

Exam Questions Databricks-Certified-Professional-Data-Engineer

Databricks Certified Data Engineer Professional Exam

<https://www.2passeasy.com/dumps/Databricks-Certified-Professional-Data-Engineer/>



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

An upstream source writes Parquet data as hourly batches to directories named with the current date. A nightly batch job runs the following code to ingest all data from the previous day as indicated by the date variable:

```
(spark.read
  .format("parquet")
  .load(f"/mnt/raw_orders/{date}")
  .dropDuplicates(["customer_id", "order_id"])
  .write
  .mode("append")
  .saveAsTable("orders")
)
```

Assume that the fields customer_id and order_id serve as a composite key to uniquely identify each order.

If the upstream system is known to occasionally produce duplicate entries for a single order hours apart, which statement is correct?

- A. Each write to the orders table will only contain unique records, and only those records without duplicates in the target table will be written.
- B. Each write to the orders table will only contain unique records, but newly written records may have duplicates already present in the target table.
- C. Each write to the orders table will only contain unique records; if existing records with the same key are present in the target table, these records will be overwritten.
- D. Each write to the orders table will only contain unique records; if existing records with the same key are present in the target table, the operation will fail.
- E. Each write to the orders table will run deduplication over the union of new and existing records, ensuring no duplicate records are present.

Answer: B

Explanation:

This is the correct answer because the code uses the dropDuplicates method to remove any duplicate records within each batch of data before writing to the orders table. However, this method does not check for duplicates across different batches or in the target table, so it is possible that newly written records may have duplicates already present in the target table. To avoid this, a better approach would be to use Delta Lake and perform an upsert operation using mergeInto. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "DROP DUPLICATES" section.

NEW QUESTION 3

In order to facilitate near real-time workloads, a data engineer is creating a helper function to leverage the schema detection and evolution functionality of Databricks Auto Loader. The desired function will automatically detect the schema of the source directly, incrementally process JSON files as they arrive in a source directory, and automatically evolve the schema of the table when new fields are detected.

The function is displayed below with a blank:

Which response correctly fills in the blank to meet the specified requirements?

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

Answer: B

Explanation:

Option B correctly fills in the blank to meet the specified requirements. Option B uses the "cloudFiles.schemaLocation" option, which is required for the schema detection and evolution functionality of Databricks Auto Loader. Additionally, option B uses the "mergeSchema" option, which is required for the schema evolution functionality of Databricks Auto Loader. Finally, option B uses the "writeStream" method, which is required for the incremental processing of JSON files as they

arrive in a source directory. The other options are incorrect because they either omit the required options, use the wrong method, or use the wrong format.
References:

? Configure schema inference and evolution in Auto Loader:

<https://docs.databricks.com/en/ingestion/auto-loader/schema.html>

? Write streaming data: <https://docs.databricks.com/spark/latest/structured-streaming/writing-streaming-data.html>

NEW QUESTION 4

A data ingestion task requires a one-TB JSON dataset to be written out to Parquet with a target part-file size of 512 MB. Because Parquet is being used instead of Delta Lake, built-in file-sizing features such as Auto-Optimize & Auto-Compaction cannot be used.

Which strategy will yield the best performance without shuffling data?

- A. Set `spark.sql.files.maxPartitionBytes` to 512 MB, ingest the data, execute the narrow transformations, and then write to parquet.
- B. Set `spark.sql.shuffle.partitions` to 2,048 partitions ($1\text{TB} \times 1024 \times 1024 / 512$), ingest the data, execute the narrow transformations, optimize the data by sorting it (which automatically repartitions the data), and then write to parquet.
- C. Set `spark.sql.adaptive.advisoryPartitionSizeInBytes` to 512 MB bytes, ingest the data, execute the narrow transformations, coalesce to 2,048 partitions ($1\text{TB} \times 1024 \times 1024 / 512$), and then write to parquet.
- D. Ingest the data, execute the narrow transformations, repartition to 2,048 partitions ($1\text{TB} \times 1024 \times 1024 / 512$), and then write to parquet.
- E. Set `spark.sql.shuffle.partitions` to 512, ingest the data, execute the narrow transformations, and then write to parquet.

Answer: B

Explanation:

The key to efficiently converting a large JSON dataset to Parquet files of a specific size without shuffling data lies in controlling the size of the output files directly.

