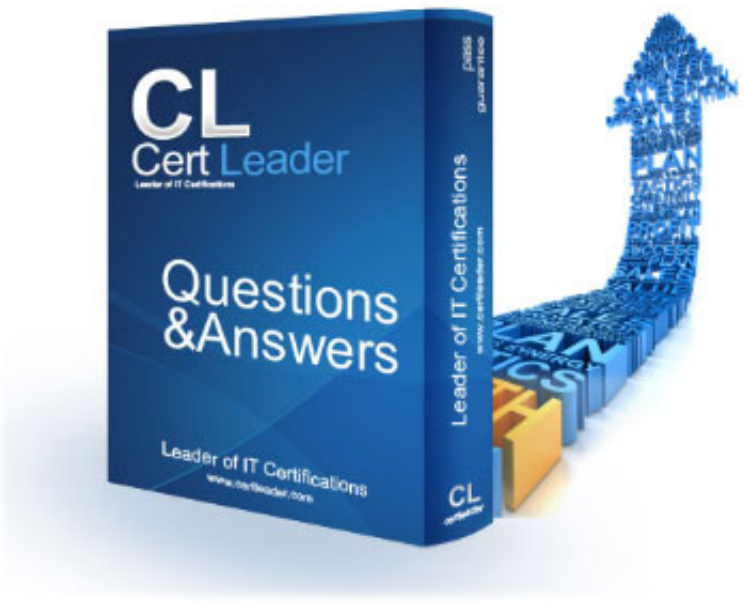
Databricks-Certified-Professional-Data-Engineer Dumps

Databricks Certified Data Engineer Professional Exam

https://www.certleader.com/Databricks-Certified-Professional-Data-Engineer-dumps.html



NEW QUESTION 1
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 2
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 3
The business intelligence team has a dashboard configured to track various summary metrics for retail stories. 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 persists the products_per_order table in memory to quickly the dashboard with each query.
B. Populate the dashboard by configuring a nightly batch job to save the required 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 4
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 5
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 beaccessible 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 6
Which statement characterizes the general programming model used by Spark Structured Streaming?
A. Structured Streaming leverages the parallel processing of GPUs to achieve highly parallel data throughput.
B. Structured Streaming is implemented as a messaging bus and is derived from Apache Kafka.
C. Structured Streaming uses specialized hardware and I/O streams to achieve sub- second latency for data transfer.
D. Structured Streaming models new data arriving in a data stream as new rows appended to an unbounded table.
E. Structured Streaming relies on a distributed network of nodes that hold incremental state values for cached stages.

