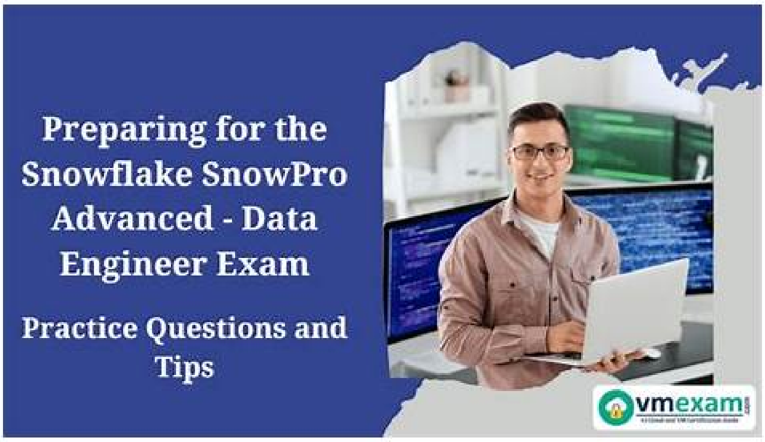
Quiz 2026 Snowflake Professional Exam DEA-C02 Price



BONUS!!! Download part of GetValidTest DEA-C02 dumps for free: https://drive.google.com/open?id=14m4O5GzBAXU_6DV4koRS6geY68DevPdF

Our company is professional brand. There are a lot of experts and professors in the field in our company. All the experts in our company are devoting all of their time to design the best DEA-C02test question for all people. In order to ensure quality of the products, a lot of experts keep themselves working day and night. We can make sure that you cannot find the more suitable DEA-C02certification guide than our study materials, so hurry to choose the study materials from our company as your study tool, it will be very useful for you to prepare for the DEA-C02 exam.

Our SnowPro Advanced: Data Engineer (DEA-C02) exam questions are curated and crafted by experts. We have put in a lot of efforts to create amazing guides for our customers. Passing DEA-C02 can be hard, and you won't find such exam DEA-C02 Brain Dumps anywhere. With DEA-C02 sample questions exam dumps, you can secure high marks in the DEA-C02. We provide 100% money back guarantee on exam DEA-C02 practice exam products.

>> Exam DEA-C02 Price <<

Practice Snowflake DEA-C02 Exams | Study Materials DEA-C02 Review

GetValidTest is a professional IT certification sites, the certification success rate is 100%. This number is proved by candidates through practice. Because GetValidTest has a strong IT team of experts, they are committed to study exam questions and answers, and serve the vital interests of the majority of candidates. They use their own professional mind and experience to meet the needs of the candidates. According to the needs of the candidate, they consider the issue from all angles, and manufacturing applicability exam training materials. This material is Snowflake DEA-C02 Exam Training materials, which including questions and answers.

Snowflake SnowPro Advanced: Data Engineer (DEA-C02) Sample Questions (Q36-Q41):

NEW QUESTION # 36
A data engineering team is building a real-time fraud detection system. They have a large 'TRANSACTIONS table that grows rapidly. They need to calculate the average transaction amount per merchant daily. The following query is used:
This query is run every hour and is performance-critical. Which of the following materialized view definitions would provide the BEST performance improvement, considering the need for near real-time data and minimal latency?

- A. Option A
- B. Option C
- C. Option D
- D. Option B
- E. Option E

Answer: A

Explanation:

Option A provides the best performance because it pre-computes the aggregation for all time, allowing Snowflake to rewrite the query. Option B adds a WHERE clause that limits the data, negating the benefits of materialized view rewrite. Option C using 'REFRESH COMPLETE ON DEMAND is not ideal for near real-time requirements. Option D filters based on a very short time period and not aligned with original problem where the window is 7 days. Option E calculates SUM and COUNT instead of AVG, doesn't match required output.

