DATABRICKS-CERTIFIED-PR OFESSIONAL-DATA-ENGINEER$^{Q\&As}$

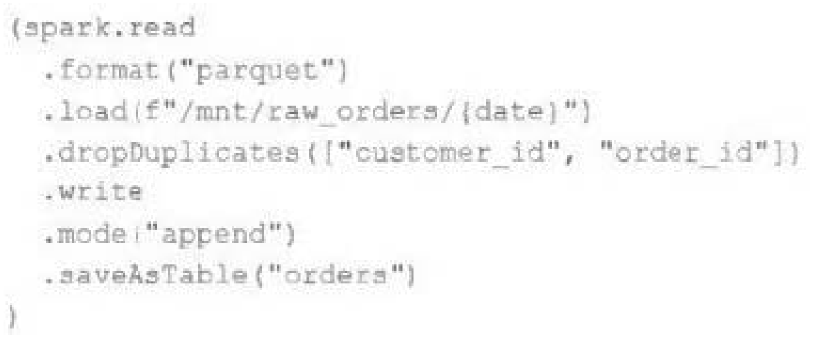
Databricks Certified Professional Data Engineer Exam

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

SATISFACTION GUARANTEED
100%
SATISFACTION GUARANTEED

QUESTION 1

An upstream source writes Parquet data as hourly batches to directories named with the current date. A nightly batch job runs the following code to ingest all data from the previous day as indicated by thedatevariable:



```
(spark.read
  .format("parquet")
  .load(f"/mnt/raw_orders/{date}")
  .dropDuplicates(["customer_id", "order_id"])
  .write
  .mode("append")
  .saveAsTable("orders")
)
```

Assume that the fieldscustomer_idandorder_idserve as a composite key to uniquely identify each order.

If the upstream system is known to occasionally produce duplicate entries for a single order hours apart, which statement is correct?

A. Each write to the orders table willonly contain unique records, and only those records without duplicates in the target table will be written.

B. Each write to the orders table willonly contain unique records, but newly written records may have duplicates already present in the target table.

C. Each write to the orders table will only contain unique records; if existing records with the same key are present in the target table, these records will be overwritten.

D. Each write to the orders table will only contain unique records; if existing records with the same key are present in the target table, the operation will tail.

E. Each write to the orders table willrun deduplication over the union of new and existing records, ensuringno duplicate records are present.

Correct Answer: B

Explanation: This is the correct answer because the code uses the dropDuplicates method to remove any duplicate records within each batch of data before writing to the orders table. However, this method does not check for duplicates across different batches or in the target table, so it is possible that newly written records may have duplicates already present in the target table. To avoid this, a better approach would be to use Delta Lake and perform an upsert operation usingmergeInto. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "DROP DUPLICATES" section.

QUESTION 2

A junior data engineer is working to implement logic for a Lakehouse table namedsilver_device_recordings. The source data contains 100 unique fields in a highly nested JSON structure.

Thesilver_device_recordingstable will be used downstream to power several production monitoring dashboards and a production model. At present, 45 of the 100 fields are being used in at least one of these applications.

The data engineer is trying to determine the best approach for dealing with schema declaration given the highly-nested structure of the data and the numerous fields.

Which of the following accurately presents information about Delta Lake and Databricks that may impact their decision-making process?

A. The Tungsten encodingused by Databricksis optimized for storing stringdata; newly- added nativesupport for querying JSON strings means that stringtypes are always most efficient.

B. Because Delta Lake uses Parquet for data storage, data types can be easily evolved by just modifying file footer information in place.

C. Human labor in writingcode is the largest cost associated with data engineering workloads; as such, automatingtable declaration logic should be a priority in all migration workloads.

D. Because Databricks will infer schema using types that allow all observed data to be processed, setting types manually provides greater assurance of data quality enforcement.

E. Schema inference and evolution on .Databricks ensure that inferred types will always accurately match the data types used by downstream systems.

