

Databricks

Exam Questions Databricks-Certified-Professional-Data-Engineer

Databricks Certified Data Engineer Professional Exam



NEW QUESTION 1

A Databricks job has been configured with 3 tasks, each of which is a Databricks notebook. Task A does not depend on other tasks. Tasks B and C run in parallel, with each having a serial dependency on Task A.

If task A fails during a scheduled run, which statement describes the results of this run?

- A. Because all tasks are managed as a dependency graph, no changes will be committed to the Lakehouse until all tasks have successfully been completed.
- B. Tasks B and C will attempt to run as configured; any changes made in task A will be rolled back due to task failure.
- C. Unless all tasks complete successfully, no changes will be committed to the Lakehouse; because task A failed, all commits will be rolled back automatically.
- D. Tasks B and C will be skipped; some logic expressed in task A may have been committed before task failure.
- E. Tasks B and C will be skipped; task A will not commit any changes because of stage failure.

Answer: D

Explanation:

When a Databricks job runs multiple tasks with dependencies, the tasks are executed in a dependency graph. If a task fails, the downstream tasks that depend on it are skipped and marked as Upstream failed. However, the failed task may have already committed some changes to the Lakehouse before the failure occurred, and those changes are not rolled back automatically. Therefore, the job run may result in a partial update of the Lakehouse. To avoid this, you can use the transactional writes feature of Delta Lake to ensure that the changes are only committed when the entire job run succeeds.

Alternatively, you can use the Run if condition to configure tasks to run even when some or all of their dependencies have failed, allowing your job to recover from failures and

continue running. References:

? transactional writes: <https://docs.databricks.com/delta/delta-intro.html#transactional-writes>

? Run if: <https://docs.databricks.com/en/workflows/jobs/conditional-tasks.html>

NEW QUESTION 2

Review the following error traceback:

Which statement describes the error being raised?

- A. The code executed was PvSoark but was executed in a Scala notebook.
- B. There is no column in the table named heartrateheartrateheartrate
- C. There is a type error because a column object cannot be multiplied.
- D. There is a type error because a DataFrame object cannot be multiplied.
- E. There is a syntax error because the heartrate column is not correctly identified as a column.

Answer: E

Explanation:

The error being raised is an AnalysisException, which is a type of exception that occurs when Spark SQL cannot analyze or execute a query due to some logical or semantic error¹. In this case, the error message indicates that the query cannot resolve the column name 'heartrateheartrateheartrate' given the input columns 'heartrate' and 'age'. This means that there is no column in the table named 'heartrateheartrateheartrate', and the query is invalid. A possible cause of this error is a typo or a copy-paste mistake in the query. To fix this error, the query should use a valid column name that exists in the table, such as 'heartrate'.

References: AnalysisException

NEW QUESTION 3

A junior data engineer has been asked to develop a streaming data pipeline with a grouped aggregation using DataFrame df. The pipeline needs to calculate the average humidity and average temperature for each non-overlapping five-minute interval. Events are recorded once per minute per device.

Streaming DataFrame df has the following schema:

"device_id INT, event_time TIMESTAMP, temp FLOAT, humidity FLOAT" Code block:

Choose the response that correctly fills in the blank within the code block to complete this task.

- A. `to_interval("event_time", "5 minutes").alias("time")`
- B. `window("event_time", "5 minutes").alias("time")`
- C. `"event_time"`
- D. `window("event_time", "10 minutes").alias("time")`
- E. `lag("event_time", "10 minutes").alias("time")`

Answer: B

Explanation:

This is the correct answer because the window function is used to group streaming data by time intervals. The window function takes two arguments: a time column and a window duration. The window duration specifies how long each window is, and must be a multiple of 1 second. In this case, the window duration is "5 minutes", which means each window will cover a non-overlapping five-minute interval. The window function also returns a struct column with two fields: start and end, which represent the start and end time of each window. The alias function is used to rename the struct column as "time". Verified References:

[Databricks Certified Data Engineer Professional], under "Structured Streaming" section; Databricks Documentation, under "WINDOW" section.

<https://www.databricks.com/blog/2017/05/08/event-time-aggregation-watermarking-apache-sparks-structured-streaming.html>

NEW QUESTION 4

A user new to Databricks is trying to troubleshoot long execution times for some pipeline logic they are working on. Presently, the user is executing code cell-by-cell, using display() calls to confirm code is producing the logically correct results as new transformations are added to an operation. To get a measure of average time to execute, the user is running each cell multiple times interactively.

Which of the following adjustments will get a more accurate measure of how code is likely to perform in production?

- A. Scala is the only language that can be accurately tested using interactive notebooks; because the best performance is achieved by using Scala code compiled to JAR
- B. all PySpark and Spark SQL logic should be refactored.
- C. The only way to meaningfully troubleshoot code execution times in development notebooks is to use production-sized data and production-sized clusters with Run All execution.
- D. Production code development should only be done using an IDE; executing code against a local build of open source Spark and Delta Lake will provide the

most accurate benchmarks for how code will perform in production.

E. Calling `display()` forces a job to trigger, while many transformations will only add to the logical query plan; because of caching, repeated execution of the same logic does not provide meaningful results.

F. The Jobs UI should be leveraged to occasionally run the notebook as a job and track execution time during incremental code development because Photon can only be enabled on clusters launched for scheduled jobs.

Answer: D

Explanation:

In Databricks notebooks, using the `display()` function triggers an action that forces Spark to execute the code and produce a result. However, Spark operations are generally divided into transformations and actions. Transformations create a new dataset from an existing one and are lazy, meaning they are not computed immediately but added to a logical plan. Actions, like `display()`, trigger the execution of this logical plan. Repeatedly running the same code cell can lead to misleading performance measurements due to caching. When a dataset is used multiple times, Spark's optimization mechanism caches it in memory, making subsequent executions faster. This behavior does not accurately represent the first-time execution performance in a production environment where data might not be cached yet.

To get a more realistic measure of performance, it is recommended to:

? Clear the cache or restart the cluster to avoid the effects of caching.

? Test the entire workflow end-to-end rather than cell-by-cell to understand the cumulative performance.

? Consider using a representative sample of the production data, ensuring it includes various cases the code will encounter in production.

References:

? Databricks Documentation on Performance Optimization: Databricks Performance Tuning

? Apache Spark Documentation: RDD Programming Guide - Understanding transformations and actions

NEW QUESTION 5

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 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 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.

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 writing code is the largest cost associated with data engineering workloads; as such, automating table 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.

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.

NEW QUESTION 6

A Delta Lake table in the Lakehouse named `customer_parsams` is used in churn prediction by the machine learning team. The table contains information about customers derived from a number of upstream sources. Currently, the data engineering team populates this table nightly by overwriting the table with the current valid values derived from upstream data sources.

Immediately after each update succeeds, the data engineer team would like to determine the difference between the new version and the previous of the table.

Given the current implementation, which method can be used?

A. Parse the Delta Lake transaction log to identify all newly written data files.

B. Execute `DESCRIBE HISTORY customer_churn_params` to obtain the full operation metrics for the update, including a log of all records that have been added or modified.

C. Execute a query to calculate the difference between the new version and the previous version using Delta Lake's built-in versioning and time travel functionality.

D. Parse the Spark event logs to identify those rows that were updated, inserted, or deleted.

Answer: C

Explanation:

Delta Lake provides built-in versioning and time travel capabilities, allowing users to query previous snapshots of a table. This feature is particularly useful for understanding changes between different versions of the table. In this scenario, where the table is overwritten nightly, you can use Delta Lake's time travel feature to execute a query comparing the latest version of the table (the current state) with its previous version. This approach effectively identifies the differences (such as new, updated, or deleted records) between the two versions. The other options do not provide a straightforward or efficient way to directly compare different versions of a Delta Lake table.

References:

? Delta Lake Documentation on Time Travel: Delta Time Travel

? Delta Lake Versioning: Delta Lake Versioning Guide

NEW QUESTION 7

The data engineer team is configuring environment for development testing, and production before beginning migration on a new data pipeline. The team requires extensive testing on both the code and data resulting from code execution, and the team want to develop and test against similar production data as possible. A junior data engineer suggests that production data can be mounted to the development testing environments, allowing pre production code to execute against production data. Because all users have Admin privileges in the development environment, the junior data engineer has offered to configure permissions and mount this data for the team. Which statement captures best practices for this situation?

