Snowflake.ARA-R01.v2024-04-16.q49

Exam Code:	ARA-R01
Exam Name:	SnowPro Advanced: Architect Recertification Exam
Certification Provider:	Snowflake
Free Question Number:	49
Version:	v2024-04-16
# of views:	928
# of Questions views:	490
https://www.freepdfdumps.com/Snowflake.ARA-R01.v2024-04-16.q49.html	

NEW QUESTION: 1
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 (LEAVE A REPLY)
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. References:
SnowPro Advanced: Architect Exam Study Guide, page 11, section 2.3.1
Snowflake Documentation: Enabling the Search Optimization Service

NEW QUESTION: 2
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;
```
```
alter warehouse snowpark_opt_wh set max_concurrency_level = 16;
```
D.

Answer: A (LEAVE A REPLY)

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. References:

[Snowpark-optimized Warehouses] 1

[Training Machine Learning Models with Snowpark Python] 2

[Snowflake Shorts: Snowpark Optimized Warehouses] 3

NEW QUESTION: 3

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: (SHOW ANSWER)

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.

References:

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

NEW 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: B,C (LEAVE A REPLY)
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

NEW QUESTION: 5
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 (LEAVE A REPLY)
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.
```
createpipe mypipeas
copyintomytable
from(
select*from@mystage
wherecol1='A'andcol2>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.
```
createpipe mypipeas
copyintomytable (col1, col2, col3)
from(
selectcol3, col1, col2from@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.
```
createpipe mypipeas
copyintomytable (col1, col2)
from(
selectcol1, col2from@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.
```
createpipe mypipeas
copyintomytable (col1, col2)
from(
selectcol1::date, col2from@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.
```
createpipe mypipeas
copyintomytable (col1, col2, col3)
from(
selects.col1, s.col2, t.col3from@mystages
```

joinothertable tons.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.
createpipe mypipeas
copyintomytable
from(
select*from@mystage
onerror abort
);
References:
1: CREATE PIPE | Snowflake Documentation
2: Transforming Data During a Load | Snowflake Documentation

NEW QUESTION: 6
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 dataimports.
Answer: B,D (LEAVE A REPLY)
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.
References:
Considerations for Semi-structured Data Stored in VARIANT
Snowflake Architect Training
Snowflake query performance on unique element in variant column
Snowflake variant performance

NEW QUESTION: 7
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 (LEAVE A REPLY)
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.
References:
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]

NEW QUESTION: 8
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 (LEAVE A REPLY)
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/

NEW QUESTION: 9
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 (LEAVE A REPLY)
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 data. 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.
References:
Introduction to Replication and Failover
Replicating Databases Across Multiple Accounts
Cloning Considerations

NEW QUESTION: 10
A company has a source system that provides JSON records for various loT 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 (LEAVE A REPLY)
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. References:

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.

NEW QUESTION: 11
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 (LEAVE A REPLY)
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. References:
[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

NEW QUESTION: 12
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 INTERN;
GRANT ROLE INTERN TO USER USER1;
```


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 (LEAVE A REPLY)

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.

References: : Snowflake - Principle of Least Privilege : Snowflake - Access Control Privileges : Snowflake - Public Role : Snowflake - Ownership and Grants

NEW QUESTION: 13

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 (LEAVE A REPLY)
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.

NEW QUESTION: 14
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 (LEAVE A REPLY)
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. References: Access Control Privileges | Snowflake Documentation, Using Dynamic Data Masking | Snowflake Documentation, Using Row Access Policies | Snowflake Documentation, Using Session Policies | Snowflake Documentation

NEW QUESTION: 15
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: (SHOW ANSWER)
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. References: Snowflake Documentation: Using Snowpipe, Snowflake Documentation: Pausing and Resuming a Pipe

NEW QUESTION: 16
The following DDL command was used to create a task based on a stream:

```
CREATE TASK ts_insert_new_customers
  WAREHOUSE = MY_WH
  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: (SHOW ANSWER)
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. References:
[CREATE TASK | Snowflake Documentation]
[Using Streams and Tasks | Snowflake Documentation]
[CREATE WAREHOUSE | Snowflake Documentation]

NEW QUESTION: 17
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 (LEAVE A REPLY)

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

NEW QUESTION: 18
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: (SHOW ANSWER)
For ingesting a large volume of CSV data into Snowflake using Snowpipe, especially for a substantial amount like 10 TB, theon error = SKIP_FILEoption in theCOPY INTOcommand 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.

NEW QUESTION: 19
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 (LEAVE A REPLY)
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. References:
Snowflake Connector for Kafka
Snowpipe
Snowflake Streams
Snowflake Spark Connector

NEW QUESTION: 20
An Architect needs to design a Snowflake account and database strategy to store and analyze large amounts of structured and semi-structured data. 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 (LEAVE A REPLY)
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. References:
Snowflake Documentation:
Database Cloning, Snowflake Documentation: Database Sharing, [Snowflake Documentation:
Warehouse Scaling]

NEW 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: (SHOW ANSWER)
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.
References:
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

NEW QUESTION: 22
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 (LEAVE A REPLY)
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.
References:
Snowflake Documentation: Storage Integrations
Snowflake Documentation: External Stages

NEW QUESTION: 23
A company has a table with that has corrupted data, named Data. 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 (LEAVE A REPLY)

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

NEW QUESTION: 24
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_format =
(
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 (LEAVE A REPLY)
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.
References: : COPY INTO <table> : Transforming Data During a Load

NEW QUESTION: 25
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 (LEAVE A REPLY)
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. References:
[Introduction to Secure Data Sharing] 1
[Working with Shares] 2
[Choosing How to Share Database Objects] 3

NEW QUESTION: 26
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 (
UniqueId number,
DeviceId varchar(20),
DeviceManufacturer varchar(50)
CustomerId varchar(20),
IOT_timestamp timestamp_ntz,
City varchar(80),
Location varchar(50)
)
```


The general query patterns for the table are:
1. DeviceId, lOT_timestamp and CustomerId 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 UniqueId
Which field(s) should be used for the clustering key?