? Setting `spark.sql.files.maxPartitionBytes` to 512 MB configures Spark to process data in chunks of 512 MB. This setting directly influences the size of the part-files in the output, aligning with the target file size.

? Narrow transformations (which do not involve shuffling data across partitions) can then be applied to this data.

? Writing the data out to Parquet will result in files that are approximately the size specified by `spark.sql.files.maxPartitionBytes`, in this case, 512 MB.

? The other options involve unnecessary shuffles or repartitions (B, C, D) or an incorrect setting for this specific requirement (E).

References:

? Apache Spark Documentation: Configuration - `spark.sql.files.maxPartitionBytes`

? Databricks Documentation on Data Sources: Databricks Data Sources Guide

NEW QUESTION 5

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 6

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 7

A data engineer needs to capture pipeline settings from an existing in the workspace, and use them to create and version a JSON file to create a new pipeline. Which command should the data engineer enter in a web terminal configured with the Databricks CLI?

- A. Use the get command to capture the settings for the existing pipeline; remove the pipeline_id and rename the pipeline; use this in a create command
- B. Stop the existing pipeline; use the returned settings in a reset command
- C. Use the alone command to create a copy of an existing pipeline; use the get JSON command to get the pipeline definition; save this to git
- D. Use list pipelines to get the specs for all pipelines; get the pipeline spec from the return results parse and use this to create a pipeline

Answer: A

Explanation:

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

? Databricks Documentation on CLI for Pipelines: Databricks CLI - Pipelines

NEW QUESTION 8

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 9

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 10

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 10

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

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

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

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

Answer: E

Explanation:

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

NEW QUESTION 12

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 16

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 20

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

- A. New files are identified by listing the input directory; new files are incrementally and idempotently loaded into the target Delta Lake table.
- B. Cloud vendor-specific queue storage and notification services are configured to track newly arriving files; new files are incrementally and idempotently into the target Delta Lake table.
- C. Webhook trigger Databricks job to run anytime new data arrives in a source directory; new data automatically merged into target tables using rules inferred from the data.
- D. New files are identified by listing the input directory; the target table is materialized by directory querying all valid files in the source directory.

Answer: A

Explanation:

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

References:

? Databricks Auto Loader Documentation: Auto Loader Guide

? Delta Lake and Auto Loader: Delta Lake Integration

NEW QUESTION 23

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 28

Incorporating unit tests into a PySpark application requires upfront attention to the design of your jobs, or a potentially significant refactoring of existing code.

Which statement describes a main benefit that offset this additional effort?

- A. Improves the quality of your data
- B. Validates a complete use case of your application

- C. Troubleshooting is easier since all steps are isolated and tested individually
- D. Yields faster deployment and execution times
- E. Ensures that all steps interact correctly to achieve the desired end result

Answer: A

NEW QUESTION 30

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

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

Why are the cloned tables no longer working?

- A. The data files compacted by vacuum are not tracked by the cloned metadata; running refresh on the cloned table will pull in recent changes.
- B. Because Type 1 changes overwrite existing records, Delta Lake cannot guarantee data consistency for cloned tables.
- C. The metadata created by the clone operation is referencing data files that were purged as invalid by the vacuum command
- D. Running vacuum automatically invalidates any shallow clones of a table; deep clone should always be used when a cloned table will be repeatedly queried.

Answer: C

Explanation:

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

NEW QUESTION 34

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

Which statement describes what the number alongside this field represents?

- A. The job_id is returned in this field.
- B. The job_id and number of times the job has been are concatenated and returned.
- C. The number of times the job definition has been run in the workspace.
- D. The globally unique ID of the newly triggered run.

Answer: D

Explanation:

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

NEW QUESTION 38

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. Incremental state information should be maintained for 10 minutes for late-arriving data.

