DUMPSARENA

SnowPro Advanced: Data Engineer Certification Exam

Snowflake DEA-C01

Version Demo

Total Demo Questions: 13

Total Premium Questions: 130

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, Data Movement	33
Topic 2, Data Transformation	25
Topic 3, Data Storage	20
Topic 4, Data Security	26
Topic 5, Performance Optimization	26
Total	130

QUESTION NO: 1

Ryan, a Data Engineer, wants to improve the performance of large, complex queries against large

data sets. He decided to Scale up underlying warehouse/cluster. What is correct Snowflake consideration while scaling up so that he can achieve better performance results? [Select all that apply]

A. Resizing can help reduce the queuing that occurs if a warehouse does not have enough compute resources to process all the queries that are submitted concurrently.

B. Scaling up is not intended for handling concurrency issues; instead, use additional warehouses to handle the workload or use a multi-cluster warehouse (if this feature is available for your account).

C. Snowflake supports resizing a warehouse at any time, even while running.

D. Resizing a running warehouse does not impact queries that are already being processed by the warehouse; the additional compute resources, once fully provisioned, are only used for queued and new queries.

E. Resizing between a 5XL or 6XL warehouse to a 4XL or smaller warehouse results in a brief period during which the customer is charged for both the new warehouse and the old warehouse while the old warehouse is quiesced.

Explanation:

ANSWER: A B C D E

Explanation:

When scaling up (resizing) a Snowflake virtual warehouse, the key considerations are how resizing affects throughput, concurrency, and in-flight work. Increasing warehouse size can improve performance for large/complex queries by providing more compute resources, and it can also reduce queuing when the warehouse is under-provisioned for the submitted workload. However, scaling up is not the primary mechanism for handling high concurrency; Snowflake recommends adding more warehouses or using a multi-cluster warehouse to scale out concurrent query processing. Snowflake also allows resizing at any time, including while the warehouse is running. A critical operational detail is that resizing does not speed up queries already executing; the new size takes effect for queries that start after the resize (typically queued or newly submitted queries once the additional resources are provisioned). Finally, for certain downsize transitions (notably from very large sizes such as 5XL/6XL down to 4XL or smaller), Snowflake may briefly bill for both the old and new warehouse while the old one is quiesced, which is an important cost consideration during resizing operations.

References: https://docs.snowflake.com/en/user-guide/warehouses-overview, https://docs.snowflake.com/en/user-guide/warehouses-tasks

QUESTION NO: 2

Snowflake supports using key pair authentication for enhanced authentication security as an alternative to basic authentication (i.e. username and password). Select the list of SnowFlake Clients support

the same? [Select All that Apply]

A. Go Driver

B. Node.js

C. SnowFlake Connector for Spark

D. SnowSQL

E. SnowCD

ANSWER: A B C D

Explanation:

Key-pair authentication in Snowflake uses an RSA public/private key pair where the public key is registered on the Snowflake user and the client proves possession of the private key during login. Snowflake supports this method across multiple official clients and drivers. The Go driver supports key-pair authentication by providing the private key (or private key file) in the connection configuration. The Node.js driver also supports key-pair authentication via connection parameters that supply the private key material. SnowSQL supports key-pair authentication as well, allowing users to authenticate with a private key file instead of a password. Additionally, the Snowflake Connector for Spark supports key-pair authentication because it relies on the Snowflake JDBC driver, which supports key-pair authentication through standard JDBC properties (for example, providing a private key). These are all documented Snowflake-supported approaches for passwordless authentication and are commonly recommended for automation and service accounts.

References: https://docs.snowflake.com/en/user-guide/key-pair-auth, https://docs.snowflake.com/en/user-guide/snowsql-start#using-key-pair-authentication

QUESTION NO: 3

Melissa, Senior Data Engineer, looking out to optimize query performance for one of the Critical Control Dashboard, she found that most of the searches by the users on the control dashboards are based on Equality search on all the underlying columns mostly. Which Best techniques she should consider here?

A. She can go for clustering on underlying tables which can speedup Equality searches.

B. A materialized view speeds both equality searches and range searches.

C. The search optimization service would best fit here as it can be applied to all underlying columns & speeds up equality searches.
(Correct)

D. Melissa can create Indexes & Hints on the searchable columns to speed up Equality search.

Explanation:

ANSWER: C

Explanation:

The search optimization service is the best fit when the workload is dominated by highly selective equality predicates across many different columns. Enabling search optimization builds and maintains additional search access paths that allow Snowflake to prune micro-partitions more effectively for point-lookups (e.g., `WHERE col = 'X'`) without requiring you to choose and maintain a specific clustering key. This aligns with the scenario where users perform equality searches on “all the underlying columns mostly,” because search optimization can be enabled at the table level and then benefits supported column types broadly, rather than only a single clustering key or a limited projection like a materialized view. It’s specifically designed to accelerate selective point queries and can be applied surgically to the tables that back critical dashboards where latency matters. As with any performance feature, it has ongoing maintenance cost, so it’s typically reserved for critical, repetitive lookup patterns—exactly the described dashboard use case.