- A. Because access to production data will always be verified using passthrough credentials it is safe to mount data to any Databricks development environment.
- B. All developer, testing and production code and data should exist in a single unified workspace; creating separate environments for testing and development further reduces risks.
- C. In environments where interactive code will be executed, production data should only be accessible with read permissions; creating isolated databases for each environment further reduces risks.
- D. Because delta Lake versions all data and supports time travel, it is not possible for user error or malicious actors to permanently delete production data, as such it is generally safe to mount production data anywhere.

Answer: C

Explanation:

The best practice in such scenarios is to ensure that production data is handled securely and with proper access controls. By granting only read access to production data in development and testing environments, it mitigates the risk of unintended data modification. Additionally, maintaining isolated databases for different environments helps to avoid accidental impacts on production data and systems. References:

? Databricks best practices for securing data:
<https://docs.databricks.com/security/index.html>

NEW QUESTION 8

Which of the following technologies can be used to identify key areas of text when parsing Spark Driver log4j output?

- A. Regex
- B. Julia
- C. pyspark.ml.feature
- D. Scala Datasets
- E. C++

Answer: A

Explanation:

Regex, or regular expressions, are a powerful way of matching patterns in text. They can be used to identify key areas of text when parsing Spark Driver log4j output, such as the log level, the timestamp, the thread name, the class name, the method name, and the message. Regex can be applied in various languages and frameworks, such as Scala, Python, Java, Spark SQL, and Databricks notebooks. References:

? <https://docs.databricks.com/notebooks/notebooks-use.html#use-regular-expressions>
 ? <https://docs.databricks.com/spark/latest/spark-sql/udf-scala.html#using-regular-expressions-in-udfs>
 ? https://docs.databricks.com/spark/latest/sparkr/functions/regexp_extract.html
 ? https://docs.databricks.com/spark/latest/sparkr/functions/regexp_replace.html

NEW QUESTION 9

A data engineer is configuring a pipeline that will potentially see late-arriving, duplicate records.

In addition to de-duplicating records within the batch, which of the following approaches allows the data engineer to deduplicate data against previously processed records as it is inserted into a Delta table?

- A. Set the configuration `delta.deduplicate = true`.
- B. VACUUM the Delta table after each batch completes.
- C. Perform an insert-only merge with a matching condition on a unique key.
- D. Perform a full outer join on a unique key and overwrite existing data.
- E. Rely on Delta Lake schema enforcement to prevent duplicate records.

Answer: C

Explanation:

To deduplicate data against previously processed records as it is inserted into a Delta table, you can use the merge operation with an insert-only clause. This allows you to insert new records that do not match any existing records based on a unique key, while ignoring duplicate records that match existing records. For example, you can use the following syntax:

```
MERGE INTO target_table USING source_table ON target_table.unique_key = source_table.unique_key WHEN NOT MATCHED THEN INSERT *
```

This will insert only the records from the source table that have a unique key that is not present in the target table, and skip the records that have a matching key. This way, you can avoid inserting duplicate records into the Delta table.

References:

? <https://docs.databricks.com/delta/delta-update.html#upsert-into-a-table-using-merge>
 ? <https://docs.databricks.com/delta/delta-update.html#insert-only-merge>

NEW QUESTION 10

A Databricks job has been configured with 3 tasks, each of which is a Databricks notebook. Task A does not depend on other tasks. Tasks B and C run in parallel, with each having a serial dependency on task A.

If tasks A and B complete successfully but task C fails during a scheduled run, which statement describes the resulting state?

- A. All logic expressed in the notebook associated with tasks A and B will have been successfully completed; some operations in task C may have completed successfully.
- B. All logic expressed in the notebook associated with tasks A and B will have been successfully completed; any changes made in task C will be rolled back due to task failure.
- C. All logic expressed in the notebook associated with task A will have been successfully completed; tasks B and C will not commit any changes because of stage failure.
- D. Because all tasks are managed as a dependency graph, no changes will be committed to the Lakehouse until all tasks have successfully been completed.

E. Unless all tasks complete successfully, no changes will be committed to the Lakehouse; because task C failed, all commits will be rolled back automatically.

Answer: A

Explanation:

The query uses the CREATE TABLE USING DELTA syntax to create a Delta Lake table from an existing Parquet file stored in DBFS. The query also uses the LOCATION keyword to specify the path to the Parquet file as /mnt/finance_eda_bucket/tx_sales.parquet. By using the LOCATION keyword, the query creates an external table, which is a table that is stored outside of the default warehouse directory and whose metadata is not managed by Databricks. An external table can be created from an existing directory in a cloud storage system, such as DBFS or S3, that contains data files in a supported format, such as Parquet or CSV. The resulting state after running the second command is that an external table will be created in the storage container mounted to /mnt/finance_eda_bucket with the new name prod.sales_by_store. The command will not change any data or move any files in the storage container; it will only update the table reference in the metastore and create a new Delta transaction log for the renamed table. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "ALTER TABLE RENAME TO" section; Databricks Documentation, under "Create an external table" section.

NEW QUESTION 10

Which statement characterizes the general programming model used by Spark Structured Streaming?

- A. Structured Streaming leverages the parallel processing of GPUs to achieve highly parallel data throughput.
- B. Structured Streaming is implemented as a messaging bus and is derived from Apache Kafka.
- C. Structured Streaming uses specialized hardware and I/O streams to achieve sub-second latency for data transfer.
- D. Structured Streaming models new data arriving in a data stream as new rows appended to an unbounded table.
- E. Structured Streaming relies on a distributed network of nodes that hold incremental state values for cached stages.

Answer: B

Explanation:

This is the correct answer because it characterizes the general programming model used by Spark Structured Streaming, which is to treat a live data stream as a table that is being continuously appended. This leads to a new stream processing model that is very similar to a batch processing model, where users can express their streaming computation using the same Dataset/DataFrame API as they would use for static data. The Spark SQL engine will take care of running the streaming query incrementally and continuously and updating the final result as streaming data continues to arrive. Verified References: [Databricks Certified Data Engineer Professional], under "Structured Streaming" section; Databricks Documentation, under "Overview" section.

NEW QUESTION 14

Which statement describes the correct use of pyspark.sql.functions.broadcast?

- A. It marks a column as having low enough cardinality to properly map distinct values to available partitions, allowing a broadcast join.
- B. It marks a column as small enough to store in memory on all executors, allowing a broadcast join.
- C. It caches a copy of the indicated table on attached storage volumes for all active clusters within a Databricks workspace.
- D. It marks a DataFrame as small enough to store in memory on all executors, allowing a broadcast join.
- E. It caches a copy of the indicated table on all nodes in the cluster for use in all future queries during the cluster lifetime.

Answer: D

Explanation:

<https://spark.apache.org/docs/3.1.3/api/python/reference/api/pyspark.sql.functions.broadcast.html>

The broadcast function in PySpark is used in the context of joins. When you mark a DataFrame with broadcast, Spark tries to send this DataFrame to all worker nodes so that it can be joined with another DataFrame without shuffling the larger DataFrame across the nodes. This is particularly beneficial when the DataFrame is small enough to fit into the memory of each node. It helps to optimize the join process by reducing the amount of data that needs to be shuffled across the cluster, which can be a very expensive operation in terms of computation and time.

The pyspark.sql.functions.broadcast function in PySpark is used to hint to Spark that a DataFrame is small enough to be broadcast to all worker nodes in the cluster. When this hint is applied, Spark can perform a broadcast join, where the smaller DataFrame is sent to each executor only once and joined with the larger DataFrame on each executor. This can significantly reduce the amount of data shuffled across the network and can improve the performance of the join operation. In a broadcast join, the entire smaller DataFrame is sent to each executor, not just a specific column or a cached version on attached storage. This function is particularly useful when one of the DataFrames in a join operation is much smaller than the other, and can fit comfortably in the memory of each executor node.

References:

? Databricks Documentation on Broadcast Joins: Databricks Broadcast Join Guide

? PySpark API Reference: pyspark.sql.functions.broadcast

NEW QUESTION 19

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?

- 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.

References:

? Software Testing Fundamentals: Unit Testing

NEW QUESTION 21

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 constraints and multi-table inserts to validate records on write.

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

- 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. References:

? Databricks Documentation on Delta Lake: Delta Lake Guide

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

NEW QUESTION 23

Which REST API call can be used to review the notebooks configured to run as tasks in a multi-task job?