Answer: B

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

NEW QUESTION 7
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 8
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 ofdefault 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. Storinq 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 9
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 10
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 10
The data science team has created and logged a production model using MLflow. The following code correctly imports and applies the production model to output the predictions as a new DataFrame named preds with the schema "customer_id LONG, predictions DOUBLE, date DATE".

```
from pyspark.sql.functions import current_date

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

The data science team would like predictions saved to a Delta Lake table with the ability to compare all predictions across time. Churn predictions will be made at most once per day.
Which code block accomplishes this task while minimizing potential compute costs?
A) preds.write.mode("append").saveAsTable("churn_preds")
B) preds.write.format("delta").save("/preds/churn_preds")
C)

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

D)

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

E)

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

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

Answer: A

NEW QUESTION 14
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 17
The data governance team has instituted a requirement that all tables containing Personal Identifiable Information (PH) must be clearly annotated. This includes adding column comments, table comments, and setting the custom table property "contains_pii" = true.
The following SQL DDL statement is executed to create a new table:
Which command allows manual confirmation that these three requirements have been met?

A. DESCRIBE EXTENDED dev.pii test
B. DESCRIBE DETAIL dev.pii test
C. SHOW TBLPROPERTIES dev.pii test
D. DESCRIBE HISTORY dev.pii test
E. SHOW TABLES dev

Answer: A

Explanation:
This is the correct answer because it allows manual confirmation that these three requirements have been met. The requirements are that all tables containing Personal Identifiable Information (PII) must be clearly annotated, which includes adding column comments, table comments, and setting the custom table property "contains_pii" = true. The DESCRIBE EXTENDED command is used to display detailed information about a table, such as its schema, location, properties, and comments. By using this command on the dev.pii_test table, one can verify that the table has been created with the correct column comments, table comment, and custom table property as specified in the SQL DDL statement. Verified References: [Databricks Certified Data Engineer Professional], under "Lakehouse" section; Databricks Documentation, under "DESCRIBE EXTENDED" section.

NEW QUESTION 22
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 27
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 28
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 30
Assuming that the Databricks CLI has been installed and configured correctly, which Databricks CLI command can be used to upload a custom Python Wheel to object storage mounted with the DBFS for use with a production job?

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

Answer: B

Explanation:
The libraries command group allows you to install, uninstall, and list libraries on Databricks clusters. You can use the libraries install command to install a custom Python Wheel on a cluster by specifying the --whl option and the path to the wheel file. For example, you can use the following command to install a custom Python Wheel named mylib-0.1-py3-none-any.whl on a cluster with the id 1234-567890-abcde123:
databricks libraries install --cluster-id1234-567890-abcde123--whldbfs:/mnt/mylib/mylib-0.1-py3-none-any.whl
This will upload the custom Python Wheel to the cluster and make it available for use with a production job. You can also use the libraries uninstall command to uninstall a library from a cluster, and the libraries list command to list the libraries installed on a cluster. References:
? Libraries CLI (legacy): https://docs.databricks.com/en/archive/dev-tools/cli/libraries-cli.html
? Library operations: https://docs.databricks.com/en/dev- tools/cli/commands.html#library-operations
? Install or update the Databricks CLI: https://docs.databricks.com/en/dev- tools/cli/install.html

NEW QUESTION 35
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 abronze 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 39
The data architect has decided that once data has been ingested from external sources into the
Databricks Lakehouse, table access controls will be leveraged to manage permissions for all production tables and views.
The following logic was executed to grant privileges for interactive queries on a production database to the core engineering group.
GRANT USAGE ON DATABASE prod TO eng; GRANT SELECT ON DATABASE prod TO eng;
Assuming these are the only privileges that have been granted to the eng group and that these users are not workspace administrators, which statement describes their privileges?

A. Group members have full permissions on the prod database and can also assign permissions to other users or groups.
B. Group members are able to list all tables in the prod database but are not able to see the results of any queries on those tables.
C. Group members are able to query and modify all tables and views in the prod database,but cannot create new tables or views.
D. Group members are able to query all tables and views in the prod database, but cannot create or edit anything in the database.
E. Group members are able to create, query, and modify all tables and views in the prod database, but cannot define custom functions.

Answer: D

Explanation:
The GRANT USAGE ON DATABASE prod TO eng command grants the eng group the permission to use the prod database, which means they can list and access the tables and views in the database. The GRANT SELECT ON DATABASE prod TO eng command grants the eng group the permission to select data from the tables and views in the prod database, which means they can query the data using SQL or DataFrame API. However, these commands do not grant the eng group any other permissions, such as creating, modifying, or deleting tables and views, or defining custom functions. Therefore, the eng group members are able to query all tables and views in the prod database, but cannot create or edit anything in the database. References:
? Grant privileges on a database: https://docs.databricks.com/en/security/auth-authz/table-acls/grant-privileges-database.html
? Privileges you can grant on Hive metastore objects: https://docs.databricks.com/en/security/auth-authz/table-acls/privileges.html

NEW QUESTION 40
A production cluster has 3 executor nodes and uses the same virtual machine type for the driver and executor.
When evaluating the Ganglia Metrics for this cluster, which indicator would signal a bottleneck caused by code executing on the driver?

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

Answer: E

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

NEW QUESTION 44
The data engineering team maintains the following code:

```
accountDF = spark.table("accounts")
orderDF = spark.table("orders")
itemDF = spark.table("items")

orderWithItemDF = (orderDF.join(
    itemDF,
    orderDF.itemID == itemDF.itemID)
  .select(
    orderDF.accountID,
    orderDF.itemID,

    itemDF.itemName))

finalDF = (accountDF.join(
    orderWithItemDF,
    accountDF.accountID == orderWithItemDF.accountID)
  .select(
    orderWithItemDF["*"],

    accountDF.city))

