time Travel
AT | BEFORE
Snowflake Time Travel & Fail-safe

Question: 121

A global company needs to securely share its sales and Inventory data with a vendor using a Snowflake account. The company has its Snowflake account In the AWS eu-west 2 Europe (London) region. The vendor's Snowflake account Is on the Azure platform in the West Europe region. How should the company's Architect configure the data share?

A.
1. Create a share.
2. Add objects to the share.
3. Add a consumer account to the share for the vendor to access.

B.
1. Create a share.
2. Create a reader account for the vendor to use.
3. Add the reader account to the share.

C.
1. Create a new role called db_share.
2. Grant the db_share role privileges to read data from the company database and schema.
3. Create a user for the vendor.
4. Grant the ds_share role to the vendor's users. D.

1. Promote an existing database in the company's local account to primary.
2. Replicate the database to Snowflake on Azure in the West-Europe region.
3. Create a share and add objects to the share.
4. Add a consumer account to the share for the vendor to access.

Answer: A

Explanation:
The correct way to securely share data with a vendor using a Snowflake account on a different cloud platform and region is to create a share, add objects to the share, and add a consumer account to the share for the vendor to access. This way, the company can control what data is shared, who can access it, and how long the share is valid. The vendor can then query the shared data without copying or moving it to their own account. The other options are either incorrect or inefficient, as they involve creating unnecessary reader accounts, users, roles, or database replication.
https://learn.snowflake.com/en/certifications/snowpro-advanced-architect/

Question: 122

A company wants to Integrate its main enterprise identity provider with federated authentication with Snowflake. The authentication integration has been configured and roles have been created in Snowflake. However, the users are not automatically appearing in Snowflake when created and their group membership is not reflected in their assigned rotes.
How can the missing functionality be enabled with the LEAST amount of operational overhead?

A. OAuth must be configured between the identity provider and Snowflake. Then the authorization server must be configured with the right mapping of users and roles.
B. OAuth must be configured between the identity provider and Snowflake. Then the authorization server must be configured with the right mapping of users, and the resource server must be configured with the right mapping of role assignment.
C. SCIM must be enabled between the identity provider and Snowflake. Once both are synchronized through SCIM, their groups will get created as group accounts in Snowflake and the proper roles can be granted.
D. SCIM must be enabled between the identity provider and Snowflake. Once both are synchronized through SCIM. users will automatically get created and their group membership will be reflected as roles In Snowflake.

Answer: D

Explanation:

The best way to integrate an enterprise identity provider with federated authentication and enable automatic user creation and role assignment in Snowflake is to use SCIM (System for Cross-domain Identity Management). SCIM allows Snowflake to synchronize with the identity provider and create users and groups based on the information provided by the identity provider. The groups are mapped to roles in Snowflake, and the users are assigned the roles based on their group membership. This way, the identity provider remains the source of truth for user and group management, and Snowflake automatically reflects the changes without manual intervention. The other options are either incorrect or incomplete, as they involve using OAuth, which is a protocol for authorization, not authentication or user provisioning, and require additional configuration of authorization and resource servers.

Question: 123

An Architect has designed a data pipeline that Is receiving small CSV files from multiple sources. All of the files are landing in one location. Specific files are filtered for loading into Snowflake tables using the copy command. The loading performance is poor.
What changes can be made to Improve the data loading performance?

A. Increase the size of the virtual warehouse.
B. Create a multi-cluster warehouse and merge smaller files to create bigger files.
C. Create a specific storage landing bucket to avoid file scanning.
D. Change the file format from CSV to JSON.

Answer: B

Explanation:

According to the Snowflake documentation, the data loading performance can be improved by following some best practices and guidelines for preparing and staging the data files. One of the recommendations is to aim for data files that are roughly 100-250 MB (or larger) in size compressed, as this will optimize the number of parallel operations for a load. Smaller files should be aggregated and larger files should be split to achieve this size range. Another recommendation is to use a multicluster warehouse for loading, as this will allow for scaling up or out the compute resources depending on the load demand. A single-cluster warehouse may not be able to handle the load concurrency and throughput efficiently. Therefore, by creating a multi-cluster warehouse and merging smaller files to create bigger files, the data loading performance can be improved. Reference: Data Loading Considerations Preparing Your Data Files Planning a Data Load

Question: 124

A table, EMP_ TBL has three records as shown:

```
create or replace TABLE EMP_TBL (
ID NUMBER(38,0),
NAME VARCHAR(16777216)
);
```

ID	NAME
1	Name1
2	Name2
3	Name3

The following variables are set for the session:

```
set tbl_ref = 'EMP_TBL';
set col_ref = 'NAME';
set (var1, var2, var3) = (select 'Name1', 'Name2', 'Name3')
```

Which SELECT statements will retrieve all three records? (Select TWO).

A. Select * FROM Stbl_ref WHERE Scol_ref IN ('Name1','Nam2','Name3');
B. SELECT * FROM EMP_TBL WHERE identifier(Scol_ref) IN ('Name1','Name2', 'Name3');
C. SELECT * FROM identifier<Stbl_ref> WHERE NAME IN ($var1, $var2, $var3);
D. SELECT * FROM identifier($tbl_ref) WHERE ID IN Cvarl','var2','var3');
E. SELECT * FROM $tb1_ref WHERE $col_ref IN ($var1, Svar2, Svar3);

Answer: B, E

Explanation:

The correct answer is B and E because they use the correct syntax and values for the identifier function and the session variables.

The identifier function allows you to use a variable or expression as an identifier (such as a table name or column name) in a SQL statement. It takes a single argument and returns it as an identifier. For example, identifier($tbl_ref) returns EMP_TBL as an identifier.

The session variables are set using the SET command and can be referenced using the $ sign. For example, $var1 returns Name1 as a value.

Option A is incorrect because it uses Stbl_ref and Scol_ref, which are not valid session variables or identifiers. They should be $tbl_ref and $col_ref instead.

