DATABRICKS-CERTIFIED-PROFESSIONAL-DATA-ENGINEER Q&As

Databricks Certified Professional Data Engineer Exam

Pass Databricks DATABRICKS-CERTIFIED-PROFESSIONAL-DATA-ENGINEER Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.geekcert.com/databricks-certified-professional-data-engineer.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Databricks Official Exam Center

QUESTION 1

A table in the Lakehouse named customer_churn_params is used in churn prediction by the machine learning team. The table contains information about customers derived from a number of upstream sources. Currently, the data engineering team populates this table nightly by overwriting the table with the current valid values derived from upstream data sources.

The churn prediction model used by the ML team is fairly stable in production. The team is only interested in making predictions on records that have changed in the past 24 hours.

Which approach would simplify the identification of these changed records?

A. Apply the churn model to all rows in the customer_churn_params table, but implement logic to perform an upsert into the predictions table that ignores rows where predictions have not changed.

B. Convert the batch job to a Structured Streaming job using the complete output mode; configure a Structured Streaming job to read from the customer_churn_params table and incrementally predict against the churn model.

C. Calculate the difference between the previous model predictions and the current customer_churn_params on a key identifying unique customers before making new predictions; only make predictions on those customers not in the previous predictions.

D. Modify the overwrite logic to include a field populated by calling spark.sql.functions.current_timestamp() as data are being written; use this field to identify records written on a particular date.

E. Replace the current overwrite logic with a merge statement to modify only those records that have changed; write logic to make predictions on the changed records identified by the change data feed.

Correct Answer: E

The approach that would simplify the identification of the changed records is to replace the current overwrite logic with a merge statement to modify only those records that have changed, and write logic to make predictions on the changed records identified by the change data feed. This approach leverages the Delta Lake features of merge and change data feed, which are designed to handle upserts and track row-level changes in a Delta table12. By using merge, the data engineering team can avoid overwriting the entire table every night, and only update or insert the records that have changed in the source data. By using change data feed, the ML team can easily access the change events that have occurred in the customer_churn_params table, and filter them by operation type (update or insert) and timestamp. This way, they can only make predictions on the records that have changed in the past 24 hours, and avoid re-processing the unchanged records. The other options are not as simple or efficient as the proposed approach, because: Option A would require applying the churn model to all rows in the customer_churn_params table, which would be wasteful and redundant. It would also require implementing logic to perform an upsert into the predictions table, which would be more complex than using the merge statement. Option B would require converting the batch job to a Structured Streaming job, which would involve changing the data ingestion and processing logic. It would also require using the complete output mode, which would output the entire result table every time there is a change in the source data, which would be inefficient and costly. Option C would require calculating the difference between the previous model predictions and the current customer_churn_params on a key identifying unique customers, which would be computationally expensive and prone to errors. It would also require storing and accessing the previous predictions, which would add extra storage and I/O costs. Option D would require modifying the overwrite logic to include a field populated by calling spark.sql.functions.current_timestamp() as data are being written, which would add extra complexity and overhead to the data engineering job. It would also require using this field to identify records written on a particular date, which would be less accurate and reliable than using the change data feed. References: Merge, Change data feed

QUESTION 2

A user new to Databricks is trying to troubleshoot long execution times for some pipeline logic they are working on. Presently, the user is executing code cell-by-cell, using display() calls to confirm code is producing the logically correct results as new transformations are added to an operation. To get a measure of average time to execute, the user is running each cell multiple times interactively.

Which of the following adjustments will get a more accurate measure of how code is likely to perform in production?

A. Scala is the only language that can be accurately tested using interactive notebooks; because the best performance is achieved by using Scala code compiled to JARs. all PySpark and Spark SQL logic should be refactored.

B. The only way to meaningfully troubleshoot code execution times in development notebooks Is to use production-sized data and production-sized clusters with Run All execution.

C. Production code development should only be done using an IDE; executing code against a local build of open source Spark and Delta Lake will provide the most accurate benchmarks for how code will perform in production.

D. Calling display () forces a job to trigger, while many transformations will only add to the logical query plan; because of caching, repeated execution of the same logic does not provide meaningful results.

E. The Jobs UI should be leveraged to occasionally run the notebook as a job and track execution time during incremental code development because Photon can only be enabled on clusters launched for scheduled jobs.

Correct Answer: D

In Databricks notebooks, using the display() function triggers an action that forces Spark to execute the code and produce a result. However, Spark operations are generally divided into transformations and actions. Transformations create a

new dataset from an existing one and are lazy, meaning they are not computed immediately but added to a logical plan. Actions, like display(), trigger the execution of this logical plan. Repeatedly running the same code cell can lead to

misleading performance measurements due to caching. When a dataset is used multiple times, Spark\\'s optimization mechanism caches it in memory, making subsequent executions faster. This behavior does not accurately represent the

first-time execution performance in a production environment where data might not be cached yet.