References: https://docs.snowflake.com/en/user-guide/search-optimization-service, https://docs.snowflake.com/en/sql-reference/sql/alter-table-search-optimization

QUESTION NO: 4

Which of the following security and governance tools/technologies are known to provide native connectivity to Snowflake? [Select 2]

A. ALTR

B. Baffle

C. BIG Squid

D. Dataiku

E. Zepl

Explanation:

ANSWER: A B D

Explanation:

ALTR and Baffle are security/governance vendors that integrate directly with Snowflake and are commonly referenced in Snowflake's security ecosystem as partners providing native-style connectivity/integration patterns for Snowflake deployments. These types of tools typically connect to Snowflake using supported interfaces (for example, Snowflake connectors/drivers) and then apply governance controls such as data access policies, masking/tokenization, monitoring, and auditing aligned to Snowflake's security model. In practice, organizations use these integrations to enforce fine-grained protection of sensitive data while keeping data in Snowflake, supporting compliance requirements and centralized governance workflows. Snowflake's partner ecosystem highlights security and governance solutions that work with Snowflake, and both ALTR and Baffle are positioned in that category for Snowflake customers looking to extend governance and data protection capabilities. For additional context on Snowflake's security ecosystem and partner integrations, see Snowflake's partner listings and security overview documentation: https://www.snowflake.com/partners/ and https://docs.snowflake.com/en/user-guide/security.

QUESTION NO: 5

External Function is a type of UDF & can be Scaler or Tabular?

A. TRUE

B. FALSE

Explanation:

ANSWER: B

Explanation:

"FALSE" is correct because Snowflake external functions are restricted to scalar behavior: they return exactly one value per input row. While Snowflake supports both scalar and tabular user-defined functions in general (for example, SQL UDFs can be scalar and table functions exist for returning result sets), external functions specifically are designed to call out to an external service (such as an API hosted behind an API Gateway) and map each input row to a single returned value. This scalar-only requirement is part of the external function contract and is important for predictable execution, row-by-row evaluation, and integration with Snowflake's query processing model. In practice, if you need to return multiple rows/columns from an external service, you typically redesign the pattern (e.g., return a VARIANT/OBJECT/ARRAY payload and then parse/flatten it in Snowflake) rather than expecting the external function itself to be tabular.

References: https://docs.snowflake.com/en/sql-reference/external-functions, https://docs.snowflake.com/en/developer-guide/udf/udf-overview

QUESTION NO: 6

Clones can be cloned, with no limitations on the number or iterations of clones that can be created (e.g. you can create a clone of a clone of a clone, and so on), which results in a n-level hierarchy of cloned objects, each with their own portion of shared and independent data storage?

A. TRUE

B. FALSE

ANSWER: A

Explanation:

TRUE is correct. In Snowflake, a clone is a metadata-only copy created using zero-copy cloning. Because the clone initially references the same underlying micro-partitions as the source object, it is fast and does not duplicate storage at creation time. Snowflake also allows cloning from an existing clone, and you can repeat this process without a fixed “depth” limit, effectively forming an n-level lineage of cloned objects. As changes are made to any object in that lineage, Snowflake’s copy-on-write behavior causes only the modified micro-partitions to diverge, so each clone ends up with a mix of shared storage (unchanged partitions still referenced from the original lineage) and independent storage (new/changed partitions owned by that specific object). This is a key design point of zero-copy cloning: multiple objects can share the same physical data until modifications require new storage. This behavior applies to supported objects such as tables, schemas, and databases, and is foundational to workflows like branching datasets for development, testing, and recovery.

References: https://docs.snowflake.com/en/user-guide/object-clone, https://docs.snowflake.com/en/user-guide/data-time-travel

QUESTION NO: 7

In one of your created Schema, you have been required to create Internal Stages, what are the Incorrect considerations you can noticed from the below options? [Select All that Apply]

A. User stages can be altered or dropped just like Table Stage.

B. Table stage type is designed to store files that are staged and managed by one or more users but only loaded into a single table.

C. A named internal stage type can store files that are staged and managed by one or more users and loaded into one or more tables.

