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

Shared by Yang on 24-05-2024

Question 1

Question Type: MultipleChoice

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

The silver_device_recordings table will be used downstream for highly selective joins on a number of fields, and will also be leveraged by the machine learning team to filter on a handful of relevant fields, in total, 15 fields have been identified that will often be used for filter and join logic.

The data engineer is trying to determine the best approach for dealing with these nested fields before declaring the table schema.

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

Options:

A- Because Delta Lake uses Parquet for data storage, Dremel encoding information for nesting can be directly referenced by the Delta transaction log.

B- Tungsten encoding used by Databricks is optimized for storing string data: newly-added native support for querying JSON strings means that string types are always most efficient.

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

D- By default Delta Lake collects statistics on the first 32 columns in a table; these statistics are leveraged for data skipping when executing selective queries.

Answer:

D

Explanation:

Delta Lake, built on top of Parquet, enhances query performance through data skipping, which is based on the statistics collected for each file in a table. For tables with a large number of columns, Delta Lake by default collects and stores statistics only for the first 32 columns. These statistics include min/max values and null counts, which are used to optimize query execution by skipping irrelevant data files. When dealing with highly nested JSON structures, understanding this behavior is crucial for schema design, especially when determining which fields should be flattened or prioritized in the table structure to leverage data skipping efficiently for performance optimization. Reference: Databricks documentation on Delta Lake optimization techniques, including data skipping and statistics collection (https://docs.databricks.com/delta/optimizations/index.html).

Question 2

Question Type: MultipleChoice

In order to prevent accidental commits to production data, a senior data engineer has instituted a policy that all development work will reference clones of Delta Lake tables. After testing both deep and shallow clone, development tables are created using shallow clone.

A few weeks after initial table creation, the cloned versions of several tables implemented as Type 1 Slowly Changing Dimension (SCD) stop working. The transaction logs for the source tables show that vacuum was run the day before.

Why are the cloned tables no longer working?

Options:

A- The data files compacted by vacuum are not tracked by the cloned metadata; running refresh on the cloned table will pull in recent changes.

B- Because Type 1 changes overwrite existing records, Delta Lake cannot guarantee data consistency for cloned tables.

C- The metadata created by the clone operation is referencing data files that were purged as invalid by the vacuum command

D- Running vacuum automatically invalidates any shallow clones of a table; deep clone should always be used when a cloned table will be repeatedly queried.

Answer:

C

Explanation:

In Delta Lake, a shallow clone creates a new table by copying the metadata of the source table without duplicating the data files. When the vacuum command is run on the source table, it removes old data files that are no longer needed to maintain the transactional log's integrity, potentially including files referenced by the shallow clone's metadata. If these files are purged, the shallow cloned tables will reference non-existent data files, causing them to stop working properly. This highlights the dependency of shallow clones on the source table's data files and the impact of data management operations like vacuum on these clones. Reference: Databricks documentation on Delta Lake, particularly the sections on cloning tables (shallow and deep cloning) and data retention with the vacuum command (https://docs.databricks.com/delta/index.html).

Question 3

Question Type: MultipleChoice

A Data engineer wants to run unit's tests using common Python testing frameworks on python functions defined across several Databricks notebooks currently used in production.

How can the data engineer run unit tests against function that work with data in production?

Options:

A- Run unit tests against non-production data that closely mirrors production

B- Define and unit test functions using Files in Repos

C- Define units test and functions within the same notebook

D- Define and import unit test functions from a separate Databricks notebook

Answer:

A

Explanation:

The best practice for running unit tests on functions that interact with data is to use a dataset that closely mirrors the production data. This approach allows data engineers to validate the logic of their functions without the risk of affecting the actual production data. It's important to have a representative sample of production data to catch edge cases and ensure the functions will work correctly when used in a production environment.

Databricks Documentation on Testing: Testing and Validation of Data and Notebooks

Question 4

Question Type: **MultipleChoice**

Which is a key benefit of an end-to-end test?

Options:

A- It closely simulates real world usage of your application.

B- It pinpoint errors in the building blocks of your application.

C- It provides testing coverage for all code paths and branches.

D- It makes it easier to automate your test suite

Answer:

A

Explanation:

End-to-end testing is a methodology used to test whether the flow of an application, from start to finish, behaves as expected. The key benefit of an end-to-end test is that it closely simulates real-world, user behavior, ensuring that the system as a whole operates correctly.

Software Testing: End-to-End Testing

Question 5

Question Type: **MultipleChoice**

A data engineer is testing a collection of mathematical functions, one of which calculates the area under a curve as described by another function.

Which kind of the test does the above line exemplify?

Options:

A- Integration

B- Unit

C- Manual

D- functional

Answer:

B

Explanation:

A unit test is designed to verify the correctness of a small, isolated piece of code, typically a single function. Testing a mathematical function that calculates the area under a curve is an example of a unit test because it is testing a specific, individual function to ensure it operates as expected.

Software Testing Fundamentals: Unit Testing

Question 6

Question Type: MultipleChoice

What is a method of installing a Python package scoped at the notebook level to all nodes in the currently active cluster?

Options:

A- Use &Pip install in a notebook cell

B- Run source env/bin/activate in a notebook setup script

C- Install libraries from PyPi using the cluster UI

D- Use &sh install in a notebook cell

Answer:

C

Explanation:

Installing a Python package scoped at the notebook level to all nodes in the currently active cluster in Databricks can be achieved by using the Libraries tab in the cluster UI. This interface allows you to install libraries across all nodes in the cluster. While the %pip command in a notebook cell would only affect the driver node, using the cluster UI ensures that the package is installed on all nodes.

Databricks Documentation on Libraries: Libraries

Question 7

Question Type: MultipleChoice

A data engineer needs to capture pipeline settings from an existing in the workspace, and use them to create and version a JSON file to create a new pipeline.

Which command should the data engineer enter in a web terminal configured with the Databricks CLI?

Options:

A- Use the get command to capture the settings for the existing pipeline; remove the pipeline_id and rename the pipeline; use this in a create command

B- Stop the existing pipeline; use the returned settings in a reset command

C- Use the alone command to create a copy of an existing pipeline; use the get JSON command to get the pipeline definition; save this to git

D- Use list pipelines to get the specs for all pipelines; get the pipeline spec from the return results parse and use this to create a pipeline

Answer:

A

Explanation:

The Databricks CLI provides a way to automate interactions with Databricks services. When dealing with pipelines, you can use the databricks pipelines get --pipeline-id command to capture the settings of an existing pipeline in JSON format. This JSON can then be modified by removing the pipeline_id to prevent conflicts and renaming the pipeline to create a new pipeline. The modified JSON file can then be used with the databricks pipelines create command to create a new pipeline with those settings.

Databricks Documentation on CLI for Pipelines: Databricks CLI - Pipelines