- A. /jobs/runs/list
- B. /jobs/runs/get-output
- C. /jobs/runs/get
- D. /jobs/get
- E. /jobs/list

Answer: D

Explanation:

This is the correct answer because it is the REST API call that can be used to review the notebooks configured to run as tasks in a multi-task job. The REST API is an interface that allows programmatically interacting with Databricks resources, such as clusters, jobs, notebooks, or tables. The REST API uses HTTP methods, such as GET, POST, PUT, or DELETE, to perform operations on these resources. The /jobs/get endpoint is a GET method that returns information about a job given its job ID. The information includes the job settings, such as the name, schedule, timeout, retries, email notifications, and tasks. The tasks are the units of work that a job executes. A task can be a notebook task, which runs a notebook with specified parameters; a jar task, which runs a JAR uploaded to DBFS with specified main class and arguments; or a python task, which runs a Python file uploaded to DBFS with specified parameters. A multi-task job is a job that has more than one task configured to run in a specific order or in parallel. By using the /jobs/get endpoint, one can review the notebooks configured to run as tasks in a multi-task job.

Verified References: [Databricks Certified Data Engineer Professional], under "Databricks Jobs" section; Databricks Documentation, under "Get" section; Databricks Documentation, under "JobSettings" section.

NEW QUESTION 26

A Structured Streaming job deployed to production has been experiencing delays during peak hours of the day. At present, during normal execution, each microbatch of data is processed in less than 3 seconds. During peak hours of the day, execution time for each microbatch becomes very inconsistent, sometimes exceeding 30 seconds. The streaming write is currently configured with a trigger interval of 10 seconds.

Holding all other variables constant and assuming records need to be processed in less than 10 seconds, which adjustment will meet the requirement?

- A. Decrease the trigger interval to 5 seconds; triggering batches more frequently allows idle executors to begin processing the next batch while longer running tasks from previous batches finish.
- B. Increase the trigger interval to 30 seconds; setting the trigger interval near the maximum execution time observed for each batch is always best practice to ensure no records are dropped.
- C. The trigger interval cannot be modified without modifying the checkpoint directory; to maintain the current stream state, increase the number of shuffle partitions to maximize parallelism.
- D. Use the trigger once option and configure a Databricks job to execute the query every 10 seconds; this ensures all backlogged records are processed with each batch.
- E. Decrease the trigger interval to 5 seconds; triggering batches more frequently may prevent records from backing up and large batches from causing spill.

Answer: E

Explanation:

The adjustment that will meet the requirement of processing records in less than 10 seconds is to decrease the trigger interval to 5 seconds. This is because triggering batches more frequently may prevent records from backing up and large batches from causing spill. Spill is a phenomenon where the data in memory exceeds the available capacity and has to be written to disk, which can slow down the processing and increase the execution time¹. By reducing the trigger interval, the streaming query can process smaller batches of data more quickly and avoid spill. This can also improve the latency and throughput of the streaming job².

The other options are not correct, because:

? Option A is incorrect because triggering batches more frequently does not allow idle executors to begin processing the next batch while longer running tasks from previous batches finish. In fact, the opposite is true. Triggering batches more frequently may cause concurrent batches to compete for the same resources and cause contention and backpressure². This can degrade the performance and stability of the streaming job.

? Option B is incorrect because increasing the trigger interval to 30 seconds is not a good practice to ensure no records are dropped. Increasing the trigger interval means that the streaming query will process larger batches of data less frequently, which can increase the risk of spill, memory pressure, and timeouts¹². This can also increase the latency and reduce the throughput of the streaming job.

? Option C is incorrect because the trigger interval can be modified without modifying the checkpoint directory. The checkpoint directory stores the metadata and state of the streaming query, such as the offsets, schema, and configuration³. Changing the trigger interval does not affect the state of the streaming query, and does not require a new checkpoint directory. However, changing the number of shuffle partitions may affect the state of the streaming query, and may require a new checkpoint directory⁴.

? Option D is incorrect because using the trigger once option and configuring a Databricks job to execute the query every 10 seconds does not ensure that all backlogged records are processed with each batch. The trigger once option means that the streaming query will process all the available data in the source and then stop⁵. However, this does not guarantee that the query will finish processing within 10 seconds, especially if there are a lot of records in the source. Moreover, configuring a Databricks job to execute the query every 10 seconds may cause overlapping or missed batches, depending on the execution time of the

query.

References: Memory Management Overview, Structured Streaming Performance Tuning Guide, Checkpointing, Recovery Semantics after Changes in a Streaming Query, Triggers

NEW QUESTION 27

What statement is true regarding the retention of job run history?

- A. It is retained until you export or delete job run logs
- B. It is retained for 30 days, during which time you can deliver job run logs to DBFS or S3
- C. It is retained for 60 days, during which you can export notebook run results to HTML
- D. It is retained for 60 days, after which logs are archived
- E. It is retained for 90 days or until the run-id is re-used through custom run configuration

Answer: C

NEW QUESTION 30

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?

- 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. References: Databricks documentation on Delta Lake optimization techniques, including data skipping and statistics collection (<https://docs.databricks.com/delta/optimizations/index.html>).

NEW QUESTION 32

A junior member of the data engineering team is exploring the language interoperability of Databricks notebooks. The intended outcome of the below code is to register a view of all sales that occurred in countries on the continent of Africa that appear in the geo_lookup table.

Before executing the code, running SHOW TABLES on the current database indicates the database contains only two tables: geo_lookup and sales.

```
Cmd 1
%python
countries_af = [x[0] for x in
spark.table("geo_lookup").filter("continent='AF'").select("country").collect()]
```

```
Cmd 2
%sql
CREATE VIEW sales_af AS
SELECT *
FROM sales
WHERE city IN countries_af
AND CONTINENT = "AF"
```

Which statement correctly describes the outcome of executing these command cells in order in an interactive notebook?

- A. Both commands will succeed
- B. Executing show tables will show that countries at and sales at have been registered as views.
- C. Cmd 1 will succeed
- D. Cmd 2 will search all accessible databases for a table or view named countries af: if this entity exists, Cmd 2 will succeed.
- E. Cmd 1 will succeed and Cmd 2 will fail, countries at will be a Python variable representing a PySpark DataFrame.
- F. Both commands will fail
- G. No new variables, tables, or views will be created.
- H. Cmd 1 will succeed and Cmd 2 will fail, countries at will be a Python variable containing a list of strings.

Answer: E

Explanation:

This is the correct answer because Cmd 1 is written in Python and uses a list comprehension to extract the country names from the geo_lookup table and store them in a Python variable named countries af. This variable will contain a list of strings, not a PySpark DataFrame or a SQL view. Cmd 2 is written in SQL and tries to create a view named sales af by selecting from the sales table where city is in countries af. However, this command will fail because countries af is not a valid SQL entity and cannot be used in a SQL query. To fix this, a better approach would be to use spark.sql() to execute a SQL query in Python and pass the countries af variable as a parameter. Verified References: [Databricks Certified Data Engineer Professional], under "Language Interoperability" section; Databricks Documentation, under "Mix languages" section.

NEW QUESTION 37

To reduce storage and compute costs, the data engineering team has been tasked with curating a series of aggregate tables leveraged by business intelligence dashboards, customer-facing applications, production machine learning models, and ad hoc analytical queries.

The data engineering team has been made aware of new requirements from a customer-facing application, which is the only downstream workload they manage entirely. As a result, an aggregate table used by numerous teams across the organization will need to have a number of fields renamed, and additional fields will also be added.

Which of the solutions addresses the situation while minimally interrupting other teams in the organization without increasing the number of tables that need to be managed?

- A. Send all users notice that the schema for the table will be changing; include in the communication the logic necessary to revert the new table schema to match historic queries.
- B. Configure a new table with all the requisite fields and new names and use this as the source for the customer-facing application; create a view that maintains the original data schema and table name by aliasing select fields from the new table.
- C. Create a new table with the required schema and new fields and use Delta Lake's deep clone functionality to sync up changes committed to one table to the corresponding table.
- D. Replace the current table definition with a logical view defined with the query logic currently writing the aggregate table; create a new table to power the customer-facing application.
- E. Add a table comment warning all users that the table schema and field names will be changing on a given date; overwrite the table in place to the specifications of the customer-facing application.

