Welcome to Cert007 - Your Ultimate IT Certification Partner



- Real Exam Questions
- Free Updates
- Expert Support
- Instant Access
- Money-Back Guarantee

Visit us at **https://www.cert007.com/** for more information

Exam : **ARA-C01**

Title : SnowPro Advanced Architect Certification

Version : DEMO

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: A, D
Explanation:
In Snowflake, the change tracking metadata for a table is utilized by the MERGE command and the STREAM object. The MERGE command uses change tracking to determine how to apply updates and inserts efficiently based on differences between source and target tables. STREAM objects, on the other hand, specifically capture and store change data, enabling incremental processing based on changes made to a table since the last stream offset was committed.
Reference: Snowflake Documentation on MERGE and STREAM Objects.

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

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

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

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: D
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.