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. withWatermark("event_time", "10 minutes")
- B. awaitArrival("event_time", "10 minutes")
- C. await("event_time + '10 minutes'")
- D. slidingWindow("event_time", "10 minutes")
- E. delayWrite("event_time", "10 minutes")

Answer: A

Explanation:

The correct answer is A. withWatermark("event_time", "10 minutes"). This is because the question asks for incremental state information to be maintained for 10 minutes for late-arriving data. The withWatermark method is used to define the watermark for late data. The watermark is a timestamp column and a threshold that tells the system

how long to wait for late data. In this case, the watermark is set to 10 minutes. The other options are incorrect because they are not valid methods or syntax for watermarking in Structured Streaming. References:

? Watermarking: <https://docs.databricks.com/spark/latest/structured-streaming/watermarks.html>

? Windowed aggregations: <https://docs.databricks.com/spark/latest/structured-streaming/window-operations.html>

NEW QUESTION 42

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 47

The data engineering team is migrating an enterprise system with thousands of tables and views into the Lakehouse. They plan to implement the target architecture using a series of bronze, silver, and gold tables. Bronze tables will almost exclusively be used by production data engineering workloads, while silver tables will be used to support both data engineering and machine learning workloads. Gold tables will largely serve business intelligence and reporting purposes. While personal identifying information (PII) exists in all tiers of data, pseudonymization and anonymization rules are in place for all data at the silver and gold levels.

The organization is interested in reducing security concerns while maximizing the ability to collaborate across diverse teams.

Which statement exemplifies best practices for implementing this system?

- A. Isolating tables in separate databases based on data quality tiers allows for easy permissions management through database ACLs and allows physical separation of default storage locations for managed tables.
- B. Because databases on Databricks are merely a logical construct, choices around database organization do not impact security or discoverability in the Lakehouse.
- C. Storing all production tables in a single database provides a unified view of all data assets available throughout the Lakehouse, simplifying discoverability by granting all users view privileges on this database.
- D. Working in the default Databricks database provides the greatest security when working with managed tables, as these will be created in the DBFS root.
- E. Because all tables must live in the same storage containers used for the database they're created in, organizations should be prepared to create between dozens and thousands of databases depending on their data isolation requirements.

Answer: A

Explanation:

This is the correct answer because it exemplifies best practices for implementing this system. By isolating tables in separate databases based on data quality tiers, such as bronze, silver, and gold, the data engineering team can achieve several benefits. First, they can easily manage permissions for different users and groups through database ACLs, which allow granting or revoking access to databases, tables, or views. Second, they can physically separate the default storage locations for managed tables in each database, which can improve performance and reduce costs. Third, they can provide a clear and consistent naming convention for the tables in each database, which can improve discoverability and usability. Verified References: [Databricks Certified Data Engineer Professional], under "Lakehouse" section; Databricks Documentation, under "Database object privileges" section.

NEW QUESTION 50

A Delta Lake table was created with the below query:

Consider the following query:

```
DROP TABLE prod.sales_by_store -
```

If this statement is executed by a workspace admin, which result will occur?

- A. Nothing will occur until a COMMIT command is executed.
- B. The table will be removed from the catalog but the data will remain in storage.
- C. The table will be removed from the catalog and the data will be deleted.
- D. An error will occur because Delta Lake prevents the deletion of production data.
- E. Data will be marked as deleted but still recoverable with Time Travel.

Answer: C

Explanation:

When a table is dropped in Delta Lake, the table is removed from the catalog and the data is deleted. This is because Delta Lake is a transactional storage layer that provides ACID guarantees. When a table is dropped, the transaction log is updated to reflect the deletion of the table and the data is deleted from the

underlying storage. References:

? <https://docs.databricks.com/delta/quick-start.html#drop-a-table>

? <https://docs.databricks.com/delta/delta-batch.html#drop-table>

NEW QUESTION 55

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 60

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 64

A developer has successfully configured credential for Databricks Repos and cloned a remote Git repository. They do not have privileges to make changes to the main branch, which is the only branch currently visible in their workspace.

Use Response to pull changes from the remote Git repository commit and push changes to a branch that appeared as a change were pulled.

- A. Use Repos to merge all differences and make a pull request back to the remote repository.
- B. Use repos to merge all difference and make a pull request back to the remote repository.
- C. Use Repos to create a new branch commit all changes and push changes to the remote Git repository.
- D. Use repos to create a fork of the remote repository commit all changes and make a pull request on the source repository

Answer: C

Explanation:

In Databricks Repos, when a user does not have privileges to make changes directly to the main branch of a cloned remote Git repository, the recommended approach is to create a new branch within the Databricks workspace. The developer can then make changes in this new branch, commit those changes, and push the new branch to the remote Git repository. This workflow allows for isolated development without affecting the main branch, enabling the developer to propose changes via a pull request from the new branch to the main branch in the remote repository. This method adheres to common Git collaboration workflows, fostering code review and collaboration while ensuring the integrity of the main branch.

