

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 seeks to leverage Delta Lake's Change Data Feed functionality to create a Type 1 table representing all of the values that have ever been valid for all rows in a bronze table created with the property `delta.enableChangeDataFeed = true`. They plan to execute the following code as a daily job:

Which statement describes the execution and results of running the above query multiple times?

- A. Each time the job is executed, newly updated records will be merged into the target table, overwriting previous values with the same primary keys.
- B. Each time the job is executed, the entire available history of inserted or updated records will be appended to the target table, resulting in many duplicate entries.
- C. Each time the job is executed, the target table will be overwritten using the entire history of inserted or updated records, giving the desired result.
- D. Each time the job is executed, the differences between the original and current versions are calculated; this may result in duplicate entries for some records.
- E. Each time the job is executed, only those records that have been inserted or updated since the last execution will be appended to the target table giving the desired result.

Answer: B

Explanation:

Reading table's changes, captured by CDF, using `spark.read` means that you are reading them as a static source. So, each time you run the query, all table's changes (starting from the specified startingVersion) will be read.

NEW QUESTION 4

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 5

The business intelligence team has a dashboard configured to track various summary metrics for retail stores. This includes total sales for the previous day alongside totals and averages for a variety of time periods. The fields required to populate this dashboard have the following schema:

For Demand forecasting, the Lakehouse contains a validated table of all itemized sales updated incrementally in near real-time. This table named `products_per_order`, includes the following fields:

Because reporting on long-term sales trends is less volatile, analysts using the new dashboard only require data to be refreshed once daily. Because the dashboard will be queried interactively by many users throughout a normal business day, it should return results quickly and reduce total compute associated with each materialization.

Which solution meets the expectations of the end users while controlling and limiting possible costs?

- A. Use the Delta Cache to persist the `products_per_order` table in memory to quickly refresh the dashboard with each query.
- B. Populate the dashboard by configuring a nightly batch job to save the required data to quickly update the dashboard with each query.
- C. Use Structure Streaming to configure a live dashboard against the `products_per_order` table within a Databricks notebook.
- D. Define a view against the `products_per_order` table and define the dashboard against this view.

Answer: D

Explanation:

Given the requirement for daily refresh of data and the need to ensure quick response times for interactive queries while controlling costs, a nightly batch job to pre-compute and save the required summary metrics is the most suitable approach.

? By pre-aggregating data during off-peak hours, the dashboard can serve queries quickly without requiring on-the-fly computation, which can be resource-intensive and slow, especially with many users.

? This approach also limits the cost by avoiding continuous computation throughout the day and instead leverages a batch process that efficiently computes and stores the necessary data.

? The other options (A, C, D) either do not address the cost and performance requirements effectively or are not suitable for the use case of less frequent data refresh and high interactivity.

References:

? Databricks Documentation on Batch Processing: Databricks Batch Processing

? Data Lakehouse Patterns: Data Lakehouse Best Practices

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 version 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

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 9

A Delta Lake table was created with the below query:

Realizing that the original query had a typographical error, the below code was executed: ALTER TABLE prod.sales_by_stor RENAME TO prod.sales_by_store Which result will occur after running the second command?

- A. The table reference in the metastore is updated and no data is changed.
- B. The table name change is recorded in the Delta transaction log.
- C. All related files and metadata are dropped and recreated in a single ACID transaction.
- D. The table reference in the metastore is updated and all data files are moved.
- E. A new Delta transaction log is created for the renamed table.

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 result that will occur after running the second command is that the table reference in the metastore is updated and no data is changed. The metastore is a service that stores metadata about tables, such as their schema, location, properties, and partitions. The metastore allows users to access tables using SQL commands or Spark APIs without knowing their physical location or format. When renaming an external table using the ALTER TABLE RENAME TO command, only the table reference in the metastore is updated with the new name; no data files or directories are moved or changed in the storage system. The table will still point to the same location and use the same format as before. However, if renaming a managed table, which is a table whose metadata and data are both managed by Databricks, both the table reference in the metastore and the data files in the default warehouse directory are moved and renamed accordingly. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "ALTER TABLE RENAME TO" section; Databricks Documentation, under "Metastore" section; Databricks Documentation, under "Managed and external tables" section.

