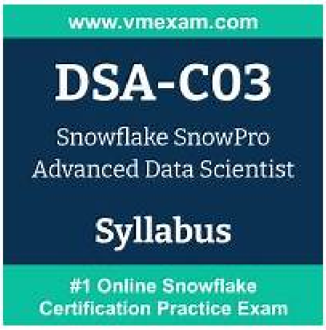
Pass Guaranteed Quiz Updated Snowflake - DSA-C03 Pdf Exam Dump



What's more, part of that Prep4sures DSA-C03 dumps now are free: https://drive.google.com/open?id=1P0FnaECY180uuullh_v3EjDmrHhr21zu

The pressure we face comes from all aspects. As the social situation changes, these pressures will only increase. We cannot change the external environment. What we can do is improve our own strength. However, blindly taking measures may have the opposite effect. So here comes your best assistant-our DSA-C03 Practice Engine. If you study with our DSA-C03 exam materials, you can become better no only because that you can learn more, but also because you can get the admired DSA-C03 certification.

Users are buying something online (such as DSA-C03 learning materials), always want vendors to provide a fast and convenient sourcing channel to better ensure the user's use. Because without a quick purchase process, users of our DSA-C03 learning materials will not be able to quickly start their own review program. So, our company employs many experts to design a fast sourcing channel for our DSA-C03 Learning Materials. All users can implement fast purchase and use our learning materials.

>> DSA-C03 Pdf Exam Dump <<

DSA-C03 free certkingdom demo & DSA-C03 latest pdf dumps

DSA-C03 Guide Quiz helped over 98 percent of exam candidates get the certificate. Before you really attend the Snowflake DSA-C03 exam and choose your materials, we want to remind you of the importance of holding a certificate like this one. Obtaining a Snowflake DSA-C03 certificate likes this one can help you master a lot of agreeable outcomes in the future, like higher salary, the opportunities to promotion and being trusted by the superiors and colleagues.

Snowflake SnowPro Advanced: Data Scientist Certification Exam Sample Questions (Q134-Q139):

NEW QUESTION # 134
You have implemented a Python UDTF in Snowflake to train a machine learning model incrementally using incoming data'. The UDTF performs well initially, but as the volume of data processed increases significantly, you observe a noticeable degradation in performance and an increase in query execution time. You suspect that the bottleneck is related to the way the model is being

updated and persisted within the UDTF. Which of the following optimization strategies, or combination of strategies, would be MOST effective in addressing this performance issue?

- A. Persist the trained model to a Snowflake stage after each batch update. Use a separate UDF (User-Defined Function) to load the model from the stage before processing new data. This decouples model training from inference.
- B. Rewrite the UDTF in Java or Scala, as these languages generally offer better performance compared to Python for computationally intensive tasks. Use the same machine learning libraries that you used with Python.
- C. Leverage Snowflake's external functions and a cloud-based ML platform (e.g., SageMaker, Vertex A1) to offload the model training process. The UDTF would then only be responsible for data preparation and calling the external function.
- D. Instead of updating the model incrementally within the UDTF for each row, batch the incoming data into larger chunks and perform model updates only on these batches. Use Snowflake's VARIANT data type to store these batches temporarily.
- E. Use the 'cachetools' library within the UDTF to cache intermediate results and reduce redundant calculations during each function call. Configure the cache with a maximum size and eviction policy appropriate for the data volume.

Answer: A,C,D

Explanation:
Options B, C, and D offer the most effective strategies for optimizing performance when training a model incrementally with a Python UDTF in Snowflake. Batching updates (B) reduces the overhead of model updates. Persisting the model to a Snowflake stage (C) decouples training from inference and allows for model reuse. Offloading training to an external function (D) leverages dedicated ML infrastructure. Caching (A) might offer some marginal improvement but is unlikely to address the core performance bottleneck. While Java or Scala (E) can be faster than Python, rewriting the UDTF is a significant undertaking and might not be necessary if other optimization strategies are applied effectively. Also the question is specific about Python. In summary, consider batching and persistence as key in performance optimization.