Option C is incorrect because it uses identifier<Stbl_ref>, which is not a valid syntax for the identifier function. It should be identifier($tbl_ref) instead.

Option D is incorrect because it uses Cvarl, var2, and var3, which are not valid session variables or values. They should be $var1, $var2, and $var3 instead. Reference:

Snowflake Documentation: Identifier Function

Snowflake Documentation: Session Variables

Snowflake Learning: SnowPro Advanced: Architect Exam Study Guide

Question: 125

A data platform team creates two multi-cluster virtual warehouses with the AUTO_SUSPEND value set to NULL on one.

and '0' on the other. What would be the execution behavior of these virtual warehouses?

A. Setting a '0' or NULL value means the warehouses will never suspend.
B. Setting a '0' or NULL value means the warehouses will suspend immediately.
C. Setting a '0' or NULL value means the warehouses willsuspend after the default of 600seconds.
D. Setting a '0' value means the warehouses will suspend immediately, and NULLmeans the warehouses will never suspend.

Answer: D

Explanation:

The AUTO_SUSPEND parameter controls the amount of time, in seconds, of inactivity after which a warehouse is automatically suspended. If the parameter is set to NULL, the warehouse never suspends. If the parameter is set to '0', the warehouse suspends immediately after executing a query. Therefore, the execution behavior of the two virtual warehouses will be different depending on the AUTO_SUSPEND value. The warehouse with NULL value will keep running until it is manually suspended or the resource monitor limits are reached. The warehouse with '0' value will suspend as

soon as it finishes a query and release the compute resources. Reference:
ALTER WAREHOUSE
Parameters

Question: 126

A user named USER_01 needs access to create a materialized view on a schema EDW. STG_SCHEM
A. How can this access be provided?
A. GRANT CREATE MATERIALIZED VIEW ON SCHEMA EDW.STG_SCHEMA TO USER USER_01;
B. GRANT CREATE MATERIALIZED VIEW ON DATABASE EDW TO USER USERJD1;
C. GRANT ROLE NEW_ROLE TO USER USER_01;
GRANT CREATE MATERIALIZED VIEW ON SCHEMA ECW.STG_SCHEKA TO NEW_ROLE;
D. GRANT ROLE NEW_ROLE TO USER_01;
GRANT CREATE MATERIALIZED VIEW ON EDW.STG_SCHEMA TO NEW_ROLE;

Answer: A

Explanation:

The correct answer is A because it grants the specific privilege to create a materialized view on the schema EDW.STG_SCHEMA to the user USER_01 directly.
Option B is incorrect because it grants the privilege to create a materialized view on the entire database EDW, which is too broad and unnecessary. Also, there is a typo in the user name (USERJD1 instead of USER_01).
Option C is incorrect because it grants the privilege to create a materialized view on a different schema (ECW.STG_SCHEKA instead of EDW.STG_SCHEMA). Also, there is no need to create a new role for this purpose.
Option D is incorrect because it grants the privilege to create a materialized view on an invalid object (EDW.STG_SCHEMA is not a valid schema name, it should be EDW.STG_SCHEMA). Also, there is no need to create a new role for this purpose. Reference: Snowflake Documentation: CREATE MATERIALIZED VIEW
Snowflake Documentation: Working with Materialized Views
[Snowflake Documentation: GRANT Privileges on a Schema]

Question: 127

An Architect Is designing a data lake with Snowflake. The company has structured, semi-structured, and unstructured dat
a. The company wants to save the data inside the data lake within the Snowflake system. The company is planning on sharing data among Its corporate branches using Snowflake data sharing. What should be considered when sharing the unstructured data within Snowflake?

A. A pre-signed URL should be used to save the unstructured data into Snowflake in order to share data over secure views, with no time limit for the URL.

B. A scoped URL should be used to save the unstructured data into Snowflake in order to share data over secure views, with a 24-hour time limit for the URL.

C. A file URL should be used to save the unstructured data into Snowflake in order to share data over secure views, with a 7-day time limit for the URL.

D. A file URL should be used to save the unstructured data into Snowflake in order to share data over secure views, with the "expiration_time" argument defined for the URL time limit.

Answer: D

Explanation:

According to the Snowflake documentation, unstructured data files can be shared by using a secure view and Secure Data Sharing. A secure view allows the result of a query to be accessed like a table, and a secure view is specifically designated for data privacy. A scoped URL is an encoded URL that permits temporary access to a staged file without granting privileges to the stage. The URL expires when the persisted query result period ends, which is currently 24 hours. A scoped URL is recommended for file administrators to give scoped access to data files to specific roles in the same account. Snowflake records information in the query history about who uses a scoped URL to access a file, and when. Therefore, a scoped URL is the best option to share unstructured data within Snowflake, as it provides security, accountability, and control over the data access. Reference: Sharing unstructured Data with a secure view Introduction to Loading Unstructured Data

Question: 128

Consider the following scenario where a masking policy is applied on the CREDICARDND column of the CREDITCARDINFO table. The masking policy definition Is as follows:

```
create or replace masking policy creditcardnojnask as (val string) returns string -> case
when i3_rcle_in_session('PI_ANALYTIC3') then right(val,4)
else ••"MASKED"”
```

Sample data for the CREDITCARDINFO table is as follows: NAME EXPIRYDATE CREDITCARDNO
JOHN DOE 2022-07-23 4321 5678 9012 1234
if the Snowflake system rotes have not been granted any additional roles, what will be the result?

A. The sysadmin can see the CREDICARDND column data in clear text.

B. The owner of the table will see the CREDICARDND column data in clear text.

C. Anyone with the PI_ANALYTICS role will see the last 4 characters of the CREDICARDND column data in dear text.

D. Anyone with the PI_ANALYTICS role will see the CREDICARDND column as*** 'MASKED* **'.

Answer: D

Explanation:

The masking policy defined in the image indicates that if a user has the PI_ANALYTICS role, they will be able to see the last 4 characters of the CREDITCARDNO column data in clear text. Otherwise, they will see 'MASKED'. Since Snowflake system roles have not been granted any additional roles, they won't have the PI_ANALYTICS role and therefore cannot view the last 4 characters of credit card numbers.

