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 junior developer complains that the code in their notebook isn\\'t producing the correct results in the development environment. A shared screenshot reveals that while they\\'re using a notebook versioned with Databricks Repos, they\\'re using a personal branch that contains old logic. The desired branch nameddev-2.3.9is not available from the branch selection dropdown.

Which approach will allow this developer to review the current logic for this notebook?

A. Use Repos to make a pull request use the Databricks REST API to update the current branch to dev-2.3.9

B. Use Repos to pull changes from the remote Git repository and select the dev-2.3.9 branch.

C. Use Repos to checkout the dev-2.3.9 branch and auto-resolve conflicts with the current branch

D. Merge all changes back to the main branch in the remote Git repository and clone the repo again

E. Use Repos to merge the current branch and the dev-2.3.9 branch, then make a pull request to sync with the remote repository

Correct Answer: B

Explanation: This is the correct answer because it will allow the developer to update their local repository with the latest changes from the remote repository and switch to the desired branch. Pulling changes will not affect the current branch or create any conflicts, as it will only fetch the changes and not merge them. Selecting the dev-2.3.9 branch from the dropdown will checkout that branch and display its contents in the notebook. Verified References: [Databricks Certified Data Engineer Professional], under "Databricks Tooling" section; Databricks Documentation, under "Pull changes from a remote repository" section.

QUESTION 2

The data engineering team maintains the following code:

```
accountDF = spark.table("accounts")
orderDF = spark.table("orders")
itemDF = spark.table("items")

orderWithItemDF = (orderDF.join(
    itemDF,
    orderDF.itemID == itemDF.itemID)
  .select(
    orderDF.accountID,
    orderDF.itemID,

    itemDF.itemName))

finalDF = (accountDF.join(
    orderWithItemDF,
    accountDF.accountID == orderWithItemDF.accountID)
  .select(
    orderWithItemDF["*"],

    accountDF.city))

(finalDF.write
  .mode("overwrite")
  .table("enriched_itemized_orders_by_account"))
```

Assuming that this code produces logically correct results and the data in the source tables has been de-duplicated and validated, which statement describes what will occur when this code is executed?

A. A batch job will update the enriched_itemized_orders_by_account table, replacing only those rows that have different values than the current version of the table, using accountID as the primary key.

B. The enriched_itemized_orders_by_account table will be overwritten using the current valid version of data in each of the three tables referenced in the join logic.

C. An incremental job will leverage information in the state store to identify unjoined rows in the source tables and write these rows to the enriched_iteinized_orders_by_account table.

D. An incremental job will detect if new rows have been written to any of the source tables; if new rows are detected, all results will be recalculated and used to overwrite the enriched_itemized_orders_by_account table.

E. No computation will occur until enriched_itemized_orders_by_account is queried; upon query materialization, results will be calculated using the current valid version of data in each of the three tables referenced in the join logic.

Correct Answer: B

Explanation: This is the correct answer because it describes what will occur when this code is executed. The code uses three Delta Lake tables as input sources: accounts, orders, and order_items. These tables are joined together using SQL queries to create a view called new_enriched_itemized_orders_by_account, which contains information about each order item and its associated account details. Then, the code uses write.format("delta").mode("overwrite") to overwrite a target table called enriched_itemized_orders_by_account using the data from the view. This means that every time this code is executed, it will replace all existing data in the target table with new data based on the current valid version of data in each of the three input tables. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Write to Delta tables" section.

QUESTION 3

The viewupdatesrepresents an incremental batch of all newly ingested data to be inserted or updated in thecustomerstable.

The following logic is used to process these records.

```
MERGE INTO customers
USING (
  SELECT updates.customer_id as merge_ey, updates.*
  FROM updates

  UNION ALL

  SELECT NULL as merge_key, updates.*
  FROM updates JOIN customers
  ON updates.customer_id = customers.customer_id
  WHERE customers.current = true AND updates.address <> customers.address
) staged_updates
ON customers.customer_id = mergeKey
WHEN MATCHED AND customers.current = true AND customers.address <> staged_updates.address THEN
  UPDATE SET current = false, end_date = staged_updates.effective_date
WHEN NOT MATCHED THEN
  INSERT(customer_id, address, current, effective_date, end_date)
  VALUES(staged_updates.customer_id, staged_updates.address, true, staged_updates.effective_date,
null)
```

Which statement describes this implementation?