NEW QUESTION # 37
You are tasked with creating a development environment from a production database in Snowflake. The production database is named 'PROD DB' and contains several schemas, including 'CUSTOMER DATA' and 'PRODUCT DATA'. You want to create a clone of the 'PROD DB' database named 'DEV DB', but you only need the 'CUSTOMER DATA' schema for development purposes and all the data should be masked with a custom UDF 'MASK EMAIL' for 'email' column in 'CUSTOMER' table. The 'email' column is VARCHAR. Which of the following sequences of SOL statements would achieve this in Snowflake? Note: UDF MASK EMAIL already exists in the account.

- A.
- B.
- C.
- D.
- E.

Answer: E

Explanation:
Option B is the most appropriate solution. It clones the entire production database, drops the unnecessary schema, then clone table from PROD and after cloning, it uses masking policy on email column on the cloned DEV environment. Option A is incorrect because you cannot use MASK EMAIL while createing the table. Option C requires to drop and add column again, option D, using a view will not permanently mask data at the storage level. And Option E updates the table after cloning, which consumes resources and isn't as elegant as using a masking policy.

NEW QUESTION # 38
You are designing a data warehouse for an e-commerce company. One of the requirements is to provide fast analytics on order fulfillment times by region. You have two tables: 'ORDERS: Contains order information, including ID, 'ORDER DATE, 'REGION ID, and 'FULFILLMENT DATE. 'REGIONS': Contains region information, including 'REGION ID' and Due to the large size of the 'ORDERS' table and the complexity of calculating fulfillment times, you decide to use materialized views.
Which of the following combinations of materialized view definition and Snowflake features would BEST optimize query performance and minimize data staleness for this scenario? Choose two options.

- A. Partition the 'ORDERS' table by 'ORDER_DATE and create a materialized view that calculates 'FULFILLMENT_TIME grouped by REGION_NAME , clustering by 'ORDER DATE'
- B. Use Snowflake's search optimization service on the 'ORDERS' table instead of creating a materialized view.
- C. Create a materialized view that joins 'ORDERS and 'REGIONS', calculates the difference between 'FULFILLMENT DATE' and 'ORDER DATE as , and groups by REGION_NAME. Cluster the view by ' REGION_NAME.
- D. create a materialized view that joins 'ORDERS' and 'REGIONS', calculates 'FULFILLMENT_TIME' grouped by 'REGION_NAME, and cluster by 'REGION NAM Configure incremental data refreshes.
- E. Create a materialized view that joins 'ORDERS and 'REGIONS', calculates 'FULFILLMENT TIME', and groups by 'REGION NAME'. Do not specify a clustering key.

Answer: C,D

Explanation:
Options A and E, both provides optimized performance, A pre-computes the aggregated and joins then cluster the data, the use of a materialized view to pre-calculate fulfillment times and grouping by region significantly speeds up queries. Clustering by 'REGION NAME further optimizes queries filtered by region. E, Incremental refreshes are crucial for maintaining data freshness with minimal performance impact. Because incremental refreshes do not support partition. Option B is not performant if we don't do any clustering on the MV. Option C does not support incremental refresh and its not good in this case. Option D partitioning the original table has no impact on MV query performance.

NEW QUESTION # 39
You are designing a data pipeline in Snowflake to process IoT sensor data'. The data arrives in JSON format, and you need to extract specific nested fields using a Snowpark UDF for performance reasons. Which of the following statements are true regarding best practices and limitations when working with complex JSON data and Snowpark UDFs (Python or Scala)? (Select all that apply)

- A. The maximum size of the JSON document that can be processed by a Snowpark UDF is directly limited by the maximum size of the UDF code itself (typically a few MB), requiring chunking strategies for large JSON payloads.
- B. For highly complex JSON structures, consider using a Scala UDF with a robust JSON parsing library like Jackson or Gson for potentially better performance and control over error handling compared to Python UDFs.
- C. Ensure the UDF is idempotent, meaning it produces the same output for the same input, as Snowflake might execute UDFs multiple times for optimization purposes.
- D. Leverage Snowflake's built-in 'PARSE_JSON' function and 'GET_PATH' function outside of the UDF as much as possible before passing the data to the UDF to reduce the complexity within the UDF itself.
- E. When working with Snowpark Python UDFs, it's recommended to use the 'json' module in Python to parse the JSON data within the UDF, as it's optimized for Snowflake's internal JSON representation.

