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

Shared by Collins on 22-07-2024

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

The DevOps team has configured a production workload as a collection of notebooks scheduled to run daily using the Jobs UI. A new data engineering hire is onboarding to the team and has requested access to one of these notebooks to review the production logic.

What are the maximum notebook permissions that can be granted to the user without allowing accidental changes to production code or data?

Options:

A- Can manage

B- Can edit

C- Can run

D- Can Read

Answer:

D

Explanation:

Granting a user 'Can Read' permissions on a notebook within Databricks allows them to view the notebook's content without the ability to execute or edit it. This level of permission ensures that the new team member can review the production logic for learning or auditing purposes without the risk of altering the notebook's code or affecting production data and workflows. This approach aligns with best practices for maintaining security and integrity in production environments, where strict access controls are essential to prevent unintended modifications. Reference: Databricks documentation on access control and permissions for notebooks within the workspace (https://docs.databricks.com/security/access-control/workspace-acl.html).

Question 2

Question Type: **MultipleChoice**

A user wants to use DLT expectations to validate that a derived table report contains all records from the source, included in the table validation_copy.

The user attempts and fails to accomplish this by adding an expectation to the report table definition.

Which approach would allow using DLT expectations to validate all expected records are present in this table?

Options:

A- Define a SQL UDF that performs a left outer join on two tables, and check if this returns null values for report key values in a DLT

expectation for the report table.

B- Define a function that performs a left outer join on validation_copy and report and report, and check against the result in a DLT expectation for the report table

C- Define a temporary table that perform a left outer join on validation_copy and report, and define an expectation that no report key values are null

D- Define a view that performs a left outer join on validation_copy and report, and reference this view in DLT expectations for the report table

Answer:

D

Explanation:

To validate that all records from the source are included in the derived table, creating a view that performs a left outer join between the validation_copy table and the report table is effective. The view can highlight any discrepancies, such as null values in the report table's key columns, indicating missing records. This view can then be referenced in DLT (Delta Live Tables) expectations for the report table to ensure data integrity. This approach allows for a comprehensive comparison between the source and the derived table.

Databricks Documentation on Delta Live Tables and Expectations: Delta Live Tables Expectations

Question 3

Question Type: MultipleChoice

The data engineer team has been tasked with configured connections to an external database that does not have a supported native connector with Databricks. The external database already has data security configured by group membership. These groups map directly to user group already created in Databricks that represent various teams within the company.

A new login credential has been created for each group in the external database. The Databricks Utilities Secrets module will be used to make these credentials available to Databricks users.

Assuming that all the credentials are configured correctly on the external database and group membership is properly configured on Databricks, which statement describes how teams can be granted the minimum necessary access to using these credentials?

Options:

A- "Read" permissions should be set on a secret key mapped to those credentials that will be used by a given team.

B- No additional configuration is necessary as long as all users are configured as administrators in the workspace where secrets have been added.

C- "Read" permissions should be set on a secret scope containing only those credentials that will be used by a given team.

D- "Manage" permission should be set on a secret scope containing only those credentials that will be used by a given team.

Answer:

C

Explanation:

In Databricks, using the Secrets module allows for secure management of sensitive information such as database credentials. Granting 'Read' permissions on a secret key that maps to database credentials for a specific team ensures that only members of that team can access these credentials. This approach aligns with the principle of least privilege, granting users the minimum level of access required to perform their jobs, thus enhancing security.

Databricks Documentation on Secret Management: Secrets

Question 4

Question Type: MultipleChoice

A team of data engineer are adding tables to a DLT pipeline that contain repetitive expectations for many of the same data quality checks.

One member of the team suggests reusing these data quality rules across all tables defined for this pipeline.

What approach would allow them to do this?

Options:

A- Maintain data quality rules in a Delta table outside of this pipeline's target schema, providing the schema name as a pipeline parameter.

B- Use global Python variables to make expectations visible across DLT notebooks included in the same pipeline.

C- Add data quality constraints to tables in this pipeline using an external job with access to pipeline configuration files.

D- Maintain data quality rules in a separate Databricks notebook that each DLT notebook of file.

Answer:

A

Explanation:

Maintaining data quality rules in a centralized Delta table allows for the reuse of these rules across multiple DLT (Delta Live Tables) pipelines. By storing these rules outside the pipeline's target schema and referencing the schema name as a pipeline parameter, the team can apply the same set of data quality checks to different tables within the pipeline. This approach ensures consistency in data quality validations and reduces redundancy in code by not having to replicate the same rules in each DLT notebook or file.