Answer: B

Explanation:

This is the correct answer because it addresses the situation while minimally interrupting other teams in the organization without increasing the number of tables that need to be managed. The situation is that an aggregate table used by numerous teams across the organization will need to have a number of fields renamed, and additional fields will also be added, due to new requirements from a customer-facing application. By configuring a new table with all the requisite fields and new names and using this as the source for the customer-facing application, the data engineering team can meet the new requirements without affecting other teams that rely on the existing table schema and name. By creating a view that maintains the original data schema and table name by aliasing select fields from the new table, the data engineering team can also avoid duplicating data or creating additional tables that need to be managed. Verified References: [Databricks Certified Data Engineer Professional], under "Lakehouse" section; Databricks Documentation, under "CREATE VIEW" section.

NEW QUESTION 42

When evaluating the Ganglia Metrics for a given cluster with 3 executor nodes, which indicator would signal proper utilization of the VM's resources?

- A. The five Minute Load Average remains consistent/flat
- B. Bytes Received never exceeds 80 million bytes per second
- C. Network I/O never spikes
- D. Total Disk Space remains constant
- E. CPU Utilization is around 75%

Answer: E

Explanation:

In the context of cluster performance and resource utilization, a CPU utilization rate of around 75% is generally considered a good indicator of efficient resource usage. This level of CPU utilization suggests that the cluster is being effectively used without being overburdened or underutilized.

? A consistent 75% CPU utilization indicates that the cluster's processing power is being effectively employed while leaving some headroom to handle spikes in workload or additional tasks without maxing out the CPU, which could lead to performance degradation.

? A five Minute Load Average that remains consistent/flat (Option A) might indicate underutilization or a bottleneck elsewhere.

? Monitoring network I/O (Options B and C) is important, but these metrics alone don't provide a complete picture of resource utilization efficiency.

? Total Disk Space (Option D) remaining constant is not necessarily an indicator of proper resource utilization, as it's more related to storage rather than computational efficiency.

References:

? Ganglia Monitoring System: Ganglia Documentation

? Databricks Documentation on Monitoring: Databricks Cluster Monitoring

NEW QUESTION 45

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

- 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.

References:

? Databricks Documentation on Libraries: Libraries

NEW QUESTION 48

The data science team has created and logged a production model using MLflow. The following code correctly imports and applies the production model to output the predictions as a new DataFrame named `preds` with the schema "customer_id LONG, predictions DOUBLE, date DATE".

```
from pyspark.sql.functions import current_date

model = mlflow.pyfunc.spark_udf(spark, model_uri="models:/churn/prod")
df = spark.table("customers")
columns = ["account_age", "time_since_last_seen", "app_rating"]
preds = (df.select(
    "customer_id",
    model(*columns).alias("predictions"),
    current_date().alias("date")
))
```

The data science team would like predictions saved to a Delta Lake table with the ability to compare all predictions across time. Churn predictions will be made at most once per day.

Which code block accomplishes this task while minimizing potential compute costs?

- A) `preds.write.mode("append").saveAsTable("churn_preds")`
- B) `preds.write.format("delta").save("/preds/churn_preds")`
- C)

```
(preds.writeStream
    .outputMode("overwrite")
    .option("checkpointPath", "/_checkpoints/churn_preds")
    .start("/preds/churn_preds")
)
```

D)

```
(preds.write
    .format("delta")
    .mode("overwrite")
    .saveAsTable("churn_preds")
)
```

E)

```
(preds.writeStream
    .outputMode("append")
    .option("checkpointPath", "/_checkpoints/churn_preds")
    .table("churn_preds")
)
```

- A. Option A
- B. Option B
- C. Option C
- D. Option D
- E. Option E

Answer: A

NEW QUESTION 51

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:

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.

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.

NEW QUESTION 52

The data science team has created and logged a production using MLFlow. The model accepts a list of column names and returns a new column of type DOUBLE. The following code correctly imports the production model, load the customer table containing the customer_id key column into a Dataframe, and defines the feature columns needed for the model.

```
model = mlflow.pyfunc.spark_udf (spark,
model_uri="models:/churn/prod")

df = spark.table("customers")

columns = ["account_age", "time_since_last_seen", "app_rating"]
```

Which code block will output DataFrame with the schema " customer_id LONG, predictions DOUBLE"?

- A. Model, predict (df, columns)
- B. Df, map (lambda k:midel (x [columns]) ,select ("customer_id predictions")
- C. D
- D. Select ("customer_id". Model ("columns) alias ("predictions")
- E. Df.apply(model, columns). Select ("customer_id, prediction"

Answer: A

Explanation:

Given the information that the model is registered with MLflow and assuming predict is the method used to apply the model to a set of columns, we use the model.predict() function to apply the model to the DataFrame df using the specified columns. The model.predict() function is designed to take in a DataFrame and a list of column names as arguments, applying the trained model to these features to produce a predictions column. When working with PySpark, this predictions column needs to be selected alongside the customer_id to create a new DataFrame with the schema customer_id LONG, predictions DOUBLE.

References:

? MLflow documentation on using Python function models: <https://www.mlflow.org/docs/latest/models.html#python-function-python>

? PySpark MLlib documentation on model prediction: <https://spark.apache.org/docs/latest/ml-pipeline.html#pipeline>

NEW QUESTION 56

When scheduling Structured Streaming jobs for production, which configuration automatically recovers from query failures and keeps costs low?

- A. Cluster: New Job Cluster; Retries: Unlimited;Maximum Concurrent Runs: Unlimited
- B. Cluster: New Job Cluster; Retries: None;Maximum Concurrent Runs: 1
- C. Cluster: Existing All-Purpose Cluster; Retries: Unlimited;Maximum Concurrent Runs: 1
- D. Cluster: Existing All-Purpose Cluster; Retries: Unlimited;Maximum Concurrent Runs: 1
- E. Cluster: Existing All-Purpose Cluster; Retries: None;Maximum Concurrent Runs: 1

Answer: D

Explanation:

The configuration that automatically recovers from query failures and keeps costs low is to use a new job cluster, set retries to unlimited, and set maximum concurrent runs to 1. This configuration has the following advantages:

? A new job cluster is a cluster that is created and terminated for each job run. This means that the cluster resources are only used when the job is running, and no idle costs are incurred. This also ensures that the cluster is always in a clean state and has the latest configuration and libraries for the job1.

? Setting retries to unlimited means that the job will automatically restart the query in case of any failure, such as network issues, node failures, or transient errors. This improves the reliability and availability of the streaming job, and avoids data loss or inconsistency2.

? Setting maximum concurrent runs to 1 means that only one instance of the job can run at a time. This prevents multiple queries from competing for the same resources or writing to the same output location, which can cause performance degradation or data corruption3.

Therefore, this configuration is the best practice for scheduling Structured Streaming jobs for production, as it ensures that the job is resilient, efficient, and consistent.

References: Job clusters, Job retries, Maximum concurrent runs

NEW QUESTION 57

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?

- 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.

NEW QUESTION 60

The downstream consumers of a Delta Lake table have been complaining about data quality issues impacting performance in their applications. Specifically, they have complained that invalid latitude and longitude values in the activity_details table have been breaking their ability to use other geolocation processes. A junior engineer has written the following code to add CHECK constraints to the Delta Lake table:

```
ALTER TABLE activity_details
ADD CONSTRAINT valid_coordinates
CHECK (
    latitude >= -90 AND
    latitude <= 90 AND
    longitude >= -180 AND
    longitude <= 180);
```

A senior engineer has confirmed the above logic is correct and the valid ranges for latitude and longitude are provided, but the code fails when executed. Which statement explains the cause of this failure?

- A. Because another team uses this table to support a frequently running application, two- phase locking is preventing the operation from committing.
- B. The activity details table already exists; CHECK constraints can only be added during initial table creation.
- C. The activity details table already contains records that violate the constraints; all existing data must pass CHECK constraints in order to add them to an existing table.
- D. The activity details table already contains records; CHECK constraints can only be added prior to inserting values into a table.
- E. The current table schema does not contain the field valid coordinates; schema evolution will need to be enabled before altering the table to add a constraint.