NEW QUESTION 10

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 10

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 13

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 15

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 16

A junior data engineer has manually configured a series of jobs using the Databricks Jobs UI. Upon reviewing their work, the engineer realizes that they are listed as the "Owner" for each job. They attempt to transfer "Owner" privileges to the "DevOps" group, but cannot successfully accomplish this task. Which statement explains what is preventing this privilege transfer?

- A. Databricks jobs must have exactly one owner; "Owner" privileges cannot be assigned to a group.
- B. The creator of a Databricks job will always have "Owner" privileges; this configuration cannot be changed.
- C. Other than the default "admins" group, only individual users can be granted privileges on jobs.
- D. A user can only transfer job ownership to a group if they are also a member of that group.
- E. Only workspace administrators can grant "Owner" privileges to a group.

Answer: E

Explanation:

The reason why the junior data engineer cannot transfer "Owner" privileges to the "DevOps" group is that Databricks jobs must have exactly one owner, and the owner must be an individual user, not a group. A job cannot have more than one owner, and a job cannot have a group as an owner. The owner of a job is the user who created the job, or the user who was assigned the ownership by another user. The owner of a job has the highest level of permission on the job, and can grant or revoke permissions to other users or groups. However, the owner cannot transfer the ownership to a group, only to another user. Therefore, the junior data engineer's attempt to transfer "Owner" privileges to the "DevOps" group is not possible. References:

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

? Job permissions: <https://docs.databricks.com/security/access-control/table-acls/privileges.html#job-permissions>

NEW QUESTION 17

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 22

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 23

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 25

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 27

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 29

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 34

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 36

Which statement regarding stream-static joins and static Delta tables is correct?

- A. Each microbatch of a stream-static join will use the most recent version of the static Delta table as of each microbatch.
- B. Each microbatch of a stream-static join will use the most recent version of the static Delta table as of the job's initialization.
- C. The checkpoint directory will be used to track state information for the unique keys present in the join.
- D. Stream-static joins cannot use static Delta tables because of consistency issues.
- E. The checkpoint directory will be used to track updates to the static Delta table.

Answer: A

Explanation:

This is the correct answer because stream-static joins are supported by Structured Streaming when one of the tables is a static Delta table. A static Delta table is a Delta table that is not updated by any concurrent writes, such as appends or merges, during the execution of a streaming query. In this case, each microbatch of a stream-static join will use the most recent version of the static Delta table as of each microbatch, which means it will reflect any changes made to the static Delta table before the start of each microbatch. Verified References: [Databricks Certified Data Engineer Professional], under "Structured Streaming" section; Databricks Documentation, under "Stream and static joins" section.

NEW QUESTION 40

The Databricks workspace administrator has configured interactive clusters for each of the data engineering groups. To control costs, clusters are set to terminate after 30 minutes of inactivity. Each user should be able to execute workloads against their assigned clusters at any time of the day. Assuming users have been added to a workspace but not granted any permissions, which of the following describes the minimal permissions a user would need to start and attach to an already configured cluster.

- A. "Can Manage" privileges on the required cluster
- B. Workspace Admin privileges, cluster creation allowe
- C. "Can Attach To" privileges on the required cluster
- D. Cluster creation allowe
- E. "Can Attach To" privileges on the required cluster
- F. "Can Restart" privileges on the required cluster
- G. Cluster creation allowe
- H. "Can Restart" privileges on the required cluster

Answer: D

Explanation:

<https://learn.microsoft.com/en-us/azure/databricks/security/auth-authz/access-control/cluster-acl>

<https://docs.databricks.com/en/security/auth-authz/access-control/cluster-acl.html>

NEW QUESTION 42

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

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

- A. Run unit tests against non-production data that closely mirrors production
- B. Define and unit test functions using Files in Repos
- C. Define units test and functions within the same notebook
- D. Define and import unit test functions from a separate Databricks notebook

Answer: A

Explanation:

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

References:

