

DATABRICKS-CERTIFIED- PR OFSSIONAL-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.leads4pass.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

- ⚙️ **Instant Download** After Purchase
- ⚙️ **100% Money Back** Guarantee
- ⚙️ **365 Days** Free Update
- ⚙️ **800,000+** Satisfied Customers



QUESTION 1

Where in the Spark UI can one diagnose a performance problem induced by not leveraging predicate push-down?

- A. In the Executor's log file, by gripping for "predicate push-down"
- B. In the Stage's Detail screen, in the Completed Stages table, by noting the size of data read from the Input column
- C. In the Storage Detail screen, by noting which RDDs are not stored on disk
- D. In the Delta Lake transaction log. by noting the column statistics
- E. In the Query Detail screen, by interpreting the Physical Plan

Correct Answer: E

Explanation: This is the correct answer because it is where in the Spark UI one can diagnose a performance problem induced by not leveraging predicate push-down. Predicate push-down is an optimization technique that allows filtering data at the source before loading it into memory or processing it further. This can improve performance and reduce I/O costs by avoiding reading unnecessary data. To leverage predicate push-down, one should use supported data sources and formats, such as Delta Lake, Parquet, or JDBC, and use filter expressions that can be pushed down to the source. To diagnose a performance problem induced by not leveraging predicate push-down, one can use the Spark UI to access the Query Detail screen, which shows information about a SQL query executed on a Spark cluster. The Query Detail screen includes the Physical Plan, which is the actual plan executed by Spark to perform the query. The Physical Plan shows the physical operators used by Spark, such as Scan, Filter, Project, or Aggregate, and their input and output statistics, such as rows and bytes. By interpreting the Physical Plan, one can see if the filter expressions are pushed down to the source or not, and how much data is read or processed by each operator. Verified References: [Databricks Certified Data Engineer Professional], under "Spark Core" section; Databricks Documentation, under "Predicate pushdown" section; Databricks Documentation, under "Query detail page" section.

QUESTION 2

A data architect has designed a system in which two Structured Streaming jobs will concurrently write to a single bronze Delta table. Each job is subscribing to a different topic from an Apache Kafka source, but they will write data with the same schema. To keep the directory structure simple, a data engineer has decided to nest a checkpoint directory to be shared by both streams.

The proposed directory structure is displayed below:

```
./bronze
├── __checkpoint
├── __delta_log
├── year_week=2020_01
├── year_week=2020_02
└── ...
```

Which statement describes whether this checkpoint directory structure is valid for the given scenario and why?

- A. No; Delta Lake manages streaming checkpoints in the transaction log.
- B. Yes; both of the streams can share a single checkpoint directory.
- C. No; only one stream can write to a Delta Lake table.
- D. Yes; Delta Lake supports infinite concurrent writers.
- E. No; each of the streams needs to have its own checkpoint directory.

Correct Answer: E

Explanation: This is the correct answer because checkpointing is a critical feature of Structured Streaming that provides fault tolerance and recovery in case of failures. Checkpointing stores the current state and progress of a streaming query in a reliable storage system, such as DBFS or S3. Each streaming query must have its own checkpoint directory that is unique and exclusive to that query. If two streaming queries share the same checkpoint directory, they will interfere with each other and cause unexpected errors or data loss. Verified References: [Databricks Certified Data Engineer Professional], under "Structured Streaming" section; Databricks Documentation, under "Checkpointing" section.

QUESTION 3

A table is registered with the following code:

```
CREATE TABLE recent_orders AS (  
  SELECT a.user_id, a.email, b.order_id, b.order_date  
  FROM  
    (SELECT user_id, email  
     FROM users) a  
  INNER JOIN  
    (SELECT user_id, order_id, order_date  
     FROM orders  
     WHERE order_date >= (current_date() - 7)) b  
  ON a.user_id = b.user_id  
)
```

Both users and orders are Delta Lake tables. Which statement describes the results of querying recent_orders?

- A. All logic will execute at query time and return the result of joining the valid versions of the source tables at the time the query finishes.
- B. All logic will execute when the table is defined and store the result of joining tables to the DBFS; this stored data will be returned when the table is queried.
- C. Results will be computed and cached when the table is defined; these cached results will incrementally update as new records are inserted into source tables.
- D. All logic will execute at query time and return the result of joining the valid versions of the source tables at the time the query began.

E. The versions of each source table will be stored in the table transaction log; query results will be saved to DBFS with each query.

Correct Answer: D

Explanation: This is the correct answer because Delta Lake supports time travel, which allows users to query data as of a specific version or timestamp. The code uses the VERSION AS OF syntax to specify the version of each source table to be used in the join. The result of querying recent_orders will be the same as joining those versions of the source tables at query time. The query will use snapshot isolation, which means it will use a consistent snapshot of the table at the time the query began, regardless of any concurrent updates or deletes. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Query an older snapshot of a table (time travel)" section.

QUESTION 4

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 identifying sensor_id alongside the timestamp and temperature for 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 when mean (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 5

A junior data engineer has configured a workload that posts the following JSON to the Databricks REST API endpoint `2.0/jobs/create`.

```
{
  "name": "Ingest new data",
  "existing_cluster_id": "6015-954420-peace720",
  "notebook_task": {
    "notebook_path": "/Prod/ingest.py"
  }
}
```

Assuming that all configurations and referenced resources are available, which statement describes the result of executing this workload three times?

- A. Three new jobs named "Ingest new data" will be defined in the workspace, and they will each run once daily.
- B. The logic defined in the referenced notebook will be executed three times on new clusters with the configurations of the provided cluster ID.
- C. Three new jobs named "Ingest new data" will be defined in the workspace, but no jobs will be executed.
- D. One new job named "Ingest new data" will be defined in the workspace, but it will not be executed.
- E. The logic defined in the referenced notebook will be executed three times on the referenced existing all purpose cluster.

Correct Answer: C

Explanation: This is the correct answer because the JSON posted to the Databricks REST API endpoint `2.0/jobs/create` defines a new job with a name, an existing cluster id, and a notebook task. However, it does not specify any schedule or trigger for the job execution. Therefore, three new jobs with the same name and configuration will be created in the workspace, but none of them will be executed until they are manually triggered or scheduled. Verified References: [Databricks Certified Data Engineer Professional], under "Monitoring and Logging" section; [Databricks Documentation], under "Jobs API - Create" section.

[Latest DATABRICKS-CERTIFIED-PROFESSIONAL-DATA-ENGINEER Dumps](#)

[DATABRICKS-CERTIFIED-PROFESSIONAL-DATA-ENGINEER Exam Questions](#)

[DATABRICKS-CERTIFIED-PROFESSIONAL-DATA-ENGINEER Braindumps](#)