Answer: C

Explanation:

The failure is that the code to add CHECK constraints to the Delta Lake table fails when executed. The code uses ALTER TABLE ADD CONSTRAINT commands to add two CHECK constraints to a table named activity_details. The first constraint checks if the latitude value is between -90 and 90, and the second constraint checks if the longitude value is between -180 and 180. The cause of this failure is that the activity_details table already contains records that violate these constraints, meaning that they have invalid latitude or longitude values outside of these ranges. When adding CHECK constraints to an existing table, Delta Lake verifies that all existing data satisfies the constraints before adding them to the table. If any record violates the constraints, Delta Lake throws an exception and aborts the operation. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Add a CHECK constraint to an existing table" section. <https://docs.databricks.com/en/sql/language-manual/sql-ref-syntax-ddl-alter-table.html#add-constraint>

NEW QUESTION 63

A production workload incrementally applies updates from an external Change Data Capture feed to a Delta Lake table as an always-on Structured Stream job. When data was initially migrated for this table, OPTIMIZE was executed and most data files were resized to 1 GB. Auto Optimize and Auto Compaction were both turned on for the streaming production job. Recent review of data files shows that most data files are under 64 MB, although each partition in the table contains at least 1 GB of data and the total table size is over 10 TB.

Which of the following likely explains these smaller file sizes?

- A. Databricks has autotuned to a smaller target file size to reduce duration of MERGE operations
- B. Z-order indices calculated on the table are preventing file compaction
- C. Bloom filter indices calculated on the table are preventing file compaction
- D. Databricks has autotuned to a smaller target file size based on the overall size of data in the table
- E. Databricks has autotuned to a smaller target file size based on the amount of data in each partition

Answer: A

Explanation:

This is the correct answer because Databricks has a feature called Auto Optimize, which automatically optimizes the layout of Delta Lake tables by coalescing small files into larger ones and sorting data within each file by a specified column. However, Auto Optimize also considers the trade-off between file size and merge performance, and may choose a smaller target file size to reduce the duration of merge operations, especially for streaming workloads that frequently update existing records. Therefore, it is possible that Auto Optimize has autotuned to a smaller target file size based on the characteristics of the streaming production job. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Auto Optimize" section. <https://docs.databricks.com/en/delta/tune-file-size.html#autotune-table 'Autotune file size based on workload'>

NEW QUESTION 65

The data engineering team maintains a table of aggregate statistics through batch nightly updates. This includes total sales for the previous day alongside totals and averages for a variety of time periods including the 7 previous days, year-to-date, and quarter-to-date. This table is named store_sales_summary and the schema is as follows:

The table daily_store_sales contains all the information needed to update store_sales_summary. The schema for this table is: store_id INT, sales_date DATE, total_sales FLOAT If daily_store_sales is implemented as a Type 1 table and the total_sales column might be adjusted after manual data auditing, which approach is the safest to generate accurate reports in the store_sales_summary table?

- A. Implement the appropriate aggregate logic as a batch read against the daily_store_sales table and overwrite the store_sales_summary table with each Update.
- B. Implement the appropriate aggregate logic as a batch read against the daily_store_sales table and append new rows nightly to the store_sales_summary table.

- C. Implement the appropriate aggregate logic as a batch read against the daily_store_sales table and use upsert logic to update results in the store_sales_summary table.
- D. Implement the appropriate aggregate logic as a Structured Streaming read against the daily_store_sales table and use upsert logic to update results in the store_sales_summary table.
- E. Use Structured Streaming to subscribe to the change data feed for daily_store_sales and apply changes to the aggregates in the store_sales_summary table with each update.

Answer: E

Explanation:

The daily_store_sales table contains all the information needed to update store_sales_summary. The schema of the table is:

store_id INT, sales_date DATE, total_sales FLOAT

The daily_store_sales table is implemented as a Type 1 table, which means that old values are overwritten by new values and no history is maintained. The total_sales column might be adjusted after manual data auditing, which means that the data in the table may change over time.

The safest approach to generate accurate reports in the store_sales_summary table is to use Structured Streaming to subscribe to the change data feed for daily_store_sales and apply changes to the aggregates in the store_sales_summary table with each update. Structured Streaming is a scalable and fault-tolerant stream processing engine built on Spark SQL. Structured Streaming allows processing data streams as if they were tables or DataFrames, using familiar operations such as select, filter, groupBy, or join. Structured Streaming also supports output modes that specify how to write the results of a streaming query to a sink, such as append, update, or complete. Structured Streaming can handle both streaming and batch data sources in a unified manner.

The change data feed is a feature of Delta Lake that provides structured streaming sources that can subscribe to changes made to a Delta Lake table. The change data feed captures both data changes and schema changes as ordered events that can be processed by downstream applications or services. The change data feed can be configured with different options, such as starting from a specific version or timestamp, filtering by operation type or partition values, or excluding no-op changes.

By using Structured Streaming to subscribe to the change data feed for daily_store_sales, one can capture and process any changes made to the total_sales column due to manual data auditing. By applying these changes to the aggregates in the store_sales_summary table with each update, one can ensure that the reports are always consistent and accurate with the latest data. Verified References: [Databricks Certified Data Engineer Professional], under "Spark Core" section; Databricks Documentation, under "Structured Streaming" section; Databricks Documentation, under "Delta Change Data Feed" section.

NEW QUESTION 67

A data team's Structured Streaming job is configured to calculate running aggregates for item sales to update a downstream marketing dashboard. The marketing team has introduced a new field to track the number of times this promotion code is used for each item. A junior data engineer suggests updating the existing query as follows: Note that proposed changes are in bold.

Original query:

```
df.groupBy("item")
  .agg(count("item").alias("total_count"),
       mean("sale_price").alias("avg_price"))
  .writeStream
  .outputMode("complete")
  .option("checkpointLocation", "/item_agg/__checkpoint")
  .start("/item_agg")
```

Proposed query:

```
df.groupBy("item")
  .agg(count("item").alias("total_count"),
       mean("sale_price").alias("avg_price"),
       count("promo_code = 'NEW_MEMBER')".alias("new_member_promo"))
  .writeStream
  .outputMode("complete")
  .option('mergeSchema', 'true')
  .option("checkpointLocation", "/item_agg/__checkpoint")
  .start("/item_agg")
```

Which step must also be completed to put the proposed query into production?

- A. Increase the shuffle partitions to account for additional aggregates
- B. Specify a new checkpointLocation
- C. Run REFRESH TABLE delta, /item_agg'
- D. Remove .option ('mergeSchema', 'true') from the streaming write

Answer: B

Explanation:

When introducing a new aggregation or a change in the logic of a Structured Streaming query, it is generally necessary to specify a new checkpoint location. This is because the checkpoint directory contains metadata about the offsets and the state of the aggregations of a streaming query. If the logic of the query changes, such as including a new aggregation field, the state information saved in the current checkpoint would not be compatible with the new logic, potentially leading to incorrect results or failures. Therefore, to accommodate the new field and ensure the streaming job has the correct starting point and state information for aggregations, a new checkpoint location should be specified. References:

? Databricks documentation on Structured Streaming:

<https://docs.databricks.com/spark/latest/structured-streaming/index.html>

? Databricks documentation on streaming checkpoints: <https://docs.databricks.com/spark/latest/structured-streaming/production.html#checkpointing>

NEW QUESTION 71

The data engineering team maintains the following code:

```
import pyspark.sql.functions as F

(spark.table("silver_customer_sales")
 .groupBy("customer_id")
 .agg(
   F.min("sale_date").alias("first_transaction_date"),
   F.max("sale_date").alias("last_transaction_date"),
   F.mean("sale_total").alias("average_sales"),
   F.countDistinct("order_id").alias("total_orders"),
   F.sum("sale_total").alias("lifetime_value")
 ).write
 .mode("overwrite")
 .table("gold_customer_lifetime_sales_summary")
)
```

Assuming that this code produces logically correct results and the data in the source table has been de-duplicated and validated, which statement describes what will occur when this code is executed?

- A. The silver_customer_sales table will be overwritten by aggregated values calculated from all records in the gold_customer_lifetime_sales_summary table as a batch job.
- B. A batch job will update the gold_customer_lifetime_sales_summary table, replacing only those rows that have different values than the current version of the table, using customer_id as the primary key.
- C. The gold_customer_lifetime_sales_summary table will be overwritten by aggregated values calculated from all records in the silver_customer_sales table as a batch job.
- D. An incremental job will leverage running information in the state store to update aggregate values in the gold_customer_lifetime_sales_summary table.
- E. An incremental job will detect if new rows have been written to the silver_customer_sales table; if new rows are detected, all aggregates will be recalculated and used to overwrite the gold_customer_lifetime_sales_summary table.