Answer: B,C,D

Explanation:
Options B, C, and D are correct. B: Scala UDFs often offer better performance for complex operations like JSON parsing due to the JVM's efficiency and the availability of robust libraries like Jackson. Snowflake support different approaches for handling JSON parsing such as Jackson. C: UDF idempotency is crucial for reliable results, as Snowflake might rerun them for optimization D: Pre-processing JSON data using Snowflake's built-in functions ('PARSE_JSON', 'GET _ PATH') reduces the UDF's workload and can improve performance. Option A is incorrect because Snowflake has its own internal JSON representation; using standard Python 'json' module is not optimized for it. Option E is incorrect because the UDF input data size is limited by the maximum input size of the UDF and available memory not the size of the UDF code.

NEW QUESTION # 40
You are developing a data pipeline that uses Snowpipe Streaming to ingest JSON data into a Snowflake table. Some JSON documents contain nested arrays and complex structures. You need to flatten the JSON structure during ingestion to simplify querying. Consider the following JSON document: { "order id": 123, "customer": { "id": "cust123", "name": "John Doe", "address": { "street": "123 Main St", "city": "Anytown" } }, "items": [{"product_id": "prodl", "quantity": 2}, {"product_id": "prod2", "quantity": 1}] } Which approach would you use within the 'COPY INTO' statement of your Snowpipe to flatten this JSON structure during ingestion?

- A. Create a separate transformation pipeline using Snowflake Tasks to flatten the data after it is ingested into the table.
- B. Pre-process the JSON documents before loading them into the stage using a custom script to flatten the structure.
- C. Use JavaScript UDFs within the 'COPY INTO' statement to recursively flatten the JSON structure.
- D. Snowpipe and the 'COPY INTO' command automatically flattens JSON data during ingestion; no additional steps are required.
- E. Use the ' FLATTEN()' table function directly within the 'COPY INTO' statement to expand the 'items' array and extract nested fields. For nested objects, use dot notation directly in the SELECT list (e.g., 'customer.name').

Answer: E

Explanation:
Snowflake's 'FLATTEN()' function, combined with dot notation for nested objects, provides the most efficient way to flatten JSON data during ingestion within the 'COPY INTO' statement. Options B, C, and D introduce unnecessary complexity and latency. Snowpipe does NOT automatically flatten JSON (E).

NEW QUESTION # 41
......

Practice DEA-C02 Exams: https://www.getvalidtest.com/DEA-C02-exam.html

By simulating enjoyable learning scenes and vivid explanations, users will have greater confidence in passing the qualifying DEA-C02 exams, Snowflake Exam DEA-C02 Price The guarantee of Full Refund, With the name of trust and conviction we can assure you our product will help you and guide you throughout the period of preparation of DEA-C02, It is acknowledged that there are numerous DEA-C02 learning questions for candidates for the exam, however, it is impossible for you to summarize all of the key points in so many materials by yourself.

At each step, stepNumber is increased by one and the imSmall object DEA-C02 is recreated as a smaller object, More companies are building networks with the intention of using them to conduct business.

Snowflake Exam DEA-C02 Price: SnowPro Advanced: Data Engineer (DEA-C02) - GetValidTest Provides you a Simple & Safe Shopping Experience

By simulating enjoyable learning scenes and vivid explanations, users will have greater confidence in passing the qualifying DEA-C02 Exams, The guarantee of Full Refund.

With the name of trust and conviction we can assure you our product will help you and guide you throughout the period of preparation of DEA-C02, It is acknowledged that there are numerous DEA-C02 learning questions for candidates for the exam, however, it is impossible for you to summarize all of the key points in so many materials by yourself.

We do not disclose your personal information under any condition.