Databricks Documentation on Delta Live Tables: Delta Live Tables Guide

Question 5

Question Type: MultipleChoice

A Delta Lake table representing metadata about content from user has the following schema:

user_id LONG, post_text STRING, post_id STRING, longitude FLOAT, latitude FLOAT, post_time TIMESTAMP, date DATE

Based on the above schema, which column is a good candidate for partitioning the Delta Table?

Options:

A- Date

B- Post_id

C- User_id

D- Post_time

Answer:

A

Explanation:

Partitioning a Delta Lake table improves query performance by organizing data into partitions based on the values of a column. In the given schema, the date column is a good candidate for partitioning for several reasons:

Time-Based Queries: If queries frequently filter or group by date, partitioning by the date column can significantly improve performance by limiting the amount of data scanned.

Granularity: The date column likely has a granularity that leads to a reasonable number of partitions (not too many and not too few). This balance is important for optimizing both read and write performance.

Data Skew: Other columns like post_id or user_id might lead to uneven partition sizes (data skew), which can negatively impact performance.

Partitioning by post_time could also be considered, but typically date is preferred due to its more manageable granularity.

Delta Lake Documentation on Table Partitioning: Optimizing Layout with Partitioning

Question 6

Question Type: **MultipleChoice**

Spill occurs as a result of executing various wide transformations. However, diagnosing spill requires one to proactively look for key indicators.

Where in the Spark UI are two of the primary indicators that a partition is spilling to disk?

Options:

A- Stage's detail screen and Executor's files

B- Stage's detail screen and Query's detail screen

C- Driver's and Executor's log files

D- Executor's detail screen and Executor's log files

Answer:

B

Explanation:

In Apache Spark's UI, indicators of data spilling to disk during the execution of wide transformations can be found in the Stage's detail screen and the Query's detail screen. These screens provide detailed metrics about each stage of a Spark job, including information about memory usage and spill data. If a task is spilling data to disk, it indicates that the data being processed exceeds the available memory, causing Spark to spill data to disk to free up memory. This is an important performance metric as excessive spill can significantly slow down the processing.

Apache Spark Monitoring and Instrumentation: Spark Monitoring Guide

Spark UI Explained: Spark UI Documentation

Question 7

Question Type: MultipleChoice

Which statement describes the default execution mode for Databricks Auto Loader?

Options:

A- New files are identified by listing the input directory; new files are incrementally and idempotently loaded into the target Delta Lake table.

B- Cloud vendor-specific queue storage and notification services are configured to track newly arriving files; new files are incrementally and impotently into the target Delta Lake table.

C- Webhook trigger Databricks job to run anytime new data arrives in a source directory; new data automatically merged into target tables using rules inferred from the data.

D- New files are identified by listing the input directory; the target table is materialized by directory querying all valid files in the source directory.

Answer:

A

Explanation:

Databricks Auto Loader simplifies and automates the process of loading data into Delta Lake. The default execution mode of the Auto Loader identifies new files by listing the input directory. It incrementally and idempotently loads these new files into the target Delta Lake table. This approach ensures that files are not missed and are processed exactly once, avoiding data duplication. The other options describe different mechanisms or integrations that are not part of the default behavior of the Auto Loader.

Databricks Auto Loader Documentation: Auto Loader Guide

Delta Lake and Auto Loader: Delta Lake Integration

Question 8

Question Type: **MultipleChoice**

A data engineer is performing a join operating to combine values from a static userlookup table with a streaming DataFrame streamingDF.

Which code block attempts to perform an invalid stream-static join?

Options:

A- userLookup.join(streamingDF, ['userid'], how='inner')

B- streamingDF.join(userLookup, ['user_id'], how='outer')

C- streamingDF.join(userLookup, ['user_id''], how='left')

D- streamingDF.join(userLookup, ['userid'], how='inner')

E- userLookup.join(streamingDF, ['user_id'], how='right')

Answer:

E

Explanation:

In Spark Structured Streaming, certain types of joins between a static DataFrame and a streaming DataFrame are not supported. Specifically, a right outer join where the static DataFrame is on the left side and the streaming DataFrame is on the right side is not valid. This is because Spark Structured Streaming cannot handle scenarios where it has to wait for new rows to arrive in the streaming DataFrame to match rows in the static DataFrame. The other join types listed (inner, left, and full outer joins) are supported in streaming-static DataFrame joins.

Structured Streaming Programming Guide: Join Operations

Databricks Documentation on Stream-Static Joins: Databricks Stream-Static Joins