To apply a masking policy on a column in Snowflake, you need to use the ALTER TABLE … ALTER COLUMN command or the ALTER VIEW command and specify the policy name. For example, to apply

the creditcardno_mask policy on the CREDITCARDNO column of the CREDITCARDINFO table, you can use the following command:

ALTER TABLE CREDITCARDINFO ALTER COLUMN CREDITCARDNO SET MASKING POLICY creditcardno_mask;

For more information on how to create and use masking policies in Snowflake, you can refer to the following resources:

CREATE MASKING POLICY: This document explains the syntax and usage of the CREATE MASKING POLICY command, which allows you to create a new masking policy or replace an existing one. Using Dynamic Data Masking: This guide provides instructions on how to configure and use dynamic data masking in Snowflake, which is a feature that allows you to mask sensitive data based on the execution context of the user.

ALTER MASKING POLICY: This document explains the syntax and usage of the ALTER MASKING POLICY command, which allows you to modify the properties of an existing masking policy. Reference: 1: https://docs.snowflake.com/en/sql-reference/sql/create-masking-policy 2: https://docs.snowflake.com/en/user-guide/security-column-ddm-use 3: https://docs.snowflake.com/en/sql-reference/sql/alter-masking-policy

Question: 129

A company is trying to Ingest 10 TB of CSV data into a Snowflake table using Snowpipe as part of Its migration from a legacy database platform. The records need to be ingested in the MOST performant and cost-effective way.
How can these requirements be met?

A. Use ON_ERROR = continue in the copy into command.
B. Use purge = TRUE in the copy into command.
C. Use FURGE = FALSE in the copy into command.
D. Use on error = SKIP_FILE in the copy into command.

Answer: D

Explanation:

For ingesting a large volume of CSV data into Snowflake using Snowpipe, especially for a substantial amount like 10 TB, the on error = SKIP_FILE option in the COPY INTO command can be highly effective. This approach allows Snowpipe to skip over files that cause errors during the ingestion process, thereby not halting or significantly slowing down the overall data load. It helps in maintaining performance and cost-effectiveness by avoiding the reprocessing of problematic files and continuing with the ingestion of other data.

Question: 130

An Architect is troubleshooting a query with poor performance using the QUERY function. The Architect observes that the COMPILATION_TIME Is greater than the EXECUTION_TIME.
What is the reason for this?

A. The query is processing a very large dataset.
B. The query has overly complex logic.

C. The query Is queued for execution.
D. The query Is reading from remote storage

Answer: B

Explanation:

The correct answer is B because the compilation time is the time it takes for the optimizer to create an optimal query plan for the efficient execution of the query. The compilation time depends on the complexity of the query, such as the number of tables, columns, joins, filters, aggregations, subqueries, etc. The more complex the query, the longer it takes to compile.
Option A is incorrect because the query processing time is not affected by the size of the dataset, but by the size of the virtual warehouse. Snowflake automatically scales the compute resources to match the data volume and parallelizes the query execution. The size of the dataset may affect the execution time, but not the compilation time.
Option C is incorrect because the query queue time is not part of the compilation time or the execution time. It is a separate metric that indicates how long the query waits for a warehouse slot before it starts running. The query queue time depends on the warehouse load, concurrency, and priority settings.
Option D is incorrect because the query remote IO time is not part of the compilation time or the execution time. It is a separate metric that indicates how long the query spends reading data from remote storage, such as S3 or Azure Blob Storage. The query remote IO time depends on the network latency, bandwidth, and caching efficiency.
Reference:
Understanding Why Compilation Time in Snowflake Can Be Higher than Execution Time: This article explains why the total duration (compilation + execution) time is an essential metric to measure query performance in Snowflake. It discusses the reasons for the long compilation time, including query complexity and the number of tables and columns.
Exploring Execution Times: This document explains how to examine the past performance of queries and tasks using Snowsight or by writing queries against views in the ACCOUNT_USAGE schema. It also describes the different metrics and dimensions that affect query performance, such as duration, compilation, execution, queue, and remote IO time.
What is the “compilation time” and how to optimize it?: This community post provides some tips and best practices on how to reduce the compilation time, such as simplifying the query logic, using views or common table expressions, and avoiding unnecessary columns or joins.

Question: 131

A user is executing the following command sequentially within a timeframe of 10 minutes from start to finish:

```
use role syjadttin;
use warehouse compute_wh;
use schema sales.public*
create table tjil«> (numeric integer) data_retention_time_in_days«l;
create or replace table tsalescior.e clone t sales at (offset ■> -€0*30)
```

What would be the output of this query?

A. Table T_SALES_CLONE successfully created.
B. Time Travel data is not available for table T_SALES.
C. The offset -> is not a valid clause in the clone operation.
D. Syntax error line 1 at position 58 unexpected 'at’.

Answer: A

Explanation:

The query is executing a clone operation on an existing table t_sales with an offset to account for the retention time. The syntax used is correct for cloning a table in Snowflake, and the use of the at(offset => -60*30) clause is valid. This specifies that the clone should be based on the state of the table 30 minutes prior (60 seconds * 30). Assuming the table t_sales exists and has been modified within the last 30 minutes, and considering the data_retention_time_in_days is set to 1 day (which enables time travel queries for the past 24 hours), the table t_sales_clone would be successfully created based on the state of t_sales 30 minutes before the clone command was issued.

Question: 132

A Snowflake Architect Is working with Data Modelers and Table Designers to draft an ELT framework specifically for data loading using Snowpipe. The Table Designers will add a timestamp column that Inserts the current tlmestamp as the default value as records are loaded into a table. The Intent is to capture the time when each record gets loaded into the table; however, when tested the timestamps are earlier than the loae_take column values returned by the copy_history function or the Copy_HISTORY view (Account Usage).
Why Is this occurring?