References:

? Databricks documentation on using Repos with Git: <https://docs.databricks.com/repos.html>

NEW QUESTION 65

An upstream system has been configured to pass the date for a given batch of data to the Databricks Jobs API as a parameter. The notebook to be scheduled will use this parameter to load data with the following code:

```
df = spark.read.format("parquet").load(f"/mnt/source/{date}")
```

Which code block should be used to create the date Python variable used in the above code block?

- A. `date = spark.conf.get("date")`
- B. `input_dict = input() date= input_dict["date"]`
- C. `import sys date = sys.argv[1]`
- D. `date = dbutils.notebooks.getParam("date")`
- E. `dbutils.widgets.text("date", "null") date = dbutils.widgets.get("date")`

Answer: E

Explanation:

The code block that should be used to create the date Python variable used in the above code block is:

```
dbutils.widgets.text("date", "null") date = dbutils.widgets.get("date")
```

This code block uses the `dbutils.widgets` API to create and get a text widget named "date" that can accept a string value as a parameter¹. The default value of the widget is "null", which means that if no parameter is passed, the date variable will be "null". However, if a parameter is passed through the Databricks Jobs API, the date variable will be assigned the value of the parameter. For example, if the parameter is "2021-11-01", the date variable will be "2021-11-01". This way, the notebook can use the date variable to load data from the specified path.

The other options are not correct, because:

? Option A is incorrect because `spark.conf.get("date")` is not a valid way to get a parameter passed through the Databricks Jobs API. The `spark.conf` API is used to get or set Spark configuration properties, not notebook parameters².

? Option B is incorrect because `input()` is not a valid way to get a parameter passed through the Databricks Jobs API. The `input()` function is used to get user input from the standard input stream, not from the API request³.

? Option C is incorrect because `sys.argv1` is not a valid way to get a parameter passed through the Databricks Jobs API. The `sys.argv` list is used to get the command-line arguments passed to a Python script, not to a notebook⁴.

? Option D is incorrect because `dbutils.notebooks.getParam("date")` is not a valid way to get a parameter passed through the Databricks Jobs API. The `dbutils.notebooks` API is used to get or set notebook parameters when running a notebook as a job or as a subnotebook, not when passing parameters through the API⁵.

References: Widgets, Spark Configuration, `input()`, `sys.argv`, Notebooks

NEW QUESTION 69

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 74

A distributed team of data analysts share computing resources on an interactive cluster with autoscaling configured. In order to better manage costs and query throughput, the workspace administrator is hoping to evaluate whether cluster upscaling is caused by many concurrent users or resource-intensive queries.

In which location can one review the timeline for cluster resizing events?

- A. Workspace audit logs
- B. Driver's log file
- C. Ganglia
- D. Cluster Event Log
- E. Executor's log file

Answer: C

NEW QUESTION 75

A CHECK constraint has been successfully added to the Delta table named `activity_details` using the following logic:

A batch job is attempting to insert new records to the table, including a record where `latitude = 45.50` and `longitude = 212.67`.

Which statement describes the outcome of this batch insert?

- A. The write will fail when the violating record is reached; any records previously processed will be recorded to the target table.
- B. The write will fail completely because of the constraint violation and no records will be inserted into the target table.
- C. The write will insert all records except those that violate the table constraints; the violating records will be recorded to a quarantine table.
- D. The write will include all records in the target table; any violations will be indicated in the boolean column named `valid_coordinates`.
- E. The write will insert all records except those that violate the table constraints; the violating records will be reported in a warning log.

Answer: B

Explanation:

The CHECK constraint is used to ensure that the data inserted into the table meets the specified conditions. In this case, the CHECK constraint is used to ensure that the latitude and longitude values are within the specified range. If the data does not meet the specified conditions, the write operation will fail completely and no records will be inserted into the target table. This is because Delta Lake supports ACID transactions, which means that either all the data is written or none of it is written. Therefore, the batch insert will fail when it encounters a record that violates the constraint, and the target table will not be updated. References:

? Constraints: <https://docs.delta.io/latest/delta-constraints.html>

? ACID Transactions: <https://docs.delta.io/latest/delta-intro.html#acid-transactions>

NEW QUESTION 79

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 82

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 84

A junior developer complains that the code in their notebook isn't producing the correct results in the development environment. A shared screenshot reveals that while they're using a notebook versioned with Databricks Repos, they're using a personal branch that contains old logic. The desired branch named dev-2.3.9 is not available from the branch selection dropdown.