NEW QUESTION # 135
You have built an external function to train a PyTorch model using SageMaker. The model training process requires a significant amount of CPU and memory. The training data is passed from Snowflake to the external function in batches. The external function code in AWS Lambda is as follows:

```
import json import boto3 def lambda_handler(event, context):    try:     payload = json.loads(event['body'])
data = payload['data'] # Data received from Snowflake     # ... (PyTorch model training code using 'data' and invoking
SageMaker) ...     return {       'statusCode': 200,      'body': json.dumps({'status': 'Training complete'})     }
except Exception as e:     return {       'statusCode': 500,       'body': json.dumps({'error': str(e)})     }
```

The Snowflake external function is defined as follows:

```
CREATE OR REPLACE EXTERNAL FUNCTION TRAIN_MODEL(data variant) RETURNS VARIANT  API_INTEGRATION = your_api_integration
AS 'https://your_api_gateway_endpoint';
```

During testing, you encounter '500 Internal Server Error' from the external function consistently. Upon inspection of the Lambda logs, you find messages indicating 'PayloadTooLargeError'. What is the most likely cause and how do you mitigate it within the context of Snowflake and AWS Lambda?

- A. The size of the data being sent from Snowflake to the Lambda function exceeds the maximum payload size allowed by AWS API Gateway. Implement data partitioning in Snowflake and send smaller batches of data to the Lambda function, aggregating the results in a separate table.
- B. The Lambda function is timing out before the model training can complete. Increase the Lambda function's timeout setting to allow sufficient time for the training process.
- C. The size of the data being sent from Snowflake to the Lambda function exceeds the maximum payload size allowed by AWSAPI Gateway. Increase the maximum payload size limit in the API Gateway settings.
- D. The Snowflake external function definition is incorrect. Change the 'RETURNS VARIANT clause to 'RETURNS VARCHAR as the Lambda function returns a JSON string.
- E. The IAM role associated with the Lambda function lacks the necessary permissions to invoke the SageMaker training job. Grant the Lambda function's IAM role the appropriate SageMaker permissions.

Answer: A

Explanation:
The 'PayloadTooLargeError' indicates that the data being passed from Snowflake exceeds API Gateway's payload limit. Option D provides the correct solution: partitioning the data in Snowflake and sending smaller batches, which keeps individual payloads within the limits, and the training can still complete with all data through multiple calls. Option A is generally not recommended because increasing the payload size limit indefinitely can have negative impacts on API Gateway performance. Other reasons, such as Lambda timeout or SageMaker permissions could contribute, but the immediate cause is the payload size. Option C does not resolve this issue. Returns VARIANT is correct

NEW QUESTION # 136
A data science team is developing a churn prediction model using Snowpark Python. They have a feature engineering pipeline defined as a series of User Defined Functions (UDFs) that transform raw customer data stored in a Snowflake table named 'CUSTOMER DATA'. Due to the volume of data (billions of rows), they need to optimize UDF execution for performance. Which of the following strategies, when applied individually or in combination, will MOST effectively improve the performance of these UDFs within Snowpark?

- A. Leveraging external functions that call an API endpoint hosted on a cloud provider to perform data transformation. The API endpoint should utilize a serverless architecture.
- B. Using temporary tables to store intermediate results calculated by the UDFs instead of directly writing to the target table.
- C. Converting Python UDFs to Java UDFs, compiling the Java code, and deploying as a JAR file in Snowflake. Using a larger warehouse size is always the best first option.
- D. Utilizing vectorized UDFs with NumPy data types wherever possible and carefully tuning batch sizes. Ensure that the input data is already sorted before passing to the UDF.
- E. Repartitioning the DataFrame by a key that distributes data evenly across nodes before applying the UDFs, using the method and minimizing data shuffling.