A. The timestamps are different because there are parameter setup mismatches. The parameters need to be realigned
B. The Snowflake timezone parameter Is different from the cloud provider's parameters causing the mismatch.
C. The Table Designer team has not used the localtimestamp or systimestamp functions in the Snowflake copy statement.
D. The CURRENT_TIMEis evaluated when the load operation is compiled in cloud services rather than when the record is inserted into the table.

Answer: D

Explanation:

The correct answer is D because the CURRENT_TIME function returns the current timestamp at the start of the statement execution, not at the time of the record insertion. Therefore, if the load operation takes some time to complete, the CURRENT_TIME value may be earlier than the actual load time.
Option A is incorrect because the parameter setup mismatches do not affect the timestamp values. The parameters are used to control the behavior and performance of the load operation, such as the file format, the error handling, the purge option, etc.
Option B is incorrect because the Snowflake timezone parameter and the cloud provider's parameters are independent of each other. The Snowflake timezone parameter determines the session timezone for displaying and converting timestamp values, while the cloud provider's parameters determine the physical location and configuration of the storage and compute resources.

Option C is incorrect because the localtimestamp and systimestamp functions are not relevant for the Snowpipe load operation. The localtimestamp function returns the current timestamp in the session timezone, while the systimestamp function returns the current timestamp in the system timezone. Neither of them reflect the actual load time of the records. Reference:
Snowflake Documentation: Loading Data Using Snowpipe: This document explains how to use Snowpipe to continuously load data from external sources into Snowflake tables. It also describes the syntax and usage of the COPY INTO command, which supports various options and parameters to control the loading behavior.

Snowflake Documentation: Date and Time Data Types and Functions: This document explains the different data types and functions for working with date and time values in Snowflake. It also describes how to set and change the session timezone and the system timezone.
Snowflake Documentation: Querying Metadata: This document explains how to query the metadata of the objects and operations in Snowflake using various functions, views, and tables. It also describes how to access the copy history information using the COPY_HISTORY function or the COPY_HISTORY view.

Question: 133

An Architect needs to automate the daily Import of two files from an external stage into Snowflake. One file has Parquet-formatted data, the other has CSV-formatted data.
How should the data be joined and aggregated to produce a final result set?

A. Use Snowpipe to ingest the two files, then create a materialized view to produce the final result set.
B. Create a task using Snowflake scripting that will import the files, and then call a User-Defined Function (UDF) to produce the final result set.
C. Create a JavaScript stored procedure to read. join, and aggregate the data directly from the external stage, and then store the results in a table.
D. Create a materialized view to read, Join, and aggregate the data directly from the external stage, and use the view to produce the final result set

Answer: B

Explanation:

According to the Snowflake documentation, tasks are objects that enable scheduling and execution of SQL statements or JavaScript user-defined functions (UDFs) in Snowflake. Tasks can be used to automate data loading, transformation, and maintenance operations. Snowflake scripting is a feature that allows writing procedural logic using SQL statements and JavaScript UDFs. Snowflake scripting can be used to create complex workflows and orchestrate tasks. Therefore, the best option to automate the daily import of two files from an external stage into Snowflake, join and aggregate the data, and produce a final result set is to create a task using Snowflake scripting that will import the files using the COPY INTO command, and then call a UDF to perform the join and aggregation logic. The UDF can return a table or a variant value as the final result set. Reference:

Tasks
Snowflake Scripting
User-Defined Functions

Question: 134

A company has a source system that provides JSON records for various IoT operations. The JSON Is loading directly into a persistent table with a variant field. The data Is quickly growing to 100s of millions of records and performance to becoming an issue. There is a generic access pattern that Is used to filter on the create_date key within the variant field. What can be done to improve performance?

A. Alter the target table to Include additional fields pulled from the JSON records. This would Include a create_date field with a datatype of time stamp. When this field Is used in the filter, partition pruning will occur.
B. Alter the target table to include additional fields pulled from the JSON records. This would include a create_date field with a datatype of varchar. When this field is used in the filter, partition pruning will occur.
C. Validate the size of the warehouse being used. If the record count is approaching 100s of millions, size XL will be the minimum size required to process this amount of data.

D. Incorporate the use of multiple tables partitioned by date ranges. When a user or process needs to query a particular date range, ensure the appropriate base table Is used.

Answer: A

Explanation:

The correct answer is A because it improves the performance of queries by reducing the amount of data scanned and processed. By adding a create_date field with a timestamp data type, Snowflake can automatically cluster the table based on this field and prune the micro-partitions that do not match the filter condition. This avoids the need to parse the JSON data and access the variant field for every record.

Option B is incorrect because it does not improve the performance of queries. By adding a create_date field with a varchar data type, Snowflake cannot automatically cluster the table based on this field and prune the micro-partitions that do not match the filter condition. This still requires parsing the JSON data and accessing the variant field for every record.

Option C is incorrect because it does not address the root cause of the performance issue. By validating the size of the warehouse being used, Snowflake can adjust the compute resources to match the data volume and parallelize the query execution. However, this does not reduce the amount of data scanned and processed, which is the main bottleneck for queries on JSON data. Option D is incorrect because it adds unnecessary complexity and overhead to the data loading and querying process. By incorporating the use of multiple tables partitioned by date ranges, Snowflake can reduce the amount of data scanned and processed for queries that specify a date range.

However, this requires creating and maintaining multiple tables, loading data into the appropriate table based on the date, and joining the tables for queries that span multiple date ranges. Reference: Snowflake Documentation: Loading Data Using Snowpipe: This document explains how to use Snowpipe to continuously load data from external sources into Snowflake tables. It also describes the syntax and usage of the COPY INTO command, which supports various options and parameters to control the loading behavior, such as ON_ERROR, PURGE, and SKIP_FILE.

Snowflake Documentation: Date and Time Data Types and Functions: This document explains the different data types and functions for working with date and time values in Snowflake. It also describes how to set and change the session timezone and the system timezone.

Snowflake Documentation: Querying Metadata: This document explains how to query the metadata of the objects and operations in Snowflake using various functions, views, and tables. It also describes how to access the copy history information using the COPY_HISTORY function or the COPY_HISTORY view.