Answer: C

Explanation:

This code is using the pyspark.sql.functions library to group the silver_customer_sales table by customer_id and then aggregate the data using the minimum sale date, maximum sale total, and sum of distinct order ids. The resulting aggregated data is then written to the gold_customer_lifetime_sales_summary table, overwriting any existing data in that table. This is a batch job that does not use any incremental or streaming logic, and does not perform any merge or update operations. Therefore, the code will overwrite the gold table with the aggregated values from the silver table every time it is executed. References:

- ? <https://docs.databricks.com/spark/latest/dataframes-datasets/introduction-to-dataframes-python.html>
- ? <https://docs.databricks.com/spark/latest/dataframes-datasets/transforming-data-with-dataframes.html>
- ? <https://docs.databricks.com/spark/latest/dataframes-datasets/aggregating-data-with-dataframes.html>

NEW QUESTION 75

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?

- 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. References:

- ? [Databricks Documentation on Delta Live Tables: Delta Live Tables Guide](#)

NEW QUESTION 77

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?

- 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.

References:

- ? [Apache Spark Monitoring and Instrumentation: Spark Monitoring Guide](#)

? Spark UI Explained: Spark UI Documentation

NEW QUESTION 79

A Spark job is taking longer than expected. Using the Spark UI, a data engineer notes that the Min, Median, and Max Durations for tasks in a particular stage show the minimum and median time to complete a task as roughly the same, but the max duration for a task to be roughly 100 times as long as the minimum. Which situation is causing increased duration of the overall job?

- A. Task queueing resulting from improper thread pool assignment.
- B. Spill resulting from attached volume storage being too small.
- C. Network latency due to some cluster nodes being in different regions from the source data
- D. Skew caused by more data being assigned to a subset of spark-partitions.
- E. Credential validation errors while pulling data from an external system.

Answer: D

Explanation:

This is the correct answer because skew is a common situation that causes increased duration of the overall job. Skew occurs when some partitions have more data than others, resulting in uneven distribution of work among tasks and executors. Skew can be caused by various factors, such as skewed data distribution, improper partitioning strategy, or join operations with skewed keys. Skew can lead to performance issues such as long-running tasks, wasted resources, or even task failures due to memory or disk spills. Verified References: [Databricks Certified Data Engineer Professional], under "Performance Tuning" section; Databricks Documentation, under "Skew" section.

NEW QUESTION 84

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?

- 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.

NEW QUESTION 87

Assuming that the Databricks CLI has been installed and configured correctly, which Databricks CLI command can be used to upload a custom Python Wheel to object storage mounted with the DBFS for use with a production job?

- A. `configure`
- B. `fs`
- C. `jobs`
- D. `libraries`
- E. `workspace`

Answer: B

Explanation:

The `libraries` command group allows you to install, uninstall, and list libraries on Databricks clusters. You can use the `libraries install` command to install a custom Python Wheel on a cluster by specifying the `--whl` option and the path to the wheel file. For example, you can use the following command to install a custom Python Wheel named `mylib-0.1-py3-none-any.whl` on a cluster with the id `1234-567890-abcde123`:

```
databricks libraries install --cluster-id 1234-567890-abcde123 --whl dbfs:/mnt/mylib/mylib-0.1-py3-none-any.whl
```

This will upload the custom Python Wheel to the cluster and make it available for use with a production job. You can also use the `libraries uninstall` command to uninstall a library from a cluster, and the `libraries list` command to list the libraries installed on a cluster. References:

? Libraries CLI (legacy): <https://docs.databricks.com/en/archive/dev-tools/cli/libraries-cli.html>

? Library operations: <https://docs.databricks.com/en/dev-tools/cli/commands.html#library-operations>

? Install or update the Databricks CLI: <https://docs.databricks.com/en/dev-tools/cli/install.html>

NEW QUESTION 88

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 approved for the sales org.

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

- 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 job 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.

References:

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

NEW QUESTION 90

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?

- 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. References: Databricks documentation on access control and permissions for notebooks within the workspace (<https://docs.databricks.com/security/access-control/workspace-acl.html>).

NEW QUESTION 93

A table named user_ltv is being used to create a view that will be used by data analysts 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 email_ltv AS
SELECT
CASE WHEN
  is_member('marketing') THEN email
  ELSE 'REDACTED'
END AS email,
ltv
FROM user_ltv
```

An analyst who is not a member of the marketing group executes the following query: `SELECT * FROM email_ltv`

Which statement describes the results returned by this query?

- A. Three columns will be returned, but one column will be named "redacted" and contain only null values.
- B. Only the email and ltv columns will be returned; the email column will contain all null values.
- C. The email and ltv columns will be returned with the values in user ltv.
- D. The email, ag
- E. and ltv columns will be returned with the values in user ltv.
- F. Only the email and ltv columns will be returned; the email column will contain the string "REDACTED" in each row.

Answer: E

Explanation:

The code creates a view called email_ltv that selects the email and ltv columns from a table called user_ltv, which has the following schema: email STRING, age INT, ltv INT. The code also uses the CASE WHEN expression to replace the email values with the string "REDACTED" if the user is not a member of the marketing group. The user who executes the query is not a member of the marketing group, so they will only see the email and ltv columns, and the email column will contain the string "REDACTED" in each row. Verified References: [Databricks Certified Data Engineer Professional], under "Lakehouse" section; Databricks Documentation, under "CASE expression" section.

NEW QUESTION 94

Which statement regarding spark configuration on the Databricks platform is true?

- A. Spark configuration properties set for an interactive cluster with the Clusters UI will impact all notebooks attached to that cluster.
- B. When the same spar configuration property is set for an interactive to the same interactive cluster.
- C. Spark configuration set within a notebook will affect all SparkSession attached to the same interactive cluster
- D. The Databricks REST API can be used to modify the Spark configuration properties for an interactive cluster without interrupting jobs.

Answer: A

Explanation:

When Spark configuration properties are set for an interactive cluster using the Clusters UI in Databricks, those configurations are applied at the cluster level. This means that all notebooks attached to that cluster will inherit and be affected by these configurations. This approach ensures consistency across all executions within that cluster, as the Spark configuration properties dictate aspects such as memory allocation, number of executors, and other vital execution parameters. This centralized configuration management helps maintain standardized execution environments across different notebooks, aiding in debugging and performance optimization.

References:

? Databricks documentation on configuring clusters: <https://docs.databricks.com/clusters/configure.html>

NEW QUESTION 98

The following code has been migrated to a Databricks notebook from a legacy workload:

```
%sh
git clone https://github.com/foo/data_loader;
python ./data_loader/run.py;
mv ./output /dbfs/mnt/new_data
```

The code executes successfully and provides the logically correct results, however, it takes over 20 minutes to extract and load around 1 GB of data. Which statement is a possible explanation for this behavior?

- A. %sh triggers a cluster restart to collect and install Gi
- B. Most of the latency is related to cluster startup time.
- C. Instead of cloning, the code should use %sh pip install so that the Python code can get executed in parallel across all nodes in a cluster.
- D. %sh does not distribute file moving operations; the final line of code should be updated to use %fs instead.
- E. Python will always execute slower than Scala on Databrick
- F. The run.py script should be refactored to Scala.
- G. %sh executes shell code on the driver nod
- H. The code does not take advantage of the worker nodes or Databricks optimized Spark.

Answer: E

Explanation:

<https://www.databricks.com/blog/2020/08/31/introducing-the-databricks-web-terminal.html>

The code is using %sh to execute shell code on the driver node. This means that the code is not taking advantage of the worker nodes or Databricks optimized Spark. This is why the code is taking longer to execute. A better approach would be to use Databricks libraries and APIs to read and write data from Git and DBFS, and to leverage the parallelism and performance of Spark. For example, you can use the Databricks Connect feature to run your Python code on a remote Databricks cluster, or you can use the Spark Git Connector to read data from Git repositories as Spark DataFrames.

NEW QUESTION 101

The data science team has requested assistance in accelerating queries on free form text from user reviews. The data is currently stored in Parquet with the below schema:

```
item_id INT, user_id INT, review_id INT, rating FLOAT, review STRING
```

The review column contains the full text of the review left by the user. Specifically, the data science team is looking to identify if any of 30 key words exist in this field.

A junior data engineer suggests converting this data to Delta Lake will improve query performance.

Which response to the junior data engineer s suggestion is correct?

- A. Delta Lake statistics are not optimized for free text fields with high cardinality.
- B. Text data cannot be stored with Delta Lake.
- C. ZORDER ON review will need to be run to see performance gains.
- D. The Delta log creates a term matrix for free text fields to support selective filtering.
- E. Delta Lake statistics are only collected on the first 4 columns in a table.

Answer: A

Explanation:

Converting the data to Delta Lake may not improve query performance on free text fields with high cardinality, such as the review column. This is because Delta Lake collects statistics on the minimum and maximum values of each column, which are not very useful for filtering or skipping data on free text fields. Moreover, Delta Lake collects statistics on the first 32 columns by default, which may not include the review column if the table has more columns. Therefore, the junior data engineer's suggestion is not correct. A better approach would be to use a full-text search engine, such as Elasticsearch, to index and query the review column. Alternatively, you can use natural language processing techniques, such as tokenization, stemming, and lemmatization, to preprocess the review column and create a new column with normalized terms that can be used for filtering or skipping data. References:

? Optimizations: <https://docs.delta.io/latest/optimizations-oss.html>

? Full-text search with Elasticsearch: <https://docs.databricks.com/data/data-sources/elasticsearch.html>

? Natural language processing: <https://docs.databricks.com/applications/nlp/index.html>

NEW QUESTION 104

A production cluster has 3 executor nodes and uses the same virtual machine type for the driver and executor.

When evaluating the Ganglia Metrics for this cluster, which indicator would signal a bottleneck caused by code executing on the driver?

- A. The five Minute Load Average remains consistent/flat
- B. Bytes Received never exceeds 80 million bytes per second
- C. Total Disk Space remains constant
- D. Network I/O never spikes
- E. Overall cluster CPU utilization is around 25%

Answer: E

Explanation:

This is the correct answer because it indicates a bottleneck caused by code executing on the driver. A bottleneck is a situation where the performance or capacity of a system is limited by a single component or resource. A bottleneck can cause slow execution, high latency, or low throughput. A production cluster has 3 executor nodes and uses the same virtual machine type for the driver and executor. When evaluating the Ganglia Metrics for this cluster, one can look for indicators that show how the cluster resources are being utilized, such as CPU, memory, disk, or network. If the overall cluster CPU utilization is around 25%, it means that only one out of the four nodes (driver + 3 executors) is using its full CPU capacity, while the other three nodes are idle or underutilized. This suggests that the code executing on the driver is taking too long or consuming too much CPU resources, preventing the executors from receiving tasks or data to process. This can happen when the code has driver-side operations that are not parallelized or distributed, such as collecting large amounts of data to the driver, performing complex calculations on the driver, or using non-Spark libraries on the driver. Verified References: [Databricks Certified Data Engineer Professional], under "Spark Core" section; Databricks Documentation, under "View cluster status and event logs - Ganglia metrics" section; Databricks Documentation, under "Avoid

collecting large RDDs" section.

In a Spark cluster, the driver node is responsible for managing the execution of the Spark application, including scheduling tasks, managing the execution plan, and interacting with the cluster manager. If the overall cluster CPU utilization is low (e.g., around 25%), it may indicate that the driver node is not utilizing the available resources effectively and might be a bottleneck.

NEW QUESTION 109

A table is registered with the following code:

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.

Answer: B

NEW QUESTION 113

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?

- A. Can Manage
- B. Can Edit
- C. No permissions
- D. Can Read
- E. Can Run

Answer: C

Explanation:

This is the correct answer because it is the maximum notebook permissions that can be granted to the user without allowing accidental changes to production code or data. Notebook permissions are used to control access to notebooks in Databricks workspaces. There are four types of notebook permissions: Can Manage, Can Edit, Can Run, and Can Read. Can Manage allows full control over the notebook, including editing, running, deleting, exporting, and changing permissions. Can Edit allows modifying and running the notebook, but not changing permissions or deleting it. Can Run allows executing commands in an existing cluster attached to the notebook, but not modifying or exporting it. Can Read allows viewing the notebook content, but not running or modifying it. In this case, granting Can Read permission to the user will allow them to review the

production logic in the notebook without allowing them to make any changes to it or run any commands that may affect production data. Verified References: [Databricks Certified Data Engineer Professional], under "Databricks Workspace" section; Databricks Documentation, under "Notebook permissions" section.

NEW QUESTION 117

The security team is exploring whether or not the Databricks secrets module can be leveraged for connecting to an external database.

After testing the code with all Python variables being defined with strings, they upload the password to the secrets module and configure the correct permissions for the currently active user. They then modify their code to the following (leaving all other variables unchanged).

```
password = dbutils.secrets.get(scope="db_creds", key="jdbc_password")

print(password)

df = (spark
      .read
      .format("jdbc")
      .option("url", connection)
      .option("dbtable", tablename)
      .option("user", username)
      .option("password", password)
      )
```

Which statement describes what will happen when the above code is executed?

- A. The connection to the external table will fail; the string "redacted" will be printed.
- B. An interactive input box will appear in the notebook; if the right password is provided, the connection will succeed and the encoded password will be saved to DBFS.
- C. An interactive input box will appear in the notebook; if the right password is provided, the connection will succeed and the password will be printed in plain text.
- D. The connection to the external table will succeed; the string value of password will be printed in plain text.
- E. The connection to the external table will succeed; the string "redacted" will be printed.

Answer: E

Explanation:

This is the correct answer because the code is using the dbutils.secrets.get method to retrieve the password from the secrets module and store it in a variable. The secrets module allows users to securely store and access sensitive information such as passwords, tokens, or API keys. The connection to the external table will succeed because the password variable will contain the actual password value. However, when printing the password variable, the string "redacted" will be displayed instead of the plain text password, as a security measure to prevent exposing sensitive information in notebooks. Verified References: [Databricks Certified Data Engineer Professional], under "Security & Governance" section; Databricks Documentation, under "Secrets" section.

NEW QUESTION 122

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?

- 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.

References:

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

NEW QUESTION 124

The data engineering team has configured a job to process customer requests to be forgotten (have their data deleted). All user data that needs to be deleted is stored in Delta Lake tables using default table settings.

The team has decided to process all deletions from the previous week as a batch job at 1am each Sunday. The total duration of this job is less than one hour.

Every Monday at 3am, a batch job executes a series of VACUUM commands on all Delta Lake tables throughout the organization.

The compliance officer has recently learned about Delta Lake's time travel functionality. They are concerned that this might allow continued access to deleted data.

Assuming all delete logic is correctly implemented, which statement correctly addresses this concern?

- A. Because the vacuum command permanently deletes all files containing deleted records, deleted records may be accessible with time travel for around 24 hours.
- B. Because the default data retention threshold is 24 hours, data files containing deleted records will be retained until the vacuum job is run the following day.
- C. Because Delta Lake time travel provides full access to the entire history of a table, deleted records can always be recreated by users with full admin privileges.
- D. Because Delta Lake's delete statements have ACID guarantees, deleted records will be permanently purged from all storage systems as soon as a delete job completes.
- E. Because the default data retention threshold is 7 days, data files containing deleted records will be retained until the vacuum job is run 8 days later.

Answer: E

Explanation:

<https://learn.microsoft.com/en-us/azure/databricks/delta/vacuum>

NEW QUESTION 128

The view updates represents an incremental batch of all newly ingested data to be inserted or updated in the customers table.

The following logic is used to process these records.

Which statement describes this implementation?

- A. The customers table is implemented as a Type 3 table; old values are maintained as a new column alongside the current value.
- B. The customers table is implemented as a Type 2 table; old values are maintained but marked as no longer current and new values are inserted.
- C. The customers table is implemented as a Type 0 table; all writes are append only with no changes to existing values.
- D. The customers table is implemented as a Type 1 table; old values are overwritten by new values and no history is maintained.
- E. The customers table is implemented as a Type 2 table; old values are overwritten and new customers are appended.

Answer: A

Explanation:

The logic uses the MERGE INTO command to merge new records from the view updates into the table customers. The MERGE INTO command takes two arguments: a target table and a source table or view. The command also specifies a condition to match records between the target and the source, and a set of actions to perform when there is a match or not. In this case, the condition is to match records by customer_id, which is the primary key of the customers table.

The actions are to update the existing record in the target with the new values from the source, and set the current_flag to false to indicate that the record is no longer current; and to insert a new record in the target with the new values from the source, and set the current_flag to true to indicate that the record is current.

This means that old values are maintained but marked as no longer current and new values are inserted, which is the definition of a Type 2 table. Verified

References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Merge Into (Delta Lake on Databricks)" section.

NEW QUESTION 132

A Databricks SQL dashboard has been configured to monitor the total number of records present in a collection of Delta Lake tables using the following query pattern:

SELECT COUNT (*) FROM table -

Which of the following describes how results are generated each time the dashboard is updated?

- A. The total count of rows is calculated by scanning all data files
- B. The total count of rows will be returned from cached results unless REFRESH is run
- C. The total count of records is calculated from the Delta transaction logs
- D. The total count of records is calculated from the parquet file metadata
- E. The total count of records is calculated from the Hive metastore

Answer: C

Explanation:

<https://delta.io/blog/2023-04-19-faster-aggregations-metadata/#:~:text=You%20can%20get%20the%20number,a%20given%20Delta%20table%20version.>

NEW QUESTION 137

A junior data engineer on your team has implemented the following code block.