Answer: D,E

Explanation:
Vectorized UDFs (B) are optimized for performance by processing data in batches, significantly reducing the overhead associated with individual row processing. Repartitioning (E) ensures data is evenly distributed across nodes, allowing for parallel execution of UDFs and reducing skew, which can lead to performance bottlenecks. Java UDFs while faster than unoptimized Python UDFs, require extra work and maintenance, while vectorized UDFs are a more straight forward solution within Snowpark Python. Using temporary tables (D) could add overhead rather than reducing it. Using External functions (C) is a complex solution for what can be handled natively.

NEW QUESTION # 137
You are responsible for deploying a fraud detection model in Snowflake. The model needs to be validated rigorously before being put into production. Which of the following actions represent the MOST comprehensive approach to model validation within the Snowflake environment, focusing on both statistical performance and operational readiness, and using Snowflake features for validation?

- A. Calculating only the AUC (Area Under the Curve) metric on the entire dataset without performing any data splitting or cross-validation. Deploying the model if the AUC is above 0.7.
- B. Conducting a comprehensive backtesting analysis using historical data, simulating real-world scenarios, and evaluating the model's performance under different conditions. Using Snowflake's time travel feature to access historical data snapshots for accurate backtesting. Monitoring model performance using Snowflake alerts triggered by custom SQL queries against model prediction logs.
- C. Relying on a simple visual inspection of model outputs and comparing them to a small sample of known fraud cases. Skipping formal validation to accelerate the deployment process.
- D. Performing a single train/test split of the historical data and evaluating model performance metrics (e.g., accuracy, precision, recall) on the test set using standard Python libraries within a Snowflake Snowpark environment. Deploying the model directly if the metrics exceed a predefined threshold.
- E. Implementing K-fold cross-validation using Snowflake stored procedures and temporary tables to store and aggregate the results from each fold. Evaluating the model's performance across different data segments and time periods to assess its robustness. Using Snowflake streams and tasks to automate the validation process on new incoming data.

Answer: B,E

Explanation:
Options B and C represent the most comprehensive approaches. Option B utilizes K-fold cross-validation within Snowflake for robust performance evaluation across data segments and automates validation on new data using streams and tasks. Option C emphasizes backtesting with historical data using Snowflake's time travel feature and monitors performance with alerts, ensuring real-world relevance and timely detection of performance degradation. Option A is insufficient as it relies on a single train/test split. Option D is inadequate and risky due to lack of validation. Option E is also insufficient since calculating only AUC on the entire dataset results in overfitting.