Snowflake Documentation: Loading JSON Data: This document explains how to load JSON data into Snowflake tables using various methods, such as the COPY INTO command, the INSERT command, or the PUT command. It also describes how to access and query JSON data using the dot notation, the FLATTEN function, or the LATERAL join.

Snowflake Documentation: Optimizing Storage for Performance: This document explains how to optimize the storage of data in Snowflake tables to improve the performance of queries. It also describes the concepts and benefits of automatic clustering, search optimization service, and materialized views.

Question: 135

An Architect has a design where files arrive every 10 minutes and are loaded into a primary database table using Snowpipe. A secondary database is refreshed every hour with the latest data from the primary database.

Based on this scenario, what Time Travel query options are available on the secondary database?

A. A query using Time Travel in the secondary database is available for every hourly table version within the retention window.

B. A query using Time Travel in the secondary database is available for every hourly table version within and outside the retention window.

C. Using Time Travel, secondary database users can query every iterative version within each hour (the individual Snowpipe loads) in the retention window.

D. Using Time Travel, secondary database users can query every iterative version within each hour (the individual Snowpipe loads) and outside the retention window.

Answer: A

Explanation:

Snowflake's Time Travel feature allows users to query historical data within a defined retention period. In the given scenario, since the secondary database is refreshed every hour, Time Travel can be used to query each hourly version of the table as long as it falls within the retention window. This does not include individual Snowpipe loads within each hour unless they coincide with the hourly refresh.

Reference: The answer is verified using Snowflake's official documentation, which provides detailed information on Time Travel and its usage within the retention period123.

Question: 136

An Architect for a multi-national transportation company has a system that is used to check the weather conditions along vehicle routes. The data is provided to drivers.

The weather information is delivered regularly by a third-party company and this information is generated as JSON structure. Then the data is loaded into Snowflake in a column with a VARIANT data type. This table is directly queried to deliver the statistics to the drivers with minimum time lapse.

A single entry includes (but is not limited to):

- Weather condition; cloudy, sunny, rainy, etc.
- Degree
- Longitude and latitude
- Timeframe
- Location address
- Wind

The table holds more than 10 years' worth of data in order to deliver the statistics from different years and locations. The amount of data on the table increases every day.

The drivers report that they are not receiving the weather statistics for their locations in time.

What can the Architect do to deliver the statistics to the drivers faster?

A. Create an additional table in the schema for longitude and latitude. Determine a regular task to fill this

information by extracting it from the JSON dataset.

B. Add search optimization service on the variant column for longitude and latitude in order to query the information by using specific metadata.

C. Divide the table into several tables for each year by using the timeframe information from the JSON dataset in order to process the queries in parallel.

D. Divide the table into several tables for each location by using the location address information from the JSON dataset in order to process the queries in parallel.

Answer: B

Explanation:

To improve the performance of queries on semi-structured data, such as JSON stored in a VARIANT column, Snowflake's search optimization service can be utilized. By adding search optimization specifically for the longitude and latitude fields within the VARIANT column, the system can perform point lookups and substring queries more efficiently. This will allow for faster retrieval of weather statistics, which is critical for the drivers to receive timely updates.
Reference: The solution is supported by Snowflake documentation that details how search optimization can enhance query performance for semi-structured data1.

Question: 137

A new table and streams are created with the following commands:

```
CREATE OR REPLACE TABLE LETTERS (ID INT, LETTER STRING) ;

CREATE OR REPLACE STREAM STREAM_1 ON TABLE LETTERS;
CREATE OR REPLACE STREAM STREAM_2 ON TABLE LETTERS APPEND_ONLY = TRUE;
```

The following operations are processed on the newly created table:

```
INSERT INTO LETTERS VALUES (1, 'A');
INSERT INTO LETTERS VALUES (2, 'B');
INSERT INTO LETTERS VALUES (3, 'C');

TRUNCATE TABLE LETTERS;

INSERT INTO LETTERS VALUES (4, 'D');
INSERT INTO LETTERS VALUES (5, 'E');
INSERT INTO LETTERS VALUES (6, 'F');
DELETE FROM LETTERS WHERE ID = 6;
```

What would be the output of the following SQL commands, in order?

```
SELECT COUNT (*) FROM STREAM_1;
SELECT COUNT (*) FROM STREAM_2;
```

A. 2 & 6

B. 2 & 3

C. 4 & 3

D. 4 & 6

Answer: C

Explanation:

In Snowflake, a stream records data manipulation language (DML) changes to its base table since the stream was created or last consumed. STREAM_1 will show all changes including the TRUNCATE operation, while STREAM_2, being APPEND_ONLY, will not show deletions like TRUNCATE. Therefore, STREAM_1 will count the three inserts, the TRUNCATE (counted as a single operation), and the subsequent two inserts before the delete, totaling 4. STREAM_2 will only count the three initial inserts and the two after the TRUNCATE, totaling 3, as it does not count the TRUNCATE or the delete operation.

Reference: The explanation is based on the Snowflake documentation on streams, which details how streams track changes and the difference between standard and APPEND_ONLY streams12.

Question: 138

When loading data into a table that captures the load time in a column with a default value of either CURRENT_TIME () or CURRENT_TIMESTAMP () what will occur?

A. All rows loaded using a specific COPY statement will have varying timestamps based on when the rows were inserted.

B. Any rows loaded using a specific COPY statement will have varying timestamps based on when the rows were read from the source.

C. Any rows loaded using a specific COPY statement will have varying timestamps based on when the rows were created in the source.

D. All rows loaded using a specific COPY statement will have the same timestamp value.

Answer: D

Explanation:

When using the COPY command to load data into Snowflake, if a column has a default value set to CURRENT_TIME() or CURRENT_TIMESTAMP(), all rows loaded by that specific COPY command will have the same timestamp. This is because the default value for the timestamp is evaluated at the start of the COPY operation, and that same value is applied to all rows loaded by that operation. Reference: This behavior is consistent with Snowflake's documentation on the CURRENT_TIMESTAMP function, which specifies that the timestamp is captured at the time the statement is executed1.