D. A table stage is available for each table created in Snowflake.

Explanation:

ANSWER: A

Explanation:

The statement “User stages can be altered or dropped just like Table Stage.” is the incorrect consideration. In Snowflake, user stages and table stages are implicit internal stages that Snowflake automatically provides: each user gets a user stage (referenced as `@~`) and each table gets a table stage (referenced as `@%<table_name>`). Because these stages are implicit and not standalone database objects, they cannot be altered or dropped. Only named stages (created with `CREATE STAGE`) are first-class objects that support DDL operations such as `ALTER STAGE` and `DROP STAGE`, and can have privileges granted/revoked and ownership transferred. This distinction is important when designing repeatable loading patterns: if you need configurable stage properties (e.g., directory tables, comments, storage integration for external stages, etc.) or reusable access control, you typically use a named stage; if you just need a convenient per-user or per-table landing area, you use the implicit stages as-is.

References: Snowflake Docs: Create an Internal Stage, Snowflake Docs: Staging Data Files

QUESTION NO: 8

Snowflake computes and adds partitions based on the defined partition column expressions when an external table metadata is refreshed.

What are the Correct Statements to configure Partition metadata refresh in case of External Tables?

A. By default, the metadata is refreshed automatically when the object is created.

B. The object owner can configure the metadata to refresh automatically when new or updated data files are available in the external stage.

C. Metadata refresh is not required as its Managed implicitly by Snowflake.

D. Partitions of External tables is managed by External Stage Cloud provider.

E. There is nothing like adding partitions on External tables.

Explanation:

ANSWER: A B

Explanation:

For Snowflake external tables, partition information is maintained in the external table metadata and is populated/updated when the metadata is refreshed. When you create an external table, Snowflake performs an initial metadata refresh by default, which discovers the files in the referenced external stage and computes partition values based on the partition column expressions (typically parsing `METADATA$FILENAME`). After creation, you can keep this metadata (and therefore partitions) current either by manually triggering a refresh using `ALTER EXTERNAL TABLE … REFRESH` or by configuring automated refresh. Automated refresh is accomplished by integrating Snowflake with your cloud storage event notification mechanism (e.g., S3 event notifications, Azure Event Grid, or GCS Pub/Sub) so Snowflake is notified when new or changed objects arrive and can refresh the external table metadata accordingly. This is the supported way to ensure partitions are added as new files land, and it is controlled by the object owner (or a role with appropriate privileges) as part of the external table/stage notification configuration.

References: https://docs.snowflake.com/en/user-guide/tables-external-intro, https://docs.snowflake.com/en/sql-reference/sql/alter-external-table

QUESTION NO: 9

Steven created the task, what additional privileges required by Steven on the task so that he can suspend or resume the tasks?

A. Steven is already owner of the task; he can execute the task & suspend/resume the task without any additional privileges.

B. In addition to the task owner, a Steven Role must have OPERATE privilege on the task so that he can suspend or resume the task.

C. Steven must have SUSPEND privilege on the task so that he can suspend or resume the task.

D. Steven needs to have Global Managed RESUME privilege by TASK administrator.
Explanation:

ANSWER: B

Explanation:

To suspend or resume a Snowflake task, the controlling role must have the ability to operate the task. Snowflake provides this through the `OPERATE` privilege on the task, which explicitly authorizes operational actions such as `ALTER TASK ... SUSPEND` and `ALTER TASK ... RESUME`. While task ownership confers broad control, Snowflake's privilege model commonly distinguishes between owning an object and delegating day-to-day operational control to other roles; granting `OPERATE` is the standard mechanism to allow suspend/resume without transferring ownership. Therefore, the correct requirement described is that a role needs the `OPERATE` privilege on the task to suspend or resume it. This aligns with Snowflake task privilege guidance and the general object privilege framework used across Snowflake for operational actions.

References: https://docs.snowflake.com/en/user-guide/tasks-intro, https://docs.snowflake.com/en/sql-reference/sql/grant-privilege

QUESTION NO: 10

In efforts to recover the dropped child tables within schema named SCV_SCHEMA by Data Engi-neer, She found that DATA_RETENTION_TIME_IN_DAYS parameter set with value 45 days at Schema level & the data retention period for child tables explicitly set at 85 days. What will happen when she will

try to run undrop table command on Child tables to recover them on the 50th day as-suming SCV_SCHEMA is already dropped on 45th day?

A. To honor the data retention period for child tables, She will ab able to recover the child tables on 50th day as DATA_RETENTION_TIME_IN_DAYS is explicitly set with higher retention value.

B. When a schema is already dropped, the data retention period for child tables, if explicit-ly set to be different from the retention of the schema, is not honoured. So UNDROP command will fail to run on 50th day for Child tables recovery.