```
MERGE INTO events
USING new_events
ON events.event_id = new_events.event_id
WHEN NOT MATCHED
  INSERT *
```

The view new_events contains a batch of records with the same schema as the events Delta table. The event_id field serves as a unique key for this table. When this query is executed, what will happen with new records that have the same event_id as an existing record?

- A. They are merged.
- B. They are ignored.
- C. They are updated.
- D. They are inserted.
- E. They are deleted.

Answer: B

Explanation:

This is the correct answer because it describes what will happen with new records that have the same event_id as an existing record when the query is executed. The query uses the INSERT INTO command to append new records from the view new_events to the table events. However, the INSERT INTO command does not check for duplicate values in the primary key column (event_id) and does not perform any update or delete operations on existing records. Therefore, if there are new records that have the same event_id as an existing record, they will be ignored and not inserted into the table events. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Append data using INSERT INTO" section.

"If none of the WHEN MATCHED conditions evaluate to true for a source and target row pair that matches the merge_condition, then the target row is left unchanged." https://docs.databricks.com/en/sql/language-manual/delta-merge-into.html#:~:text=If%20none%20of%20the%20WHEN%20MATCHED%20conditions%20evaluate%20to%20true%20for%20a%20source%20and%20target%20row%20pair%20that%20matches%20the%20merge_condition%2C%20then%20the%20target%20row%20is%20left%20unchanged.

NEW QUESTION 138

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

- 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.

References:

? Software Testing: End-to-End Testing

NEW QUESTION 142

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?

- 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 succeeded 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.

NEW QUESTION 146

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
- E. by noting the column statistics
- F. In the Query Detail screen, by interpreting the Physical Plan

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.

NEW QUESTION 149

A small company based in the United States has recently contracted a consulting firm in India to implement several new data engineering pipelines to power artificial intelligence applications. All the company's data is stored in regional cloud storage in the United States. The workspace administrator at the company is uncertain about where the Databricks workspace used by the contractors should be deployed. Assuming that all data governance considerations are accounted for, which statement accurately informs this decision?

- A. Databricks runs HDFS on cloud volume storage; as such, cloud virtual machines must be deployed in the region where the data is stored.
- B. Databricks workspaces do not rely on any regional infrastructure; as such, the decision should be made based upon what is most convenient for the workspace administrator.
- C. Cross-region reads and writes can incur significant costs and latency; whenever possible, compute should be deployed in the same region the data is stored.
- D. Databricks leverages user workstations as the driver during interactive development; as such, users should always use a workspace deployed in a region they are physically near.
- E. Databricks notebooks send all executable code from the user's browser to virtual machines over the open internet; whenever possible, choosing a workspace region near the end users is the most secure.

Answer: C

Explanation:

This is the correct answer because it accurately informs this decision. The decision is about where the Databricks workspace used by the contractors should be deployed. The contractors are based in India, while all the company's data is stored in regional cloud storage in the United States. When choosing a region for deploying a Databricks workspace, one of the important factors to consider is the proximity to the data sources and sinks. Cross-region reads and writes can incur significant costs and latency due to network bandwidth and data transfer fees. Therefore, whenever possible, compute should be deployed in the same region the data is stored to optimize performance and reduce costs. Verified References: [Databricks Certified Data Engineer Professional], under "Databricks Workspace" section; Databricks Documentation, under "Choose a region" section.

NEW QUESTION 153

The view updates represents an incremental batch of all newly ingested data to be inserted or updated in the customers table. The following logic is used to process these records.

```

MERGE INTO customers USING (
SELECT updates.customer_id as merge_key, updates.* FROM updates
UNION ALL
SELECT NULL as merge_key, updates.* FROM updates JOIN customers
ON updates.customer_id = customers.customer_id
WHERE customers.current = true AND updates.address <> customers.address
) staged_updates
ON customers.customer_id = mergekey
WHEN MATCHED AND customers.current = true AND customers.address <> staged_updates.address THEN
UPDATE SET current = false, end_date = staged_updates.effective_date WHEN NOT MATCHED THEN
INSERT (customer_id, address, current, effective_date, end_date)
VALUES (staged_updates.customer_id, staged_updates.address, true, staged_updates.effective_date, null)

```

Which statement describes this implementation?

- A. The customers table is implemented as a Type 2 table; old values are overwritten and new customers are appended.
- B. The customers table is implemented as a Type 1 table; old values are overwritten by new values and no history is maintained.

- C. The customers table is implemented as a Type 2 table; old values are maintained but marked as no longer current and new values are inserted.
- D. The customers table is implemented as a Type 0 table; all writes are append only with no changes to existing values.

Answer: C

Explanation:

The provided MERGE statement is a classic implementation of a Type 2 SCD in a data warehousing context. In this approach, historical data is preserved by keeping old records (marking them as not current) and adding new records for changes. Specifically, when a match is found and there's a change in the address, the existing record in the customers table is updated to mark it as no longer current (current = false), and an end date is assigned (end_date = staged_updates.effective_date). A new record for the customer is then inserted with the updated information, marked as current. This method ensures that the full history of changes to customer information is maintained in the table, allowing for time-based analysis of customer data. References: Databricks documentation on implementing SCDs using Delta Lake and the MERGE statement (<https://docs.databricks.com/delta/delta-update.html#upsert-into-a-table-using-merge>).

NEW QUESTION 156

All records from an Apache Kafka producer are being ingested into a single Delta Lake table with the following schema:
key BINARY, value BINARY, topic STRING, partition LONG, offset LONG, timestamp LONG

There are 5 unique topics being ingested. Only the "registration" topic contains Personal Identifiable Information (PII). The company wishes to restrict access to PII. The company also wishes to only retain records containing PII in this table for 14 days after initial ingestion. However, for non-PII information, it would like to retain these records indefinitely.

Which of the following solutions meets the requirements?

- A. All data should be deleted biweekly; Delta Lake's time travel functionality should be leveraged to maintain a history of non-PII information.
- B. Data should be partitioned by the registration field, allowing ACLs and delete statements to be set for the PII directory.
- C. Because the value field is stored as binary data, this information is not considered PII and no special precautions should be taken.
- D. Separate object storage containers should be specified based on the partition field, allowing isolation at the storage level.
- E. Data should be partitioned by the topic field, allowing ACLs and delete statements to leverage partition boundaries.

Answer: B

Explanation:

Partitioning the data by the topic field allows the company to apply different access control policies and retention policies for different topics. For example, the company can use the Table Access Control feature to grant or revoke permissions to the registration topic based on user roles or groups. The company can also use the DELETE command to remove records from the registration topic that are older than 14 days, while keeping the records from other topics indefinitely.

Partitioning by the topic field also improves the performance of queries that filter by the topic field, as they can skip reading irrelevant partitions. References:

? Table Access Control: <https://docs.databricks.com/security/access-control/table-acls/index.html>

? DELETE: <https://docs.databricks.com/delta/delta-update.html#delete-from-a-table>

NEW QUESTION 161

.....

Thank You for Trying Our Product

We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

Databricks-Certified-Professional-Data-Engineer Practice Exam Features:

- * Databricks-Certified-Professional-Data-Engineer Questions and Answers Updated Frequently
- * Databricks-Certified-Professional-Data-Engineer Practice Questions Verified by Expert Senior Certified Staff
- * Databricks-Certified-Professional-Data-Engineer Most Realistic Questions that Guarantee you a Pass on Your FirstTry
- * Databricks-Certified-Professional-Data-Engineer Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year

100% Actual & Verified — Instant Download, Please Click
[Order The Databricks-Certified-Professional-Data-Engineer Practice Test Here](#)