Question: 139

In a managed access schema, what are characteristics of the roles that can manage object privileges? (Select TWO).

A. Users with the SYSADMIN role can grant object privileges in a managed access schema.

B. Users with the SECURITYADMIN role or higher, can grant object privileges in a managed access schema.

C. Users who are database owners can grant object privileges in a managed access schema.

D. Users who are schema owners can grant object privileges in a managed access schema.

E. Users who are object owners can grant object privileges in a managed access schema.

Answer: BD

Explanation:

In a managed access schema, the privilege management is centralized with the schema owner, who has the authority to grant object privileges within the schema. Additionally, the SECURITYADMIN role has the capability to manage object grants globally, which includes within managed access schemas. Other roles, such as SYSADMIN or database owners, do not inherently have this privilege unless explicitly granted.

Reference: The verified answers are based on Snowflake's official documentation, which outlines the roles and privileges associated with managed access schemas12.

Question: 140

An Architect is designing a data lake with Snowflake. The company has structured, semi-structured, and unstructured dat
a. The company wants to save the data inside the data lake within the Snowflake
system. The company is planning on sharing data among its corporate branches using Snowflake data sharing.

What should be considered when sharing the unstructured data within Snowflake?

A. A pre-signed URL should be used to save the unstructured data into Snowflake in order to share data over secure views, with no time limit for the URL.

B. A scoped URL should be used to save the unstructured data into Snowflake in order to share data over secure views, with a 24-hour time limit for the URL.

C. A file URL should be used to save the unstructured data into Snowflake in order to share data over secure views, with a 7-day time limit for the URL.

D. A file URL should be used to save the unstructured data into Snowflake in order to share data over secure views, with the "expiration_time" argument defined for the URL time limit.

Answer: B

Explanation:

When sharing unstructured data within Snowflake, using a scoped URL is recommended. Scoped URLs provide temporary access to staged files without granting privileges to the stage itself, enhancing security. The URL expires when the persisted query result period ends, which is currently set to 24 hours. This approach is suitable for sharing unstructured data over secure views within Snowflake's data sharing framework.

Reference: The answer is based on Snowflake's official documentation regarding the sharing of unstructured data and the use of scoped URLs1.

Question: 141

An Architect is using SnowCD to investigate a connectivity issue.

Which system function will provide a list of endpoints that the network must be able to access to use a specific Snowflake account, leveraging private connectivity?

A. SYSTEMSALLOWLIST ()

B. SYSTEMSGET_PRIVATELINK

C. SYSTEMSAUTHORIZE_PRIVATELINK

D. SYSTEMSALLOWLIST_PRIVATELINK ()

Answer: D

Explanation:

Question: 142

An Architect with the ORGADMIN role wants to change a Snowflake account from an Enterprise edition to a Business Critical edition.

How should this be accomplished?

A. Run an ALTER ACCOUNT command and create a tag of EDITION and set the tag to Business Critical.

B. Use the account's ACCOUNTADMIN role to change the edition.

C. Failover to a new account in the same region and specify the new account's edition upon creation.

D. Contact Snowflake Support and request that the account's edition be changed.

Answer: D

Explanation:

To change the edition of a Snowflake account, an organization administrator (ORGADMIN) cannot directly alter the account settings through SQL commands or the Snowflake interface. The proper procedure is to contact Snowflake Support to request an edition change for the account. This ensures that the change is managed correctly and aligns with Snowflake's operational protocols.
Reference: This process is outlined in the Snowflake documentation, which specifies that changes to an account's edition should be facilitated through Snowflake Support1.

Question: 143

An Architect is implementing a CI/CD process. When attempting to clone a table from a production to a development environment, the cloning operation fails.

What could be causing this to happen?

A. The table is transient.

B. The table has a masking policy.

C. The retention time for the table is set to zero.

D. Tables cannot be cloned from a higher environment to a lower environment.

Answer: B

Explanation:

Cloning a table with a masking policy can cause the cloning operation to fail because the masking policy is not automatically cloned with the table. This is due to the fact that the masking policy is considered a separate object with its own set of privileges1.
Reference:
Snowflake Documentation on Cloning Considerations1.

Question: 144

An Architect needs to design a solution for building environments for development, test, and preproduction, all located in a single Snowflake account. The environments should be based on production data.

Which solution would be MOST cost-effective and performant?

A. Use zero-copy cloning into transient tables.

B. Use zero-copy cloning into permanent tables.

C. Use CREATE TABLE ... AS SELECT (CTAS) statements.

D. Use a Snowflake task to trigger a stored procedure to copy data.

Answer: A

Explanation:

Zero-copy cloning is a feature in Snowflake that allows for the creation of a clone of a database, schema, or table without duplicating any data, which is cost-effective as it saves on storage costs. Transient tables are temporary and do not incur storage costs for the time they are not accessed,

making them a cost-effective option for development, test, and pre-production environments that do not require the durability of permanent tables123.

Reference:

- Snowflake Documentation on Zero-Copy Cloning3.
- Articles discussing the cost-effectiveness and performance benefits of zero-copy cloning12.

Question: 145

Which Snowflake architecture recommendation needs multiple Snowflake accounts for implementation?

A. Enable a disaster recovery strategy across multiple cloud providers.

B. Create external stages pointing to cloud providers and regions other than the region hosting the Snowflake account.

C. Enable zero-copy cloning among the development, test, and production environments.

D. Enable separation of the development, test, and production environments.

Answer: D

Explanation:

The Snowflake architecture recommendation that necessitates multiple Snowflake accounts for implementation is the separation of development, test, and production environments. This approach, known as Account per Tenant (APT), isolates tenants into separate Snowflake accounts, ensuring dedicated resources and security isolation12.