To get a more realistic measure of performance, it is recommended to:

Clear the cache or restart the cluster to avoid the effects of caching. Test the entire workflow end-to-end rather than cell-by-cell to understand the cumulative performance.

Consider using a representative sample of the production data, ensuring it includes various cases the code will encounter in production.

References:

Databricks Documentation on Performance Optimization: Databricks Performance Tuning

Apache Spark Documentation: RDD Programming Guide-Understanding transformations and actions

QUESTION 3

A team of data engineer are adding tables to a DLT pipeline that contain repetitive expectations for many of the same data quality checks.

One member of the team suggests reusing these data quality rules across all tables defined for this pipeline.

What approach would allow them to do this?

A. Maintain data quality rules in a Delta table outside of this pipeline\\'s target schema, providing the schema name as a pipeline parameter.

B. Use global Python variables to make expectations visible across DLT notebooks included in the same pipeline.

C. Add data quality constraints to tables in this pipeline using an external job with access to pipeline configuration files.

D. Maintain data quality rules in a separate Databricks notebook that each DLT notebook of file.

Correct Answer: A

Maintaining data quality rules in a centralized Delta table allows for the reuse of these rules across multiple DLT (Delta Live Tables) pipelines. By storing these rules outside the pipeline\\'s target schema and referencing the schema name as a pipeline parameter, the team can apply the same set of data quality checks to different tables within the pipeline. This approach ensures consistency in data quality validations and reduces redundancy in code by not having to replicate the same rules in each DLT notebook or file. References: Databricks Documentation on Delta Live Tables: Delta Live Tables Guide

QUESTION 4

An hourly batch job is configured to ingest data files from a cloud object storage container where each batch represent all records produced by the source system in a given hour. The batch job to process these records into the Lakehouse is sufficiently delayed to ensure no late-arriving data is missed. The user_id field represents a unique key for the data, which has the following schema:

user_id BIGINT, username STRING, user_utc STRING, user_region STRING, last_login BIGINT, auto_pay BOOLEAN, last_updated BIGINT

New records are all ingested into a table named account_history which maintains a full record of all data in the same schema as the source. The next table in the system is named account_current and is implemented as a Type 1 table representing the most recent value for each unique user_id.

Assuming there are millions of user accounts and tens of thousands of records processed hourly, which implementation can be used to efficiently update the described account_current table as part of each hourly batch job?

A. Use Auto Loader to subscribe to new files in the account history directory; configure a Structured Streaminq trigger once job to batch update newly detected files into the account current table.

B. Overwrite the account current table with each batch using the results of a query against the account history table grouping by user id and filtering for the max value of last updated.

C. Filter records in account history using the last updated field and the most recent hour processed, as well as the max last iogin by user id write a merge statement to update or insert the most recent value for each user id.

D. Use Delta Lake version history to get the difference between the latest version of account history and one version prior, then write these records to account current.

E. Filter records in account history using the last updated field and the most recent hour processed, making sure to deduplicate on username; write a merge statement to update or insert the most recent value for each username.

Correct Answer: C

This is the correct answer because it efficiently updates the account current table with only the most recent value for each user id. The code filters records in account history using the last updated field and the most recent hour processed, which means it will only process the latest batch of data. It also filters by the max last login by user id, which means it will only keep the most recent record for each user id within that batch. Then, it writes a merge statement to update or insert the most recent value for each user id into account current, which means it will perform an upsert operation based on the user id column. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Upsert into a table using merge" section.

QUESTION 5

A data engineer needs to capture pipeline settings from an existing in the workspace, and use them to create and version a JSON file to create a new pipeline.

Which command should the data engineer enter in a web terminal configured with the Databricks CLI?

A. Use the get command to capture the settings for the existing pipeline; remove the pipeline_id and rename the pipeline; use this in a create command

B. Stop the existing pipeline; use the returned settings in a reset command

C. Use the alone command to create a copy of an existing pipeline; use the get JSON command to get the pipeline definition; save this to git

D. Use list pipelines to get the specs for all pipelines; get the pipeline spec from the return results parse and use this to create a pipeline

Correct Answer: A

The Databricks CLI provides a way to automate interactions with Databricks services. When dealing with pipelines, you can use the databricks pipelines get--pipeline-id command to capture the settings of an existing pipeline in JSON format.

This JSON can then be modified by removing the pipeline_id to prevent conflicts and renaming the pipeline to create a new pipeline. The modified JSON file can then be used with the databricks pipelines create command to create a new

pipeline with those settings.

References:

Databricks Documentation on CLI for Pipelines: Databricks CLI-Pipelines