C. Child tables can be recovered using Fail-Safe SQL commands.

D. Data Engineer needs to first recover the Schema & then Child tables will automatically be recovered irrespective of Retention Inheritance.
Explanation:

ANSWER: B

Explanation:

When a schema is dropped, Snowflake currently does not honor a longer, explicitly configured data retention period on child tables if it differs from the dropped schema's retention. Instead, the dropped child tables are retained only for the same retention period as the dropped schema container. In this scenario, the schema-level DATA_RETENTION_TIME_IN_DAYS is 45 days, and the schema was dropped on day 45. That means the child tables' ability to be recovered via Time Travel/UNDROP is bounded by the schema's retention window, not the tables' explicitly set 85 days. By day 50, the schema's retention window has already elapsed, so attempting to UNDROP the child tables will not succeed because the objects are no longer available for undrop within Time Travel. To preserve a longer retention for child objects, Snowflake's documented best practice is to explicitly drop the child objects before dropping the parent container so their individual retention settings can be applied independently.

References: https://docs.snowflake.com/en/user-guide/data-time-travel, https://docs.snowflake.com/en/sql-reference/sql/undrop-table

QUESTION NO: 11

Which of the below concepts/functions helps while implementing advanced Column-level Security?

A. CURRENT_ROLE

B. INVOKER_ROLE

C. Role Hierarchy

D. CURRENT_CLIENT

Explanation:

ANSWER: A B C

Explanation:

Advanced column-level security in Snowflake is commonly implemented with masking policies, where the policy body can use context functions and role evaluation to decide whether to reveal or mask values at query time. Using `CURRENT_ROLE` helps because masking policy logic often needs to check the active role in the user's current session to determine whether the user should see cleartext data. Using `INVOKER_ROLE` helps because some access patterns (for example, when objects are accessed through views or other executed SQL contexts) benefit from evaluating the role that is effectively executing the statement, enabling more precise authorization logic inside the masking policy. Role hierarchy is also important because

Snowflake roles inherit privileges through granted roles; masking policy conditions frequently need to account for inherited access (for example, allowing a higher-level role that contains a lower-level “allowed” role). Together, these concepts enable robust, least-privilege masking decisions that align with how Snowflake authorization is actually evaluated at runtime.

References: Snowflake Docs: Dynamic Data Masking (Masking Policies), Snowflake Docs: Access Control Overview (Role hierarchy)

QUESTION NO: 12

Data Engineer looking out for quick tool for understanding the mechanics of queries & need to know more about the performance or behaviour of a particular query.

He should go to which feature of snowflake which can help him to spot typical mistakes in SQL query expressions to identify potential performance bottlenecks and improvement opportunities?

A. Query Optimizer

B. Performance Metadata table

C. Query Profile

D. Query Designer **Explanation:**

ANSWER: C

Explanation:

The correct feature is the Query Profile. Snowflake’s Query Profile is specifically designed to help engineers understand how a query executed and where time and resources were spent. It provides a visual execution plan with operator-level statistics (for example, scan, join, aggregation, and data movement), along with overall query metrics. This makes it the fastest built-in way to investigate the mechanics and behavior of a single query and to identify common performance issues such as inefficient joins, excessive spilling, skew, or large scans caused by non-selective filters. In practice, you open a completed (or running) query in the Snowflake UI and view the Query Profile to see the execution graph and detailed operator information, which directly supports spotting bottlenecks and improvement opportunities. This aligns with Snowflake guidance that Query Profile is the primary tool for query troubleshooting and performance analysis at the individual query level.

References: https://docs.snowflake.com/en/user-guide/ui-query-profile, https://docs.snowflake.com/en/sql-reference/sql/explain

QUESTION NO: 13

To advance the offset of a stream to the current table version without consuming the change data in a DML operation, which of the following operations can be done by Data Engineer? [Select 2]

A. using the CREATE OR REPLACE STREAM syntax, Recreate the STREAM

B. Insert the current change data into a temporary table. In the INSERT statement, query the stream but include a WHERE clause that filters out all of the change data (e.g. WHERE 0 = 1).

C. A stream advances the offset only when it is used in a DML transaction, so none of the options works without consuming the change data of table.

D. Delete the offset using STREAM properties SYSTEM$RESET_OFFSET( ) **Explanation:**

ANSWER: A B

Explanation:

Snowflake streams track change data by maintaining an internal offset that points to a specific transactional version of the source object. Simply querying a stream does not move that offset; the offset only advances when the stream is referenced

by a DML statement that “consumes” the stream. If you need to move the stream offset forward to the current table version without actually processing the accumulated change rows, Snowflake supports two practical approaches. One is to recreate the stream using CREATE OR REPLACE STREAM, which reinitializes the stream’s offset to the current version of the source object (effectively discarding any unconsumed change records). The other is to execute a DML statement that references the stream but filters out all rows (for example, inserting into a temporary table with a predicate like WHERE 0=1). Because the stream is used in a DML statement, Snowflake advances the offset, but no change rows are actually written anywhere due to the always-false filter. These behaviors are documented as standard stream consumption and offset management patterns in Snowflake.

References: https://docs.snowflake.com/en/user-guide/streams, https://docs.snowflake.com/en/sql-reference/sql/create-stream