Free Questions for Databricks-Certified-Professional-Data-Engineer by ebraindumps

Shared by Ball on 21-03-2024

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

The Databricks CLI is use to trigger a run of an existing job by passing the job_id parameter. The response that the job run request has been submitted successfully includes a filed run_id.

Which statement describes what the number alongside this field represents?

Options:

A- The job_id is returned in this field.

B- The job_id and number of times the job has been are concatenated and returned.

C- The number of times the job definition has been run in the workspace.

D- The globally unique ID of the newly triggered run.

Answer:

D

Explanation:

When triggering a job run using the Databricks CLI, the run_id field in the response represents a globally unique identifier for that particular run of the job. This run_id is distinct from the job_id. While the job_id identifies the job definition and is constant across all runs of that job, the run_id is unique to each execution and is used to track and query the status of that specific job run within the Databricks environment. This distinction allows users to manage and reference individual executions of a job directly.

Question 2

Question Type: MultipleChoice

What is the first of a Databricks Python notebook when viewed in a text editor?

Options:

A- %python

B- % Databricks notebook source

C- -- Databricks notebook source

D- //Databricks notebook source

Answer:

B

Explanation:

When viewing a Databricks Python notebook in a text editor, the first line indicates the format and source type of the notebook. The correct option is % Databricks notebook source, which is a magic command that specifies the start of a Databricks notebook source file.

Question 3

Question Type: MultipleChoice

The data engineer is using Spark's MEMORY_ONLY storage level.

Which indicators should the data engineer look for in the spark UI's Storage tab to signal that a cached table is not performing optimally?

Options:

A- Size on Disk is> 0

B- The number of Cached Partitions> the number of Spark Partitions

C- The RDD Block Name included the '' annotation signaling failure to cache

D- On Heap Memory Usage is within 75% of off Heap Memory usage

Answer:

C

Explanation:

In the Spark UI's Storage tab, an indicator that a cached table is not performing optimally would be the presence of the _disk annotation in the RDD Block Name. This annotation indicates that some partitions of the cached data have been spilled to disk because there wasn't enough memory to hold them. This is suboptimal because accessing data from disk is much slower than from memory. The goal of caching is to keep data in memory for fast access, and a spill to disk means that this goal is not fully achieved.

Question 4

Question Type: MultipleChoice

A data engineer wants to reflector the following DLT code, which includes multiple definition with very similar code:

```
@dlt.table(name=f"t1_dataset")
def t1_dataset():
    return spark.read.table(t1)

@dlt.table(name=f"t2_dataset")
def t2_dataset():
    return spark.read.table(t2)

@dlt.table(name=f"t3_dataset")
def t3_dataset():
    return spark.read.table(t3)

...
```

In an attempt to programmatically create these tables using a parameterized table definition, the data engineer writes the following code.

```
tables = ["t1", "t2", "t3"]

for t in tables:
    @dlt.table(name=f"{t}_dataset")
        def new_table():
```

The pipeline runs an update with this refactored code, but generates a different DAG showing incorrect configuration values for tables.

How can the data engineer fix this?

Options:

A- Convert the list of configuration values to a dictionary of table settings, using table names as keys.

B- Convert the list of configuration values to a dictionary of table settings, using different input the for loop.

C- Load the configuration values for these tables from a separate file, located at a path provided by a pipeline parameter.

D- Wrap the loop inside another table definition, using generalized names and properties to replace with those from the inner table

Answer:

A

Explanation:

The issue with the refactored code is that it tries to use string interpolation to dynamically create table names within the dlc.table decorator, which will not correctly interpret the table names. Instead, by using a dictionary with table names as keys and their configurations as values, the data engineer can iterate over the dictionary items and use the keys (table names) to properly configure the table settings. This way, the decorator can correctly recognize each table name, and the corresponding configuration settings can be applied appropriately.