A. IOT_timestamp
B. City and DeviceManuf acturer
C. DeviceId and CustomerId
D. UniqueId
Answer: (SHOW ANSWER)
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 micro-partitions and speed up thequeries. References: Clustering Keys & Clustered Tables, Micro-partitions & Data Clustering, A Complete Guide to Snowflake Clustering

NEW QUESTION: 27
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 (LEAVE A REPLY)
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.
References:
1: Masking Policies | Snowflake Documentation
2: Using Conditional Columns in Masking Policies | Snowflake Documentation
3: Secure Views | Snowflake Documentation

NEW QUESTION: 28
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 (LEAVE A REPLY)
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
References:
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

NEW QUESTION: 29
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: (SHOW ANSWER)
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.
References:
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]

NEW QUESTION: 30
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 =>
'2021-05-01 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: (SHOW ANSWER)
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.
References: : Cloning Considerations : Understanding & Using Time Travel : CREATE <object> ... CLONE

NEW QUESTION: 31
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 (LEAVE A REPLY)
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 scanthe 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.
References: : Understanding Compute Cost : MAX Function : COUNT Function : GROUP BY Clause : SHOW TABLES

NEW QUESTION: 32
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: (SHOW ANSWER)
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. References: CREATE SECURITY INTEGRATION | Snowflake Documentation, CREATE STAGE | Snowflake Documentation, CREATE STORAGE INTEGRATION | Snowflake Documentation, CREATE API INTEGRATION | Snowflake Documentation

NEW QUESTION: 33
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 (LEAVE A REPLY)
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.
References:
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

NEW QUESTION: 34
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 creditcardno_mask as (val string) returns string ->
case
when is_role_in_session('PI_ANALYTICS') then
right(val,4)
else '***MASKED***'
end;
```


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: (SHOW ANSWER)
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.
References: 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

NEW QUESTION: 35

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: (SHOW ANSWER)
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 viewsor 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.
References:
Using the SIMULATED_DATA_SHARING_CONSUMER Session Parameter
SnowPro Advanced: Architect Exam Study Guide

NEW QUESTION: 36
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 (LEAVE A REPLY)
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.
References:
Snowflake Documentation: Stored Procedures
Snowflake Documentation: Understanding Caller's Rights and Owner's Rights Stored Procedures

NEW QUESTION: 37
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: (SHOW ANSWER)
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.
References:
Snowpipe Overview | Snowflake Documentation
Introduction to Data Pipelines | Snowflake Documentation
External Functions Overview | Snowflake Documentation
Snowflake Data Marketplace Overview | Snowflake Documentation

NEW QUESTION: 38
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 (LEAVE A REPLY)
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.
References:
Snowflake Organizations Overview | Snowflake Documentation
Design Patterns for Building Multi-Tenant Applications on Snowflake

NEW QUESTION: 39
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 (LEAVE A REPLY)
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. References: CHANGES | Snowflake Documentation, Change Tracking Using Table Streams | Snowflake Documentation, MERGE | Snowflake Documentation

NEW QUESTION: 40
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 (LEAVE A REPLY)
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. Zero-copy 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 References:
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

NEW QUESTION: 41
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 (LEAVE A REPLY)
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.
References:
CREATE DATABASE | Snowflake Documentation
Importing Data from a Share | Snowflake Documentation
Importing a Share | Snowflake Documentation
CREATE SHARE | Snowflake Documentation

NEW QUESTION: 42
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 (LEAVE A REPLY)
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.
References:
Materialized Views | Snowflake Documentation

NEW QUESTION: 43
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 (LEAVE A REPLY)
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.
References:
Snowflake Documentation: Loading Data into Tables with Default Values
Snowflake Documentation: COPY INTO table

NEW QUESTION: 44
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: (SHOW ANSWER)
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 mustbe 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.
References:
Sharing data securely across regions and cloud platforms
Introduction to replication and failover
Replication considerations
Replicating account objects

NEW QUESTION: 45
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 (LEAVE A REPLY)
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.
References: : Transient Tables : Temporary Tables : Understanding & Using Time Travel

NEW QUESTION: 46
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: (SHOW ANSWER)
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.
References:
: Stages
: LIST Function

NEW QUESTION: 47
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 (LEAVE A REPLY)
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. References:
Tasks
Snowflake Scripting
User-Defined Functions

NEW QUESTION: 48
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: (SHOW ANSWER)
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.
References:
Parameters | Snowflake Documentation
Object Parameters | Snowflake Documentation
Object Privileges | Snowflake Documentation

NEW QUESTION: 49
A user named USER_01 needs access to create a materialized view on a schema EDW. STG_SCHEMA. 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: (SHOW ANSWER)
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. References:
Snowflake Documentation: CREATE MATERIALIZED VIEW
Snowflake Documentation: Working with Materialized Views
[Snowflake Documentation: GRANT Privileges on a Schema]