Reference:

- Snowflake's white paper on "Design Patterns for Building Multi-Tenant Applications on Snowflake" discusses the APT model and its requirement for separate Snowflake accounts for each tenant1.
- Snowflake Documentation on Secure Data Sharing, which mentions the possibility of sharing data across multiple accounts3.

Question: 146

A company is designing its serving layer for data that is in cloud storage. Multiple terabytes of the data will be used for reporting. Some data does not have a clear use case but could be useful for experimental analysis. This experimentation data changes frequently and is sometimes wiped out and replaced completely in a few days.

The company wants to centralize access control, provide a single point of connection for the endusers, and maintain data governance.

What solution meets these requirements while MINIMIZING costs, administrative effort, and development overhead?

A. Import the data used for reporting into a Snowflake schema with native tables. Then create external tables pointing to the cloud storage folders used for the experimentation data. Then create

two different roles with grants to the different datasets to match the different user personas, and grant these roles to the corresponding users.

B. Import all the data in cloud storage to be used for reporting into a Snowflake schema with native tables. Then create a role that has access to this schema and manage access to the data through that role.

C. Import all the data in cloud storage to be used for reporting into a Snowflake schema with native tables. Then create two different roles with grants to the different datasets to match the different user personas, and grant these roles to the corresponding users.

D. Import the data used for reporting into a Snowflake schema with native tables. Then create views that have SELECT commands pointing to the cloud storage files for the experimentation data. Then create two different roles to match the different user personas, and grant these roles to the corresponding users.

Answer: A

Explanation:

The most cost-effective and administratively efficient solution is to use a combination of native and external tables. Native tables for reporting data ensure performance and governance, while external tables allow for flexibility with frequently changing experimental data. Creating roles with specific grants to datasets aligns with the principle of least privilege, centralizing access control and simplifying user management12.

Reference:

- Snowflake Documentation on Optimizing Cost1.
- Snowflake Documentation on Controlling Cost2.

Question: 147

A Developer is having a performance issue with a Snowflake query. The query receives up to 10 different values for one parameter and then performs an aggregation over the majority of a fact table. It then
joins against a smaller dimension table. This parameter value is selected by the different query users when they execute it during business hours. Both the fact and dimension tables are loaded with new data in an overnight import process.

On a Small or Medium-sized virtual warehouse, the query performs slowly. Performance is acceptable on a size Large or bigger warehouse. However, there is no budget to increase costs. The Developer needs a recommendation that does not increase compute costs to run this query.

What should the Architect recommend?

A. Create a task that will run the 10 different variations of the query corresponding to the 10 different parameters before the users come in to work. The query results will then be cached and ready to respond quickly when the users re-issue the query.

B. Create a task that will run the 10 different variations of the query corresponding to the 10 different parameters before the users come in to work. The task will be scheduled to align with the users' working hours in order to allow the

warehouse cache to be used.

C. Enable the search optimization service on the table. When the users execute the query, the search optimization service will automatically adjust the query execution plan based on the frequently-used parameters.

D. Create a dedicated size Large warehouse for this particular set of queries. Create a new role that has USAGE permission on this warehouse and has the appropriate read permissions over the fact and dimension tables. Have users switch to this role and use this warehouse when they want to access this data.

Answer: C

Explanation:

Enabling the search optimization service on the table can improve the performance of queries that have selective filtering criteria, which seems to be the case here. This service optimizes the execution of queries by creating a persistent data structure called a search access path, which allows some micro-partitions to be skipped during the scanning process. This can significantly speed up query performance without increasing compute costs1.

Reference:

- Snowflake Documentation on Search Optimization Service1.

Question: 148

Role A has the following permissions:
. USAGE on db1
. USAGE and CREATE VIEW on schemal in db1
. SELECT on tablel in schemal

Role B has the following permissions:
. USAGE on db2
. USAGE and CREATE VIEW on schema2 in db2
. SELECT on table2 in schema2

A user has Role A set as the primary role and Role B as a secondary role.

What command will fail for this user?

A. use database db1;
use schema schemal;
create view v1 as select * from db2.schema2.table2;

B. use database db2;
use schema schema2;
create view v2 as select * from dbl.schemal. tablel;

C. use database db2;
use schema schema2;
select * from db1.schemal.tablel union select * from table2;

D. use database db1;
use schema schemal;

select * from db2.schema2.table2;

Answer: B

Explanation:

Question: 149

What actions are permitted when using the Snowflake SQL REST API? (Select TWO).

A. The use of a GET command

B. The use of a PUT command

C. The use of a ROLLBACK command

D. The use of a CALL command to a stored procedure which returns a table

E. Submitting multiple SQL statements in a single call

Answer: DE

Explanation:

Question: 150

What is the MOST efficient way to design an environment where data retention is not considered critical, and customization needs are to be kept to a minimum?

A. Use a transient database.

B. Use a transient schema.

C. Use a transient table.

D. Use a temporary table.

Answer: A

Explanation:

Question: 151

A retailer's enterprise data organization is exploring the use of Data Vault 2.0 to model its data lake solution. A Snowflake Architect has been asked to provide recommendations for using Data Vault 2.0 on Snowflake.

What should the Architect tell the data organization? (Select TWO).

A. Change data capture can be performed using the Data Vault 2.0 HASH_DIFF concept.

B. Change data capture can be performed using the Data Vault 2.0 HASH_DELTA concept.

C. Using the multi-table insert feature in Snowflake, multiple Point-in-Time (PIT) tables can be loaded in parallel

from a single join query from the data vault.

D. Using the multi-table insert feature, multiple Point-in-Time (PIT) tables can be loaded sequentially from a single join query from the data vault.

E. There are performance challenges when using Snowflake to load multiple Point-in-Time (PIT) tables in parallel from a single join query from the data vault.

Answer: A, C

Explanation:

Data Vault 2.0 on Snowflake supports the HASH_DIFF concept for change data capture, which is a method to detect changes in the data by comparing the hash values of the records. Additionally, Snowflake's multi-table insert feature allows for the loading of multiple PIT tables in parallel from a single join query, which can significantly streamline the data loading process and improve performance1.