Question 5

Question Type: MultipleChoice

The data governance team is reviewing user for deleting records for compliance with GDPR. The following logic has been implemented to propagate deleted requests from the user_lookup table to the user aggregate table.

```
(spark.read
  .format("delta")
  .option("readChangeData", True)
  .option("startingTimestamp", '2021-08-22 00:00:00')
  .option("endingTimestamp", '2021-08-29 00:00:00')
  .table("user_lookup")
  .createOrReplaceTempView("changes"))

spark.sql("""
  DELETE FROM user_aggregates
  WHERE user_id IN (
    SELECT user_id
    FROM changes
    WHERE _change_type='delete'
    )
""")
```

Assuming that user_id is a unique identifying key and that all users have requested deletion have been removed from the user_lookup table, which statement describes whether successfully executing the above logic guarantees that the records to be deleted from the user_aggregates table are no longer accessible and why?

Options:

A- No: files containing deleted records may still be accessible with time travel until a BACUM command is used to remove invalidated data files.

B- Yes: Delta Lake ACID guarantees provide assurance that the DELETE command successed fully and permanently purged these records.

C- No: the change data feed only tracks inserts and updates not deleted records.

D- No: the Delta Lake DELETE command only provides ACID guarantees when combined with the MERGE INTO command

Answer:

A

Explanation:

The DELETE operation in Delta Lake is ACID compliant, which means that once the operation is successful, the records are logically removed from the table. However, the underlying files that contained these records may still exist and be accessible via time travel to older versions of the table. To ensure that these records are physically removed and compliance with GDPR is maintained, a VACUUM command should be used to clean up these data files after a certain retention period. The VACUUM command will remove the files from the storage layer, and after this, the records will no longer be accessible.

Question 6

Question Type: MultipleChoice

A table named user_ltv is being used to create a view that will be used by data analysis on various teams. Users in the workspace are configured into groups, which are used for setting up data access using ACLs.

The user_ltv table has the following schema:

```
email STRING, age INT, ltv INT
```

The following view definition is executed:

```
CREATE VIEW user_ltv_no_minors AS
SELECT email, age, ltv
FROM user_ltv
WHERE
  CASE
    WHEN is_member("auditing") THEN TRUE
    ELSE age >= 18
  END
```

An analyze who is not a member of the auditing group executing the following query:

```
SELECT * FROM user_ltv_no_minors
```

Which result will be returned by this query?

Options:

A- All columns will be displayed normally for those records that have an age greater than 18; records not meeting this condition will be omitted.

B- All columns will be displayed normally for those records that have an age greater than 17; records not meeting this condition will be omitted.

C- All age values less than 18 will be returned as null values all other columns will be returned with the values in user_ltv.

D- All records from all columns will be displayed with the values in user_ltv.

Answer:

A

Explanation:

Given the CASE statement in the view definition, the result set for a user not in the auditing group would be constrained by the ELSE condition, which filters out records based on age. Therefore, the view will return all columns normally for records with an age greater than 18, as users who are not in the auditing group will not satisfy the is_member('auditing') condition. Records not meeting the age > 18 condition will not be displayed.

Question 7

Question Type: MultipleChoice

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.

```
In the code below, impressions is a streaming DataFrame with a watermark ("event_time", "10 minutes")
.groupBy(
window("event_time", "5 minutes"),
"id")
.count()
).     withWatermark("event_time", 2 hours)
impressions.join(clicks, expr("clickAdId = impressionAdId"), "inner")
```

Which solution would improve the performance?

A)

```
Joining on event time constraint: clickTime == impressionTime using a leftOuter join
```

B)

```
Joining on event time constraint: clickTime >= impressionTime - interval 3 hours and removing
watermarks
```

C)

```
Joining on event time constraint: clickTime + 3 hours < impressionTime - 2 hours
```

D)

```
Joining on event time constraint: clickTime >= impressionTime AND clickTime <= impressionTime +
interval 1 hour
```

Options:

A- Option A

B- Option B

C- Option C

D- Option D

Answer:

A

Explanation:

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 8

Question Type: MultipleChoice

A data architect has heard about lake's built-in versioning and time travel capabilities. For auditing purposes they have a requirement to maintain a full of all valid street addresses as they appear in the customers table.

The architect is interested in implementing a Type 1 table, overwriting existing records with new values and relying on Delta Lake time travel to support long-term auditing. A data engineer on the project feels that a Type 2 table will provide better performance and scalability.