Which approach will allow this developer to review the current logic for this notebook?

- A. Use Repos to make a pull request use the Databricks REST API to update the current branch to dev-2.3.9
- B. Use Repos to pull changes from the remote Git repository and select the dev-2.3.9 branch.
- C. Use Repos to checkout the dev-2.3.9 branch and auto-resolve conflicts with the current branch
- D. Merge all changes back to the main branch in the remote Git repository and clone the repo again
- E. Use Repos to merge the current branch and the dev-2.3.9 branch, then make a pull request to sync with the remote repository

Answer: B

Explanation:

This is the correct answer because it will allow the developer to update their local repository with the latest changes from the remote repository and switch to the desired branch. Pulling changes will not affect the current branch or create any conflicts, as it will only fetch the changes and not merge them. Selecting the dev-2.3.9 branch from the dropdown will checkout that branch and display its contents in the notebook. Verified References: [Databricks Certified Data Engineer Professional], under "Databricks Tooling" section; Databricks Documentation, under "Pull changes from a remote repository" section.

NEW QUESTION 85

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 86

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 88

A DLT pipeline includes the following streaming tables:

Raw_iot ingest raw device measurement data from a heart rate tracking device. Bgm_stats incrementally computes user statistics based on BPM measurements from raw_iot.

How can the data engineer configure this pipeline to be able to retain manually deleted or updated records in the raw_iot table while recomputing the downstream table when a pipeline update is run?

- A. Set the skipChangeCommits flag to true on bpm_stats
- B. Set the SkipChangeCommits flag to true raw_iot
- C. Set the pipelines, reset, allowed property to false on bpm_stats
- D. Set the pipelines, reset, allowed property to false on raw_iot

Answer: D

Explanation:

In Databricks Lakehouse, to retain manually deleted or updated records in the raw_iot table while recomputing downstream tables when a pipeline update is run, the property pipelines.reset.allowed should be set to false. This property prevents the system from resetting the state of the table, which includes the removal of the history of changes, during a pipeline update. By keeping this property as false, any changes to the raw_iot table, including manual deletes or updates, are retained, and recomputation of downstream tables, such as bpm_stats, can occur with the full history of data changes intact. References:

? Databricks documentation on DLT pipelines: <https://docs.databricks.com/data-engineering/delta-live-tables/delta-live-tables-overview.html>

NEW QUESTION 91

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 92

The data architect has mandated that all tables in the Lakehouse should be configured as external (also known as "unmanaged") Delta Lake tables.

Which approach will ensure that this requirement is met?

- A. When a database is being created, make sure that the LOCATION keyword is used.
- B. When configuring an external data warehouse for all table storage, leverage Databricks for all ELT.
- C. When data is saved to a table, make sure that a full file path is specified alongside the Delta format.
- D. When tables are created, make sure that the EXTERNAL keyword is used in the CREATE TABLE statement.
- E. When the workspace is being configured, make sure that external cloud object storage has been mounted.

Answer: D

Explanation:

To create an external or unmanaged Delta Lake table, you need to use the EXTERNAL keyword in the CREATE TABLE statement. This indicates that the table is not managed by the catalog and the data files are not deleted when the table is dropped. You also need to provide a LOCATION clause to specify the path where the data files are stored. For example:

```
CREATE EXTERNAL TABLE events ( date DATE, eventId STRING, eventType STRING, data STRING) USING DELTA LOCATION '/mnt/delta/events';
```

This creates an external Delta Lake table named events that references the data files in the '/mnt/delta/events' path. If you drop this table, the data files will remain intact and you can recreate the table with the same statement.