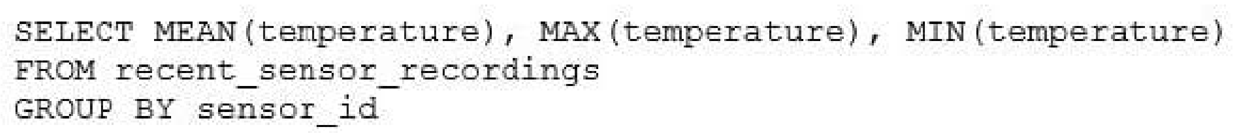
Correct Answer: D

Explanation: This is the correct answer because it accurately presents information about Delta Lake and Databricks that may impact the decision-making process of a junior data engineer who is trying to determine the best approach for dealing with schema declaration given the highly-nested structure of the data and the numerous fields. Delta Lake and Databricks support schema inference and evolution, which means that they can automatically infer the schema of a table from the source data and allow adding new columns or changing column types without affecting existing queries or pipelines. However, schema inference and evolution may not always be desirable or reliable, especially when dealing with complex or nested data structures or when enforcing data quality and consistency across different systems. Therefore, setting types manually can provide greater assurance of data quality enforcement and avoid potential errors or conflicts due to incompatible or unexpected data types. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Schema inference and partition of streaming DataFrames/ Datasets" section.

QUESTION 3

The data engineering team has configured a Databricks SQL query and alert to monitor the values in a Delta Lake table. Therecent_sensor_recordingstable contains an identifyingsensor_idalongside thetimestampandtemperaturefor the most recent 5 minutes of recordings.

The below query is used to create the alert:



```
SELECT MEAN(temperature), MAX(temperature), MIN(temperature)
FROM recent_sensor_recordings
GROUP BY sensor_id
```

The query is set to refresh each minute and always completes in less than 10 seconds. The alert is set to trigger whenmean (temperature) > 120. Notifications are triggered to be sent at most every 1 minute.

If this alert raises notifications for 3 consecutive minutes and then stops, which statement must be true?

A. The total average temperature across all sensors exceeded 120 on three consecutive executions of the query

B. Therecent_sensor_recordingstable was unresponsive for three consecutive runs of the query

C. The source query failed to update properly for three consecutive minutes and then restarted

D. The maximum temperature recording for at least one sensor exceeded 120 on three consecutive executions of the query

E. The average temperature recordings for at least one sensor exceeded 120 on three consecutive executions of the query

Correct Answer: E

Explanation: This is the correct answer because the query is using a GROUP BY clause on the sensor_id column, which means it will calculate the mean temperature for each sensor separately. The alert will trigger when the mean temperature for any sensor is greater than 120, which means at least one sensor had an average temperature above 120 for three consecutive minutes. The alert will stop when the mean temperature for all sensors drops below 120. Verified References: [Databricks Certified Data Engineer Professional], under "SQL Analytics" section; Databricks Documentation, under "Alerts" section.

QUESTION 4

Which configuration parameter directly affects the size of a spark-partition upon ingestion of data into Spark?

A. spark.sql.files.maxPartitionBytes

B. spark.sql.autoBroadcastJoinThreshold

C. spark.sql.files.openCostInBytes

D. spark.sql.adaptive.coalescePartitions.minPartitionNum

E. spark.sql.adaptive.advisoryPartitionSizeInBytes

Correct Answer: A

Explanation: This is the correct answer because spark.sql.files.maxPartitionBytes is a configuration parameter that directly affects the size of a spark-partition upon ingestion of data into Spark. This parameter configures the maximum number