Which piece of information is critical to this decision?

Options:

A- Delta Lake time travel does not scale well in cost or latency to provide a long-term versioning solution.

B- Delta Lake time travel cannot be used to query previous versions of these tables because Type 1 changes modify data files in place.

C- Shallow clones can be combined with Type 1 tables to accelerate historic queries for long-term versioning.

D- Data corruption can occur if a query fails in a partially completed state because Type 2 tables requires
Setting multiple fields in a single update.

Answer:

A

Explanation:

Delta Lake's time travel feature allows users to access previous versions of a table, providing a powerful tool for auditing and versioning. However, using time travel as a long-term versioning solution for auditing purposes can be less optimal in terms of cost and performance, especially as the volume of data and the number of versions grow. For maintaining a full history of valid street addresses as they appear in a customers table, using a Type 2 table (where each update creates a new record with versioning) might provide better scalability and performance by avoiding the overhead associated with accessing older versions of a large table. While Type 1 tables, where existing records are overwritten with new values, seem simpler and can leverage time travel for auditing, the critical piece of information is that time travel might not scale well in cost or latency for long-term versioning needs, making a Type 2 approach more viable for performance and scalability. Reference:

Databricks Documentation on Delta Lake's Time Travel: Delta Lake Time Travel

Databricks Blog on Managing Slowly Changing Dimensions in Delta Lake: Managing SCDs in Delta Lake

Question 9

Question Type: MultipleChoice

A junior data engineer is migrating a workload from a relational database system to the Databricks Lakehouse. The source system uses a star schema, leveraging foreign key constrains and multi-table inserts to validate records on write.

Which consideration will impact the decisions made by the engineer while migrating this workload?

Options:

A- All Delta Lake transactions are ACID compliance against a single table, and Databricks does not enforce foreign key constraints.

B- Databricks only allows foreign key constraints on hashed identifiers, which avoid collisions in highly-parallel writes.

C- Foreign keys must reference a primary key field; multi-table inserts must leverage Delta Lake's upsert functionality.

D- Committing to multiple tables simultaneously requires taking out multiple table locks and can lead to a state of deadlock.

Answer:

A

Explanation:

In Databricks and Delta Lake, transactions are indeed ACID-compliant, but this compliance is limited to single table transactions. Delta Lake does not inherently enforce foreign key constraints, which are a staple in relational database systems for maintaining referential integrity between tables. This means that when migrating workloads from a relational database system to Databricks Lakehouse, engineers need to reconsider how to maintain data integrity and relationships that were previously enforced by foreign key constraints.

Unlike traditional relational databases where foreign key constraints help in maintaining the consistency across tables, in Databricks Lakehouse, the data engineer has to manage data consistency and integrity at the application level or through careful design of ETL processes. Reference:

Databricks Documentation on Delta Lake: Delta Lake Guide

Databricks Documentation on ACID Transactions in Delta Lake: ACID Transactions in Delta Lake

Question 10

Question Type: **MultipleChoice**

The marketing team is looking to share data in an aggregate table with the sales organization, but the field names used by the teams do not match, and a number of marketing specific fields have not been approval for the sales org.

Which of the following solutions addresses the situation while emphasizing simplicity?

Options:

A- Create a view on the marketing table selecting only these fields approved for the sales team alias the names of any fields that should be standardized to the sales naming conventions.

B- Use a CTAS statement to create a derivative table from the marketing table configure a production jon to propagation changes.

C- Add a parallel table write to the current production pipeline, updating a new sales table that varies as required from marketing table.

D- Create a new table with the required schema and use Delta Lake's DEEP CLONE functionality to sync up changes committed to one table to the corresponding table.

Answer:

A

Explanation:

Creating a view is a straightforward solution that can address the need for field name standardization and selective field sharing between departments. A view allows for presenting a transformed version of the underlying data without duplicating it. In this scenario, the view would only include the approved fields for the sales team and rename any fields as per their naming conventions.

Databricks documentation on using SQL views in Delta Lake: https://docs.databricks.com/delta/quick-start.html#sql-views