References:

? <https://docs.databricks.com/delta/delta-batch.html#create-a-table>

? <https://docs.databricks.com/delta/delta-batch.html#drop-a-table>

NEW QUESTION 93

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

- A. %python
- B. % Databricks notebook source
- C. -- Databricks notebook source
- D. //Databricks notebook source

Answer: B

Explanation:

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

NEW QUESTION 97

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 98

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

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

- A. Date
- B. Post_id
- C. User_id
- D. Post_time

Answer: A

Explanation:

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

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

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

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

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

References:

? Delta Lake Documentation on Table Partitioning: Optimizing Layout with Partitioning

NEW QUESTION 102

Which Python variable contains a list of directories to be searched when trying to locate required modules?

- A. importlib.resource path
- B. sys.path
- C. os.path
- D. pypi.path
- E. pylib.source

Answer: B

NEW QUESTION 106

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

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

- A. userLookup.join(streamingDF, ["userid"], how="inner")
- B. streamingDF.join(userLookup, ["user_id"], how="outer")
- C. streamingDF.join(userLookup, ["user_id"], how="left")
- D. streamingDF.join(userLookup, ["userid"], how="inner")
- E. userLookup.join(streamingDF, ["user_id"], how="right")

Answer: E

Explanation:

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

References:

? Structured Streaming Programming Guide: Join Operations

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

NEW QUESTION 109

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 111

Which configuration parameter directly affects the size of a spark-partition upon ingestion of data into Spark?

- A. spark.sql.files.maxPartitionBytes
- B. spark.sql.autoBroadcastJoinThreshold
- C. spark.sql.files.openCostInBytes
- D. spark.sql.adaptive.coalescePartitions.minPartitionNum
- E. spark.sql.adaptive.advisoryPartitionSizeInBytes

Answer: A

Explanation:

This is the correct answer because spark.sql.files.maxPartitionBytes is a configuration parameter that directly affects the size of a spark-partition upon ingestion of data into Spark. This parameter configures the maximum number of bytes to pack into a single partition when reading files from file-based sources such as Parquet, JSON and ORC. The default value is 128 MB, which means each partition will be roughly 128 MB in size, unless there are too many small files or only one large file. Verified References: [Databricks Certified Data Engineer Professional], under “Spark Configuration” section; Databricks Documentation, under “Available Properties - spark.sql.files.maxPartitionBytes” section.

NEW QUESTION 114

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

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

Which piece of information is critical to this decision?

- A. Delta Lake time travel does not scale well in cost or latency to provide a long-term versioning solution.
- B. Delta Lake time travel cannot be used to query previous versions of these tables because Type 1 changes modify data files in place.
- C. Shallow clones can be combined with Type 1 tables to accelerate historic queries for long-term versioning.
- D. Data corruption can occur if a query fails in a partially completed state because Type 2 tables requiresSetting multiple fields in a single update.

Answer: A

Explanation:

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

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

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

NEW QUESTION 115

Although the Databricks Utilities Secrets module provides tools to store sensitive credentials and avoid accidentally displaying them in plain text users should still be careful with which credentials are stored here and which users have access to using these secrets.

Which statement describes a limitation of Databricks Secrets?

- A. Because the SHA256 hash is used to obfuscate stored secrets, reversing this hash will display the value in plain text.
- B. Account administrators can see all secrets in plain text by logging on to the Databricks Accounts console.
- C. Secrets are stored in an administrators-only table within the Hive Metastore; database administrators have permission to query this table by default.
- D. Iterating through a stored secret and printing each character will display secret contents in plain text.
- E. The Databricks REST API can be used to list secrets in plain text if the personal access token has proper credentials.

Answer: E

Explanation:

This is the correct answer because it describes a limitation of Databricks Secrets. Databricks Secrets is a module that provides tools to store sensitive credentials and avoid accidentally displaying them in plain text. Databricks Secrets allows creating secret scopes, which are collections of secrets that can be accessed by users or groups. Databricks Secrets also allows creating and managing secrets using the Databricks CLI or the Databricks REST API. However, a limitation of Databricks Secrets is that the Databricks REST API can be used to list secrets in plain text if the personal access token has proper credentials. Therefore, users should still be careful with which credentials are stored in Databricks Secrets and which users have access to using these secrets. Verified References: [Databricks Certified Data Engineer Professional], under “Databricks Workspace” section; Databricks Documentation, under “List secrets” section.

NEW QUESTION 117

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 119

Which of the following is true of Delta Lake and the Lakehouse?