(finalDF.write
  .mode("overwrite")
  .table("enriched_itemized_orders_by_account"))
```

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

A. A batch job will update the enriched_itemized_orders_by_account table, replacing only those rows that have different values than the current version of the table, using accountID as the primary key.
B. The enriched_itemized_orders_by_account table will be overwritten using the current valid version of data in each of the three tables referenced in the join logic.
C. An incremental job will leverage information in the state store to identify unjoined rows in the source tables and write these rows to the enriched_iteinized_orders_by_account table.
D. An incremental job will detect if new rows have been written to any of the source tables; if new rows are detected, all results will be recalculated and used to overwrite the enriched_itemized_orders_by_account table.
E. No computation will occur until enriched_itemized_orders_by_account is queried; upon query materialization, results will be calculated using the current valid version of data in each of the three tables referenced in the join logic.

Answer: B

Explanation:
This is the correct answer because it describes what will occur when this code is executed. The code uses three Delta Lake tables as input sources: accounts, orders, and order_items. These tables are joined together using SQL queries to create a view called new_enriched_itemized_orders_by_account, which contains information about each order item and its associated account details. Then, the code uses write.format(“delta”).mode(“overwrite”) to overwrite a target table called enriched_itemized_orders_by_account using the data from the view. This means that every time this code is executed, it will replace all existing data in the target table with new data based on the current valid version of data in each of the three input tables. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Write to Delta tables” section.

NEW QUESTION 47
Two of the most common data locations on Databricks are the DBFS root storage and external object storage mounted with dbutils.fs.mount().
Which of the following statements is correct?

A. DBFS is a file system protocol that allows users to interact with files stored in object storage using syntax and guarantees similar to Unix file systems.
B. By default, both the DBFS root and mounted data sources are only accessible to workspace administrators.
C. The DBFS root is the most secure location to store data, because mounted storage volumes must have full public read and write permissions.
D. Neither the DBFS root nor mounted storage can be accessed when using %sh in a Databricks notebook.
E. The DBFS root stores files in ephemeral block volumes attached to the driver, while mounted directories will always persist saved data to external storage between sessions.

Answer: A

Explanation:
DBFS is a file system protocol that allows users to interact with files stored in object storage using syntax and guarantees similar to Unix file systems1. DBFS is not a physical file system, but a layer over the object storage that provides a unified view of data across different data sources1. By default, the DBFS root is accessible to all users in the workspace, and the access to mounted data sources depends on the permissions of the storage account or container2. Mounted storage volumes do not need to have full public read and write permissions, but they do require a valid connection string or access key to be provided when mounting3. Both the DBFS root and mounted storage can be accessed when using %sh in a Databricks notebook, as long as the cluster has FUSE enabled4. The DBFS root does not store files in ephemeral block volumes attached to the driver, but in the object storage associated with the workspace1. Mounted directories will persist saved data to external storage between sessions, unless they are unmounted or deleted3. References: DBFS, Work with files on Azure Databricks, Mounting cloud object storage on Azure Databricks, Access DBFS with FUSE

NEW QUESTION 51
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 56
A nightly job ingests data into a Delta Lake table using the following code:

```
from pyspark.sql.functions import current_timestamp, input_file_name, col
from pyspark.sql.column import Column

def ingest_daily_batch(time_col: Column, year:int, month:int, day:int):
    (spark.read
        .format("parquet")
        .load(f"/mnt/daily_batch/{year}/{month}/{day}")
        .select("*",
                time_col.alias("ingest_time"),
                input_file_name().alias("source_file")
               )
        .write
        .mode("append")
        .saveAsTable("bronze")
    )
```

The next step in the pipeline requires a function that returns an object that can be used to manipulate new records that have not yet been processed to the next table in the pipeline.
Which code snippet completes this function definition? def new_records():

A. return spark.readStream.table("bronze")
B. return spark.readStream.load("bronze")C.
```
return (spark.read
          .table("bronze")
          .filter(col("ingest_time") == current_timestamp())
       )
```
D.return spark.read.option("readChangeFeed", "true").table ("bronze")
C.
```
return (spark.read
    .table("bronze")
    .filter(col("source_file") == f"/mnt/daily_batch/{year}/{month}/{day}")
)
```

Answer: E

Explanation:
https://docs.databricks.com/en/delta/delta-change-data-feed.html

NEW QUESTION 58
A junior data engineer on your team has implemented the following code block.

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

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

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

Answer: B

Explanation:
This is the correct answer because it describes what will happen with new records that have the same event_id as an existing record when the query is executed. The query uses the INSERT INTO command to append new records from the view new_events to the table events. However, the INSERT INTO command does not check for duplicate values in the primary key column (event_id) and does not perform any update or delete operations on existing records. Therefore, if there are new records that have the same event_id as an existing record, they will be ignored and not inserted into the table events. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Append data using INSERT INTO” section.
"If none of the WHEN MATCHED conditions evaluate to true for a source and target row pair that matches the merge_condition, then the target row is left unchanged." https://docs.databricks.com/en/sql/language-manual/delta-merge-into.html#:~:text=If%20none%20of%20the%20WHEN%20MATCHED%20conditions

%20evaluate%20to%20true%20for%20a%20source%20and%20target%20row%20pair%20that%20matches%20the%20merge_condition%2C%20then%20the%20target%20row%20is%20l eft%20unchanged.

NEW QUESTION 59
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 64
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 Kafkaproducer.
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. Ingestine 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 68
.....

Thank You for Trying Our Product

* 100% Pass or Money Back

All our products come with a 90-day Money Back Guarantee.

* One year free update

You can enjoy free update one year. 24x7 online support.

* Trusted by Millions

We currently serve more than 30,000,000 customers.

* Shop Securely

All transactions are protected by VeriSign!

100% Pass Your Databricks-Certified-Professional-Data-Engineer Exam with Our Prep Materials Via below:

https://www.certleader.com/Databricks-Certified-Professional-Data-Engineer-dumps.html