A. The customers table is implemented as a Type 3 table; old values are maintained as a new column alongside the current value.

B. The customers table is implemented as a Type 2 table; old values are maintained but marked as no longer current and new values are inserted.

C. The customers table is implemented as a Type 0 table; all writes are append only with no changes to existing values.

D. The customers table is implemented as a Type 1 table; old values are overwritten by new values and no history is maintained.

E. The customers table is implemented as a Type 2 table; old values are overwritten and new customers are appended.

Correct Answer: B

Explanation: The logic uses the MERGE INTO command to merge new records from the view updates into the table customers. The MERGE INTO command takes two arguments:

a target table and a source table or view. The command also specifies a condition to match records between the target and the source, and a set of actions to perform when there is a match or not. In this case, the condition is to match

records by customer_id, which is the primary key of the customers table. The actions are to update the existing record in the target with the new values from the source, and set the current_flag to false to indicate that the record is no longer

current; and to insert a new record in the target with the new values from the source, and set the current_flag to true to indicate that the record is current. This means that old values are maintained but marked as no longer current and new

values are inserted, which is the definition of a Type 2 table. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Merge Into (Delta Lake on Databricks)"

section.

QUESTION 4

The data science team has created and logged a production model using MLflow. The following code correctly imports and applies the production model to output the predictions as a new DataFrame namedpredswith the schema "customer_id LONG, predictions DOUBLE, date DATE".

```
from pyspark.sql.functions import current_date

model = mlflow.pyfunc.spark_udf(spark, model_uri="models:/churn/prod")
df = spark.table("customers")
columns = ["account_age", "time_since_last_seen", "app_rating"]
preds = (df.select(
  "customer_id",
  model(*columns).alias("predictions"),
  current_date().alias("date")
  )
```

The data science team would like predictions saved to a Delta Lake table with the ability to compare all predictions across time. Churn predictions will be made at most once per day. Which code block accomplishes this task while minimizing potential compute costs?

A. preds.write.mode("append").saveAsTable("churn_preds")
B. preds.write.format("delta").save("/preds/churn_preds") C)
C.

```
(preds.writeStream
  .outputMode("overwrite")
  .option("checkpointPath", "/_checkpoints/churn_preds")
  .start("/preds/churn_preds")
)
```

D.

```
(preds.write
  .format("delta")
  .mode("overwrite")
  .saveAsTable("churn_preds")
)
```

E.

```
(preds.writeStream
  .outputMode("append")
  .option("checkpointPath", "/_checkpoints/churn_preds")
  .table("churn_preds")
)
```

A. Option

B. Option

C. Option

D. Option

E. Option

Correct Answer: C

Explanation: This is the correct answer because it will save the predictions to a Delta Lake table with the ability to compare all predictions across time. The code uses the mergeInto method to perform an upsert operation, which means it will insert new records or update existing records based on the customer_id and date columns. This way, the table will always contain the latest predictions for each customer and date, and also keep the history of previous predictions. The code also uses a new job cluster to run the job, which will minimize the compute costs as it will be created and terminated for each run. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Upsert into a table using merge" section.

QUESTION 5

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, usingdisplay() 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

Explanation: This is the correct answer because it explains which of the following adjustments will get a more accurate measure of how code is likely to perform in production. The adjustment is that 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. When developing code in Databricks notebooks, one should be aware of how Spark handles transformations and actions. Transformations are operations that create a new DataFrame or Dataset from an existing one, such as filter, select, or join. Actions are operations that trigger a computation on a DataFrame or Dataset and return a result to the driver program or write it to storage, such as count, show, or save. Calling display() on a DataFrame or Dataset is also an action that triggers a computation and displays the result in a notebook cell. Spark uses lazy evaluation for transformations, which means that they are not executed until an action is called. Spark also uses caching to store intermediate results in memory or disk for faster access in subsequent actions. Therefore, 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. To get a more accurate measure of how code is likely to perform in production, one should avoid calling display() too often or clear the cache before running each cell. Verified References: [Databricks Certified Data Engineer Professional], under "Spark Core" section; Databricks Documentation, under "Lazy evaluation" section; Databricks Documentation, under "Caching" section.