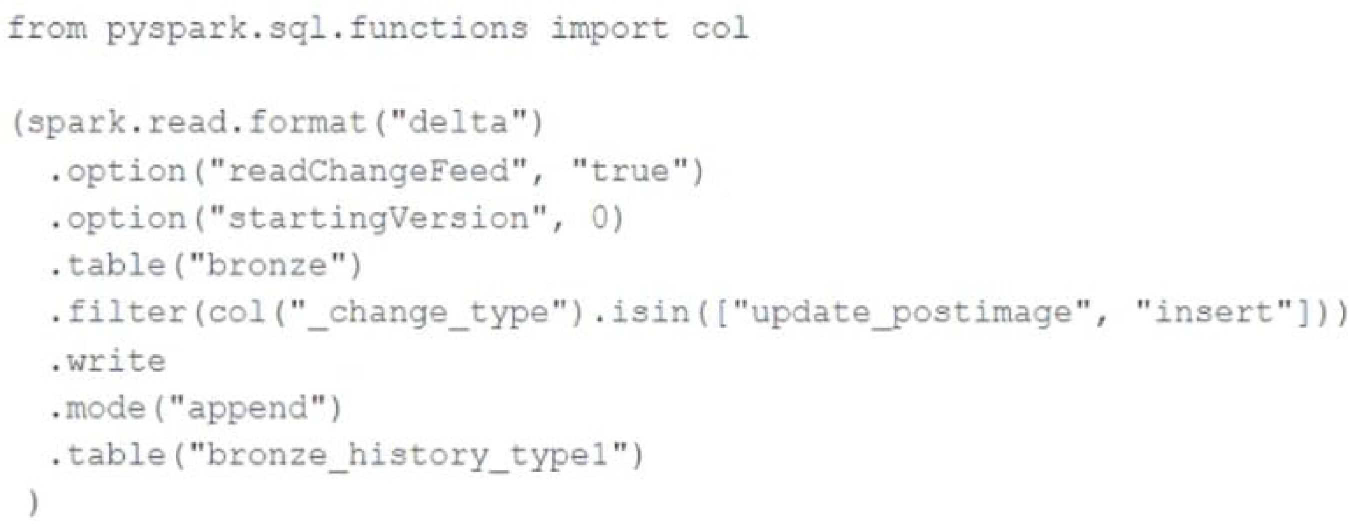
of bytes to pack into a single partition when reading files from file-based sources such as Parquet, JSON and ORC. The default value is 128 MB, which means each partition will be roughly 128 MB in size, unless there are too many small files

or only one large file. Verified References:

[Databricks Certified Data Engineer Professional], under "Spark Configuration" section; Databricks Documentation, under "Available Properties - spark.sql.files.maxPartitionBytes" section.

QUESTION 5

A junior data engineer seeks to leverage Delta Lake\\'s Change Data Feed functionality to create a Type 1 table representing all of the values that have ever been valid for all rows in abronzetable created with the propertydelta.enableChangeDataFeed = true. They plan to execute the following code as a daily job:



```
from pyspark.sql.functions import col

(spark.read.format("delta")
  .option("readChangeFeed", "true")
  .option("startingVersion", 0)
  .table("bronze")
  .filter(col("_change_type").isin(["update_postimage", "insert"]))
  .write
  .mode("append")
  .table("bronze_history_type1")
 )
```

Which statement describes the execution and results of running the above query multiple times?

A. Each time the job is executed, newly updated records will be merged into the target table, overwriting previous values with the same primary keys.

B. Each time the job is executed, the entire available history of inserted or updated records will be appended to the target table, resulting in many duplicate entries.

C. Each time the job is executed, the target table will be overwritten using the entire history of inserted or updated records, giving the desired result.

D. Each time the job is executed, the differences between the original and current versions are calculated; this may result in duplicate entries for some records.

E. Each time the job is executed, only those records that have been inserted or updated since the last execution will be appended to the target table giving the desired result.

Correct Answer: C

Explanation: This is the correct answer because it describes the execution and results of running the above query multiple times. The query uses the readChanges function to read all change events from a bronze table that has enabled change data feed. The readChanges function takes two arguments: version and options. The version argument specifies which version of the table to read changes from, and can be either a specific version number or -1 to indicate all versions. The options argument specifies additional options for reading changes, such as whether to include deletes or not. In this case, the query reads all changes from all versions of the bronze table and filters out delete events by setting includeDeletes to false. Then, it uses write.format("delta").mode("overwrite") to overwrite a target table using the entire history of inserted or updated records, giving the desired result of a Type 1 table representing all values that have ever been valid for all rows in the bronze table. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Read changes in batch queries" section.