Reference: =

- Snowflake's documentation on multi-table inserts1
- Blog post on optimizing Data Vault architecture on Snowflake2

Question: 152

A company is designing a process for importing a large amount of IoT JSON data from cloud storage into Snowflake. New sets of IoT data get generated and uploaded approximately every 5 minutes.

Once the IoT data is in Snowflake, the company needs up-to-date information from an external vendor to join to the dat

a. This data is then presented to users through a dashboard that shows different levels of aggregation. The external vendor is a Snowflake customer.

What solution will MINIMIZE complexity and MAXIMIZE performance?

A. 1. Create an external table over the JSON data in cloud storage.

2. Create a task that runs every 5 minutes to run a transformation procedure on new data, based on a saved timestamp.

3. Ask the vendor to expose an API so an external function can be used to generate a call to join the data back to the IoT data in the transformation procedure.

4. Give the transformed table access to the dashboard tool.

5. Perform the aggregations on the dashboard tool.

B. 1. Create an external table over the JSON data in cloud storage.

2. Create a task that runs every 5 minutes to run a transformation procedure on new data based on a saved timestamp.

3. Ask the vendor to create a data share with the required data that can be imported into the company's Snowflake account.

4. Join the vendor's data back to the IoT data using a transformation procedure.

5. Create views over the larger dataset to perform the aggregations required by the dashboard.

6. Give the views access to the dashboard tool.

C. 1. Create a Snowpipe to bring the JSON data into Snowflake.

2. Use streams and tasks to trigger a transformation procedure when new JSON data arrives.
3. Ask the vendor to expose an API so an external function call can be made to join the vendor's data back to the IoT data in a transformation procedure.
4. Create materialized views over the larger dataset to perform the aggregations required by the dashboard.
5. Give the materialized views access to the dashboard tool.

D. 1. Create a Snowpipe to bring the JSON data into Snowflake.
2. Use streams and tasks to trigger a transformation procedure when new JSON data arrives.
3. Ask the vendor to create a data share with the required data that is then imported into the Snowflake account.
4. Join the vendor's data back to the IoT data in a transformation procedure
5. Create materialized views over the larger dataset to perform the aggregations required by the dashboard.
6. Give the materialized views access to the dashboard tool.

Answer: D

Explanation:

Using Snowpipe for continuous, automated data ingestion minimizes the need for manual intervention and ensures that data is available in Snowflake promptly after it is generated. Leveraging Snowflake's data sharing capabilities allows for efficient and secure access to the vendor's data without the need for complex API integrations. Materialized views provide preaggregated data for fast access, which is ideal for dashboards that require high performance1234.

Reference: =

- Snowflake Documentation on Snowpipe4
- Snowflake Documentation on Secure Data Sharing2
- Best Practices for Data Ingestion with Snowflake1

Question: 153

Which command will create a schema without Fail-safe and will restrict object owners from passing on access to other users?

A. create schema EDW.ACCOUNTING WITH MANAGED ACCESS;

B. create schema EDW.ACCOUNTING WITH MANAGED ACCESS DATA_RETENTION_TIME_IN_DAYS - 7;

C. create TRANSIENT schema EDW.ACCOUNTING WITH MANAGED ACCESS
DATA_RETENTION_TIME_IN_DAYS = 1;

D. create TRANSIENT schema EDW.ACCOUNTING WITH MANAGED ACCESS
DATA_RETENTION_TIME_IN_DAYS = 7;

Answer: D

Explanation:

A transient schema in Snowflake is designed without a Fail-safe period, meaning it does not incur additional storage costs once it leaves Time Travel, and it is not protected by Fail-safe in the event of a data loss. The WITH MANAGED ACCESS option ensures that all privilege grants, including future grants on objects within the schema, are managed by the schema owner, thus restricting object owners from passing on access to other users1.

Reference: =

- Snowflake Documentation on creating schemas1
- Snowflake Documentation on configuring access control2
- Snowflake Documentation on understanding and viewing Fail-safe3

Question: 154

An Architect needs to design a data unloading strategy for Snowflake, that will be used with the COPY INTO <location> command.

Which configuration is valid?

A. Location of files: Snowflake internal location
. File formats: CSV, XML
. File encoding: UTF-8
. Encryption: 128-bit

B. Location of files: Amazon S3
. File formats: CSV, JSON
. File encoding: Latin-1 (ISO-8859)
. Encryption: 128-bit

C. Location of files: Google Cloud Storage
. File formats: Parquet
. File encoding: UTF-8
· Compression: gzip

D. Location of files: Azure ADLS
. File formats: JSON, XML, Avro, Parquet, ORC
. Compression: bzip2
. Encryption: User-supplied key

Answer: C

Explanation:

Question: 155

A company has built a data pipeline using Snowpipe to ingest files from an Amazon S3 bucket. Snowpipe is configured to load data into staging database tables. Then a task runs to load the data from the staging database tables into the reporting database tables.

The company is satisfied with the availability of the data in the reporting database tables, but the reporting tables are not pruning effectively. Currently, a size 4X-Large virtual warehouse is being used to query all of the tables in the reporting database.

What step can be taken to improve the pruning of the reporting tables?

A. Eliminate the use of Snowpipe and load the files into internal stages using PUT commands.

B. Increase the size of the virtual warehouse to a size 5X-Large.

C. Use an ORDER BY <cluster_key (s) > command to load the reporting tables.

D. Create larger files for Snowpipe to ingest and ensure the staging frequency does not exceed 1 minute.

Answer: C

Explanation:

Effective pruning in Snowflake relies on the organization of data within micro-partitions. By using an ORDER BY clause with clustering keys when loading data into the reporting tables, Snowflake can better organize the data within micro-partitions. This organization allows Snowflake to skip over irrelevant micro-partitions during a query, thus improving query performance and reducing the amount of data scanned12.

Reference: =

- Snowflake Documentation on micro-partitions and data clustering2
- Community article on recognizing unsatisfactory pruning and improving it1