- A. Because Parquet compresses data row by row
- B. strings will only be compressed when a character is repeated multiple times.
- C. Delta Lake automatically collects statistics on the first 32 columns of each table which are leveraged in data skipping based on query filters.
- D. Views in the Lakehouse maintain a valid cache of the most recent versions of source tables at all times.
- E. Primary and foreign key constraints can be leveraged to ensure duplicate values are never entered into a dimension table.
- F. Z-order can only be applied to numeric values stored in Delta Lake tables

Answer: B

Explanation:

<https://docs.delta.io/2.0.0/table-properties.html>

Delta Lake automatically collects statistics on the first 32 columns of each table, which are leveraged in data skipping based on query filters¹. Data skipping is a performance optimization technique that aims to avoid reading irrelevant data from the storage layer¹. By collecting statistics such as min/max values, null counts, and bloom filters, Delta Lake can efficiently prune unnecessary files or partitions from the query plan¹. This can significantly improve the query performance and reduce the I/O cost.

The other options are false because:

? Parquet compresses data column by column, not row by row². This allows for better compression ratios, especially for repeated or similar values within a column².

? Views in the Lakehouse do not maintain a valid cache of the most recent versions of source tables at all times³. Views are logical constructs that are defined by a SQL query on one or more base tables³. Views are not materialized by default, which means they do not store any data, but only the query definition³.

Therefore, views always reflect the latest state of the source tables when queried³. However, views can be cached manually using the `CACHE TABLE` or `CREATE TABLE AS SELECT` commands.

? Primary and foreign key constraints can not be leveraged to ensure duplicate values are never entered into a dimension table. Delta Lake does not support enforcing primary and foreign key constraints on tables. Constraints are logical rules that define the integrity and validity of the data in a table. Delta Lake relies on the application logic or the user to ensure the data quality and consistency.

? Z-order can be applied to any values stored in Delta Lake tables, not only numeric values. Z-order is a technique to optimize the layout of the data files by sorting them on one or more columns. Z-order can improve the query performance by clustering related values together and enabling more efficient data skipping. Z-order can be applied to any column that has a defined ordering, such as numeric, string, date, or boolean values.

References: Data Skipping, Parquet Format, Views, [Caching], [Constraints], [Z-Ordering]

NEW QUESTION 124

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 128

Each configuration below is identical to the extent that each cluster has 400 GB total of RAM, 160 total cores and only one Executor per VM.

Given a job with at least one wide transformation, which of the following cluster configurations will result in maximum performance?

- A. • Total VMs: 1 • 400 GB per Executor • 160 Cores / Executor
- B. • Total VMs: 8 • 50 GB per Executor • 20 Cores / Executor
- C. • Total VMs: 4 • 100 GB per Executor • 40 Cores/Executor
- D. • Total VMs: 2 • 200 GB per Executor • 80 Cores / Executor

Answer: B

Explanation:

This is the correct answer because it is the cluster configuration that will result in maximum performance for a job with at least one wide transformation. A wide transformation is a type of transformation that requires shuffling data across partitions, such as join, groupBy, or orderBy. Shuffling can be expensive and time-consuming, especially if there are too many or too few partitions. Therefore, it is important to choose a cluster configuration that can balance the trade-off between parallelism and network overhead. In this case, having 8 VMs with 50 GB per executor and 20 cores per executor will create 8 partitions, each with enough memory and CPU resources to handle the shuffling efficiently. Having fewer VMs with more memory and cores per executor will create fewer partitions, which will reduce parallelism and increase the size of each shuffle block. Having more VMs with less memory and cores per executor will create more partitions, which will increase parallelism but also increase the network overhead and the number of shuffle files. Verified References: [Databricks Certified Data Engineer Professional], under “Performance Tuning” section; Databricks Documentation, under “Cluster configurations” section.

NEW QUESTION 131

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-](https://docs.databricks.com/security/access-control/table-acls/index.html)

[acls/index.html](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 134

.....

THANKS FOR TRYING THE DEMO OF OUR PRODUCT

Visit Our Site to Purchase the Full Set of Actual Databricks-Certified-Professional-Data-Engineer Exam Questions With Answers.

We Also Provide Practice Exam Software That Simulates Real Exam Environment And Has Many Self-Assessment Features. Order the Databricks-Certified-Professional-Data-Engineer Product From:

<https://www.2passeasy.com/dumps/Databricks-Certified-Professional-Data-Engineer/>

Money Back Guarantee

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