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

NEW QUESTION 43

The following table consists of items found in user carts within an e-commerce website.

```
Carts (id LONG, items ARRAY<STRUCT<id: LONG, count: INT>>)
id items email
1001[[{id: "DESK65", count: 1}]] "u1@domain.com"
1002[[{id: "KYBD45", count: 1}, {id: "M27", count: 2}]] "u2@domain.com"
1003[[{id: "M27", count: 1}]] "u3@domain.com"
```

The following MERGE statement is used to update this table using an updates view, with schema evaluation enabled on this table.

```
MERGE INTO carts c
USING updates u
ON c.id = u.id
WHEN MATCHED
THEN UPDATE SET *
```

How would the following update be handled?

```
(new nested field, missing existing column)
id items
1001[[{id: "DESK65", count: 2, coupon: "BOGO50"}]]
```

How would the following update be handled?

- A. The update is moved to separate "restored" column because it is missing a column expected in the target schema.
- B. The new restored field is added to the target schema, and dynamically read as NULL for existing unmatched records.
- C. The update throws an error because changes to existing columns in the target schema are not supported.
- D. The new nested field is added to the target schema, and files underlying existing records are updated to include NULL values for the new field.

Answer: D

Explanation:

With schema evolution enabled in Databricks Delta tables, when a new field is added to a record through a MERGE operation, Databricks automatically modifies the table schema to include the new field. In existing records where this new field is not present, Databricks will insert NULL values for that field. This ensures that the schema remains consistent across all records in the table, with the new field being present in every record, even if it is NULL for records that did not originally include it.

References:

? Databricks documentation on schema evolution in Delta Lake: <https://docs.databricks.com/delta/delta-batch.html#schema-evolution>

NEW QUESTION 47

A data engineer wants to join a stream of advertisement impressions (when an ad was shown) with another stream of user clicks on advertisements to correlate when impression led to monetizable clicks.

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

Which solution would improve the performance?

- A) `Joining on event time constraint: clickTime == impressionTime using a leftOuter join`
- B) `Joining on event time constraint: clickTime >= impressionTime - interval 3 hours and removing watermarks`
- C) `Joining on event time constraint: clickTime + 3 hours < impressionTime - 2 hours`
- D) `Joining on event time constraint: clickTime >= impressionTime AND clickTime <= impressionTime + interval 1 hour`

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

Answer: A

Explanation:

When joining a stream of advertisement impressions with a stream of user clicks, you want to minimize the state that you need to maintain for the join. Option A suggests using a left outer join with the condition that `clickTime == impressionTime`, which is suitable for correlating events that occur at the exact same time. However, in a real-world scenario, you would likely need some leeway to account for the delay between an impression and a possible click. It's important to design the join condition and the window of time considered to optimize performance while still capturing the relevant user interactions. In this case, having the watermark can help with state management and avoid state growing unbounded by discarding old state data that's unlikely to match with new data.

NEW QUESTION 52

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 56

A data engineer, User A, has promoted a new pipeline to production by using the REST API to programmatically create several jobs. A DevOps engineer, User B, has configured an external orchestration tool to trigger job runs through the REST API. Both users authorized the REST API calls using their personal access tokens.

Which statement describes the contents of the workspace audit logs concerning these events?

- A. Because the REST API was used for job creation and triggering runs, a Service Principal will be automatically used to identify these events.
- B. Because User B last configured the jobs, their identity will be associated with both the job creation events and the job run events.
- C. Because these events are managed separately, User A will have their identity associated with the job creation events and User B will have their identity associated with the job run events.
- D. Because the REST API was used for job creation and triggering runs, user identity will not be captured in the audit logs.
- E. Because User A created the jobs, their identity will be associated with both the job creation events and the job run events.