NEW QUESTION # 138

You have a table in Snowflake named 'CUSTOMER DATA' with columns 'CUSTOMER D', 'PURCHASE AMOUNT', and 'RECENCY'. You want to perform feature scaling on 'PURCHASE AMOUNT' using Min-Max scaling and store the scaled values in a new column named 'SCALED PURCHASE _ AMOUNT'. Which of the following Snowflake SQL code snippets correctly implements this feature scaling? Note: Assume there are no NULL values in PURCHASE AMOUNT and you have privileges to create temporary tables and UDFs if necessary.

```
○ CREATE OR REPLACE TEMPORARY TABLE MIN_MAX_VALUES AS SELECT MIN(PURCHASE_AMOUNT) AS MIN_VAL, MAX(PURCHASE_AMOUNT) AS MAX_VAL FROM CUSTOMER_DATA;
ALTER TABLE CUSTOMER_DATA ADD COLUMN SCALED_PURCHASE_AMOUNT FLOAT; UPDATE CUSTOMER_DATA SET SCALED_PURCHASE_AMOUNT = (PURCHASE_AMOUNT - (SELECT
MIN_VAL FROM MIN_MAX_VALUES)) / ((SELECT MAX_VAL FROM MIN_MAX_VALUES) - (SELECT MIN_VAL FROM MIN_MAX_VALUES));

○ ALTER TABLE CUSTOMER_DATA ADD COLUMN SCALED_PURCHASE_AMOUNT FLOAT; UPDATE CUSTOMER_DATA SET SCALED_PURCHASE_AMOUNT = (PURCHASE_AMOUNT -
MIN(PURCHASE_AMOUNT)) / (MAX(PURCHASE_AMOUNT) - MIN(PURCHASE_AMOUNT));

○ CREATE OR REPLACE TEMPORARY TABLE MIN_MAX_VALUES AS SELECT MIN(PURCHASE_AMOUNT) AS MIN_VAL, MAX(PURCHASE_AMOUNT) AS MAX_VAL FROM CUSTOMER_DATA;
ALTER TABLE CUSTOMER_DATA ADD COLUMN SCALED_PURCHASE_AMOUNT FLOAT; UPDATE CUSTOMER_DATA SET SCALED_PURCHASE_AMOUNT = (PURCHASE_AMOUNT - MIN_VAL) /
(MAX_VAL - MIN_VAL) FROM MIN_MAX_VALUES;

○ ALTER TABLE CUSTOMER_DATA ADD COLUMN SCALED_PURCHASE_AMOUNT FLOAT; CREATE OR REPLACE TEMPORARY TABLE MIN_MAX_VALUES AS SELECT
MIN(PURCHASE_AMOUNT) AS MIN_VAL, MAX(PURCHASE_AMOUNT) AS MAX_VAL FROM CUSTOMER_DATA; UPDATE CUSTOMER_DATA SET SCALED_PURCHASE_AMOUNT =
(CUSTOMER_DATA.PURCHASE_AMOUNT - MIN_MAX_VALUES.MIN_VAL) / (MIN_MAX_VALUES.MAX_VAL - MIN_MAX_VALUES.MIN_VAL) FROM MIN_MAX_VALUES;

○ CREATE OR REPLACE TEMPORARY TABLE MIN_MAX_VALUES AS SELECT MIN(PURCHASE_AMOUNT) AS MIN_VAL, MAX(PURCHASE_AMOUNT) AS MAX_VAL FROM CUSTOMER_DATA;
ALTER TABLE CUSTOMER_DATA ADD COLUMN SCALED_PURCHASE_AMOUNT FLOAT; UPDATE CUSTOMER_DATA SET SCALED_PURCHASE_AMOUNT = (PURCHASE_AMOUNT - MIN_VAL) /
(MAX_VAL - MIN_VAL);
```


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

Answer: B

Explanation:
Option D is correct because it calculates the min and max purchase amounts from the CUSTOMER_DATA table and stores them in a temporary table It then adds a new column to the CUSTOMER_DATA table and updates this column with the Min-Max scaled values, properly joining the temporary table to access the min and max values during the update. Option A is syntactically correct but executes multiple subqueries within the UPDATE statement, which can be less efficient than joining with a temporary table. Option B is syntactically incorrect as aggregate functions (MIN, MAX) cannot be used directly in the UPDATE statement without a GROUP BY clause. Option C does not work as MIN_VAL and MAX VAL are column aliases that cannot be directly referenced as variables within the UPDATE statement. Option E is syntactically correct, but assumes that MIN VAL and MAX VAL variables are directly available during update, which is not how Snowflake SQL works.

NEW QUESTION # 139

......

Snowflake certification DSA-C03 exam questions and answers come out

There are several benefits of DSA-C03 Certification that can enjoy a successful candidate for the rest of their life, Customizable SnowPro Advanced: Data Scientist Certification Exam (DSA-C03) practice tests allow users set the time and DSA-C03 questions according to their needs.

So please prepare to get striking progress if you can get our DSA-C03 study guide with following traits for your information, Considerate services for our DSA-C03 learning materials: SnowPro Advanced: Data Scientist Certification Exam can be referred to as a large shining point.

We offer valuable Snowflake DSA-C03 exam materials with knowledge which is required to gain experience and to validate skills of the potential employers.