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

The data science team has created and logged a production using MLFlow. The model accepts a list of column names and returns a new column of type DOUBLE.

The following code correctly imports the production model, load the customer table containing the customer_id key column into a Dataframe, and defines the feature columns needed for the model.

```
model = mlflow.pyfunc.spark_udf (spark,
model_uri="models:/churn/prod")

df = spark.table("customers")

columns = ["account_age", "time_since_last_seen", "app_rating"]
```

Which code block will output DataFrame with the schema\\'\\' customer_id LONG, predictions DOUBLE\\'\\'?

A. Model, predict (df, columns)

B. Df, map (lambda k:midel (x [columns]) ,select (\\'\\'customer_id predictions\\'\\')

C. Df. Select (\\'\\'customer_id\\'\\'. Model (\\'\\'columns) alias (\\'\\'predictions\\'\\')

D. Df.apply(model, columns). Select (\\'\\'customer_id, prediction\\'\\'

Correct Answer: A

Given the information that the model is registered with MLflow and assuming predict is the method used to apply the model to a set of columns, we use the model.predict() function to apply the model to the DataFrame df using the specified

columns. The model.predict() function is designed to take in a DataFrame and a list of column names as arguments, applying the trained model to these features to produce a predictions column. When working with PySpark, this predictions

column needs to be selected alongside the customer_id to create a new DataFrame with the schema customer_id LONG, predictions DOUBLE.

References:

MLflow documentation on using Python function models:

https://www.mlflow.org/docs/latest/models.html#python-function-python PySpark MLlib documentation on model prediction:

https://spark.apache.org/docs/latest/ml-pipeline.html#pipeline

QUESTION 2

A data engineer is configuring a pipeline that will potentially see late-arriving, duplicate records.

In addition to de-duplicating records within the batch, which of the following approaches allows the data engineer to deduplicate data against previously processed records as it is inserted into a Delta table?

A. Set the configuration delta.deduplicate = true.

B. VACUUM the Delta table after each batch completes.

C. Perform an insert-only merge with a matching condition on a unique key.

D. Perform a full outer join on a unique key and overwrite existing data.

E. Rely on Delta Lake schema enforcement to prevent duplicate records.

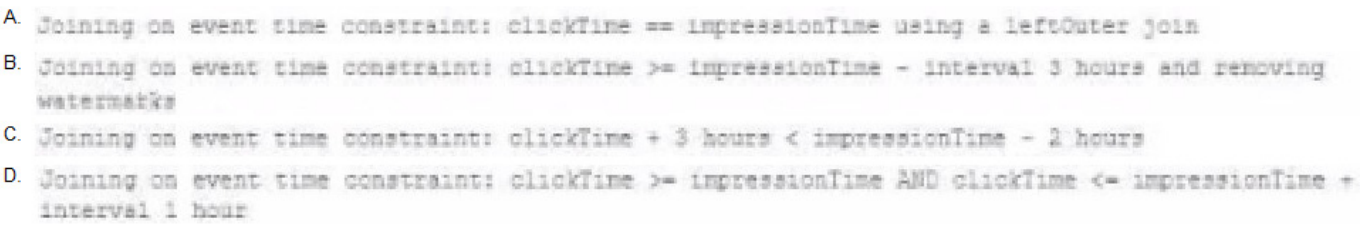
Correct Answer: C

To deduplicate data against previously processed records as it is inserted into a Delta table, you can use the merge operation with an insert-only clause. This allows you to insert new records that do not match any existing records based on a unique key, while ignoring duplicate records that match existing records. For example, you can use the following syntax: MERGE INTO target_table USING source_table ON target_table.unique_key = source_table.unique_key WHEN NOT MATCHED THEN INSERT * This will insert only the records from the source table that have a unique key that is not present in the target table, and skip the records that have a matching key. This way, you can avoid inserting duplicate records into the Delta table. References: https://docs.databricks.com/delta/delta-update.html#upsert-into-a-table-using-merge https://docs.databricks.com/delta/delta-update.html#insert-only-merge

QUESTION 3

A data engineer wants to join a stream of advertisement impressions (when an ad was shown) with another stream of user clicks on advertisements to correlate when impression led to monitizable clicks.

Which solution would improve the performance?



A. Option A

B. Option B

C. Option C

D. Option D

Correct Answer: A

When joining a stream of advertisement impressions with a stream of user clicks, you want to minimize the state that you need to maintain for the join. Option A suggests using a left outer join with the condition that clickTime == impressionTime, which is suitable for correlating events that occur at the exact same time. However, in a real-world scenario, you would likely need some leeway to account for the delay between an impression and a possible click. It\\'s important to design the join condition and the window of time considered to optimize performance while still capturing the relevant user interactions. In this case, having the watermark can help with state management and avoid state growing unbounded by discarding old state data that\\'s unlikely to match with new data.

QUESTION 4

Which statement describes Delta Lake optimized writes?

A. A shuffle occurs prior to writing to try to group data together resulting in fewer files instead of each executor writing multiple files based on directory partitions.

B. Optimized writes logical partitions instead of directory partitions partition boundaries are only represented in metadata fewer small files are written.

C. An asynchronous job runs after the write completes to detect if files could be further compacted; yes, an OPTIMIZE job is executed toward a default of 1 GB.

D. Before a job cluster terminates, OPTIMIZE is executed on all tables modified during the most recent job.

Correct Answer: A

Delta Lake optimized writes involve a shuffle operation before writing out data to the Delta table. The shuffle operation groups data by partition keys, which can lead to a reduction in the number of output files and potentially larger files, instead

of multiple smaller files. This approach can significantly reduce the total number of files in the table, improve read performance by reducing the metadata overhead, and optimize the table storage layout, especially for workloads with many

small files.

References:

Databricks documentation on Delta Lake performance tuning:

https://docs.databricks.com/delta/optimizations/auto-optimize.html

QUESTION 5

A Databricks job has been configured with 3 tasks, each of which is a Databricks notebook. Task A does not depend on other tasks. Tasks B and C run in parallel, with each having a serial dependency on Task A.

If task A fails during a scheduled run, which statement describes the results of this run?

A. Because all tasks are managed as a dependency graph, no changes will be committed to the Lakehouse until all tasks have successfully been completed.

B. Tasks B and C will attempt to run as configured; any changes made in task A will be rolled back due to task failure.

C. Unless all tasks complete successfully, no changes will be committed to the Lakehouse; because task A failed, all commits will be rolled back automatically.

D. Tasks B and C will be skipped; some logic expressed in task A may have been committed before task failure.

E. Tasks B and C will be skipped; task A will not commit any changes because of stage failure.

Correct Answer: D

When a Databricks job runs multiple tasks with dependencies, the tasks are executed in a dependency graph. If a task fails, the downstream tasks that depend on it are skipped and marked as Upstream failed. However, the failed task may have already committed some changes to the Lakehouse before the failure occurred, and those changes are not rolled back automatically. Therefore, the job run may result in a partial update of the Lakehouse. To avoid this, you can use the transactional writes feature of Delta Lake to ensure that the changes are only committed when the entire job run succeeds. Alternatively, you can use the Run if condition to configure tasks to run even when some or all of their dependencies have failed, allowing your job to recover from failures and continue running. References: transactional writes: https://docs.databricks.com/delta/delta-intro.html#transactional-writes Run if: https://docs.databricks.com/en/workflows/jobs/conditional-tasks.html