Answer: C

Explanation:

The events are that a data engineer, User A, has promoted a new pipeline to production by using the REST API to programmatically create several jobs, and a DevOps engineer, User B, has configured an external orchestration tool to trigger job runs through the REST API. Both users authorized the REST API calls using their personal access tokens. The workspace audit logs are logs that record user activities in a Databricks workspace, such as creating, updating, or deleting objects like clusters, jobs, notebooks, or tables. The workspace audit logs also capture the identity of the user who performed each activity, as well as the time and details of the activity. Because these events are managed separately, User A will have their identity associated with the job creation events and User B will have their identity associated with the job run events in the workspace audit logs. Verified References: [Databricks Certified Data Engineer Professional], under "Databricks Workspace" section; Databricks Documentation, under "Workspace audit logs" section.

NEW QUESTION 58

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

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

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

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

...
```

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

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

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

The pipeline runs an update with this refactored code, but generates a different DAG showing incorrect configuration values for tables. How can the data engineer fix this?

- A. Convert the list of configuration values to a dictionary of table settings, using table names as keys.
- B. Convert the list of configuration values to a dictionary of table settings, using different input the for loop.
- C. Load the configuration values for these tables from a separate file, located at a path provided by a pipeline parameter.

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

Answer: A

Explanation:

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

NEW QUESTION 60

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

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

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

- A. "Read" permissions should be set on a secret key mapped to those credentials that will be used by a given team.
- B. No additional configuration is necessary as long as all users are configured as administrators in the workspace where secrets have been added.
- C. "Read" permissions should be set on a secret scope containing only those credentials that will be used by a given team.
- D. "Manage" permission should be set on a secret scope containing only those credentials that will be used by a given team.

Answer: C

Explanation:

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

References:

? Databricks Documentation on Secret Management: Secrets

NEW QUESTION 62

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 67

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 71

An upstream system is emitting change data capture (CDC) logs that are being written to a cloud object storage directory. Each record in the log indicates the change type (insert, update, or delete) and the values for each field after the change. The source table has a primary key identified by the field `pk_id`.

For auditing purposes, the data governance team wishes to maintain a full record of all values that have ever been valid in the source system. For analytical purposes, only the most recent value for each record needs to be recorded. The Databricks job to ingest these records occurs once per hour, but each individual

record may have changed multiple times over the course of an hour.
 Which solution meets these requirements?

- A. Create a separate history table for each pk_id resolve the current state of the table by running a union all filtering the history tables for the most recent state.
- B. Use merge into to insert, update, or delete the most recent entry for each pk_id into a bronze table, then propagate all changes throughout the system.
- C. Iterate through an ordered set of changes to the table, applying each in turn; rely on Delta Lake's versioning ability to create an audit log.
- D. Use Delta Lake's change data feed to automatically process CDC data from an external system, propagating all changes to all dependent tables in the Lakehouse.
- E. Ingest all log information into a bronze table; use merge into to insert, update, or delete the most recent entry for each pk_id into a silver table to recreate the current table state.

Answer: B

Explanation:

This is the correct answer because it meets the requirements of maintaining a full record of all values that have ever been valid in the source system and recreating the current table state with only the most recent value for each record. The code ingests all log information into a bronze table, which preserves the raw CDC data as it is. Then, it uses merge into to perform an upsert operation on a silver table, which means it will insert new records or update or delete existing records based on the change type and the pk_id columns. This way, the silver table will always reflect the current state of the source table, while the bronze table will keep the history of all changes. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Upsert into a table using merge" section.

NEW QUESTION 76

A data pipeline uses Structured Streaming to ingest data from kafka to Delta Lake. Data is being stored in a bronze table, and includes the Kafka_generated timesamp, key, and value. Three months after the pipeline is deployed the data engineering team has noticed some latency issued during certain times of the day. A senior data engineer updates the Delta Table's schema and ingestion logic to include the current timestamp (as recoded by Apache Spark) as well the Kafka topic and partition. The team plans to use the additional metadata fields to diagnose the transient processing delays:
 Which limitation will the team face while diagnosing this problem?

- A. New fields not be computed for historic records.
- B. Updating the table schema will invalidate the Delta transaction log metadata.
- C. Updating the table schema requires a default value provided for each file added.
- D. Spark cannot capture the topic partition fields from the kafka source.

Answer: A

Explanation:

When adding new fields to a Delta table's schema, these fields will not be retrospectively applied to historical records that were ingested before the schema change. Consequently, while the team can use the new metadata fields to investigate transient processing delays moving forward, they will be unable to apply this diagnostic approach to past data that lacks these fields.

References:

? Databricks documentation on Delta Lake schema management: <https://docs.databricks.com/delta/delta-batch.html#schema-management>

NEW QUESTION 80

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 85

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 90

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 91

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 96

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 98

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 102

Which statement describes Delta Lake optimized writes?

- A. A shuffle occurs prior to writing to try to group data together resulting in fewer files instead of each executor writing multiple files based on directory partitions.
- B. Optimized writes logical partitions instead of directory partitions partition boundaries are only represented in metadata fewer small files are written.
- C. An asynchronous job runs after the write completes to detect if files could be further compacted; yes, an OPTIMIZE job is executed toward a default of 1 GB.
- D. Before a job cluster terminates, OPTIMIZE is executed on all tables modified during the most recent job.

Answer: A

Explanation:

Delta Lake optimized writes involve a shuffle operation before writing out data to the Delta table. The shuffle operation groups data by partition keys, which can lead to a reduction in the number of output files and potentially larger files, instead of multiple smaller files. This approach can significantly reduce the total number of files in the table, improve read performance by reducing the metadata overhead, and optimize the table storage layout, especially for workloads with many small files.

References:

? Databricks documentation on Delta Lake performance tuning: <https://docs.databricks.com/delta/optimizations/auto-optimize.html>

NEW QUESTION 103

Which distribution does Databricks support for installing custom Python code packages?

- A. sbt
- B. CRAN
- C. CRAM
- D. nom
- E. Wheels
- F. jars

Answer: D

NEW QUESTION 107

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 108

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 109

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 111

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 112

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 117

An external object storage container has been mounted to the location /mnt/finance_eda_bucket.

The following logic was executed to create a database for the finance team:

After the database was successfully created and permissions configured, a member of the finance team runs the following code:

If all users on the finance team are members of the finance group, which statement describes how the tx_sales table will be created?

- A. A logical table will persist the query plan to the Hive Metastore in the Databricks control plane.
- B. An external table will be created in the storage container mounted to /mnt/finance_eda_bucket.
- C. A logical table will persist the physical plan to the Hive Metastore in the Databricks control plane.
- D. An managed table will be created in the storage container mounted to /mnt/finance_eda_bucket.
- E. A managed table will be created in the DBFS root storage container.

Answer: A

Explanation:

<https://docs.databricks.com/en/lakehouse/data-objects.html>

NEW QUESTION 119

A new data engineer notices that a critical field was omitted from an application that writes its Kafka source to Delta Lake. This happened even though the critical field was in the Kafka source. That field was further missing from data written to dependent, long-term storage. The retention threshold on the Kafka service is seven days. The pipeline has been in production for three months.

Which describes how Delta Lake can help to avoid data loss of this nature in the future?

- A. The Delta log and Structured Streaming checkpoints record the full history of the Kafka producer.
- B. Delta Lake schema evolution can retroactively calculate the correct value for newly added fields, as long as the data was in the original source.
- C. Delta Lake automatically checks that all fields present in the source data are included in the ingestion layer.
- D. Data can never be permanently dropped or deleted from Delta Lake, so data loss is not possible under any circumstance.
- E. Ingesting all raw data and metadata from Kafka to a bronze Delta table creates a permanent, replayable history of the data state.

Answer: E

Explanation:

This is the correct answer because it describes how Delta Lake can help to avoid data loss of this nature in the future. By ingesting all raw data and metadata from Kafka to a bronze Delta table, Delta Lake creates a permanent, replayable history of the data state that can be used for recovery or reprocessing in case of errors or omissions in downstream applications or pipelines. Delta Lake also supports schema evolution, which allows adding new columns to existing tables without affecting existing queries or pipelines. Therefore, if a critical field was omitted from an application that writes its Kafka source to Delta Lake, it can be easily added later and the data can be reprocessed from the bronze table without losing any information. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Delta Lake core features" section.

NEW QUESTION 120

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 121

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 124

.....

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](#)