

Databricks

Exam Questions Databricks-Certified-Professional-Data-Engineer

Databricks Certified Data Engineer Professional Exam



NEW QUESTION 1

Review the following error traceback:

Which statement describes the error being raised?

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

Which strategy will yield the best performance without shuffling data?

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

Answer: B

Explanation:

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

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

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

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

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

References:

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

? Databricks Documentation on Data Sources: Databricks Data Sources Guide

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

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

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

Answer: A

Explanation:

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

conflicts and renaming the pipeline to create a new pipeline. The modified JSON file can then be used with the databricks pipelines create command to create a new pipeline with those settings. References:

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

NEW QUESTION 5

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 6

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

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

Answer: A

Explanation:

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

? <https://docs.databricks.com/notebooks/notebooks-use.html#use-regular-expressions>

? <https://docs.databricks.com/spark/latest/spark-sql/udf-scala.html#using-regular-expressions-in-udfs>

? https://docs.databricks.com/spark/latest/sparkr/functions/regexp_extract.html

? https://docs.databricks.com/spark/latest/sparkr/functions/regexp_replace.html

NEW QUESTION 7

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 8

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

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

- A. Improves the quality of your data
- B. Validates a complete use case of your application
- C. Troubleshooting is easier since all steps are isolated and tested individually
- D. Yields faster deployment and execution times
- E. Ensures that all steps interact correctly to achieve the desired end result

Answer: A

NEW QUESTION 9

A junior data engineer is migrating a workload from a relational database system to the Databricks Lakehouse. The source system uses a star schema, leveraging foreign key constraints and multi-table inserts to validate records on write.

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

- A. All Delta Lake transactions are ACID compliance against a single table, and Databricks does not enforce foreign key constraints.
- B. Databricks only allows foreign key constraints on hashed identifiers, which avoid collisions in highly-parallel writes.
- C. Foreign keys must reference a primary key field; multi-table inserts must leverage Delta Lake's upsert functionality.
- D. Committing to multiple tables simultaneously requires taking out multiple table locks and can lead to a state of deadlock.

Answer: A

Explanation:

In Databricks and Delta Lake, transactions are indeed ACID-compliant, but this compliance is limited to single table transactions. Delta Lake does not inherently enforce foreign key constraints, which are a staple in relational database systems for maintaining referential integrity between tables. This means that when migrating workloads from a relational database system to Databricks Lakehouse, engineers need to reconsider how to maintain data integrity and relationships that were previously enforced by foreign key constraints. Unlike traditional relational databases where foreign key constraints help in maintaining the consistency across tables, in Databricks Lakehouse, the data engineer has to manage data consistency and integrity at the application level or through careful design of ETL processes. References:

? Databricks Documentation on Delta Lake: Delta Lake Guide

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

NEW QUESTION 10

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 10

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 13

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 15

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

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

Which statement exemplifies best practices for implementing this system?

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

Answer: A

Explanation:

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

NEW QUESTION 16

The business reporting team requires that data for their dashboards be updated every hour. The total processing time for the pipeline that extracts transforms and load the data for their pipeline runs in 10 minutes.

Assuming normal operating conditions, which configuration will meet their service-level agreement requirements with the lowest cost?

- A. Schedule a job to execute the pipeline once an hour on a dedicated interactive cluster.
- B. Schedule a Structured Streaming job with a trigger interval of 60 minutes.
- C. Schedule a job to execute the pipeline once an hour on a new job cluster.
- D. Configure a job that executes every time new data lands in a given directory.

Answer: C

Explanation:

Scheduling a job to execute the data processing pipeline once an hour on a new job cluster is the most cost-effective solution given the scenario. Job clusters are ephemeral in nature; they are spun up just before the job execution and terminated upon completion, which means you only incur costs for the time the cluster is active. Since the total processing time is only 10 minutes, a new job cluster created for each hourly execution minimizes the running time and thus the cost, while also fulfilling the requirement for hourly data updates for the business reporting team's dashboards.

References:

? Databricks documentation on jobs and job clusters: <https://docs.databricks.com/jobs.html>

NEW QUESTION 21

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

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 25

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 27

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 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: model (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 33

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

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

Answer: D

Explanation:

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

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

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

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

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

References: Job clusters, Job retries, Maximum concurrent runs

NEW QUESTION 34

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

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

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

Answer: E

Explanation:

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

store_id INT, sales_date DATE, total_sales FLOAT

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

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

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

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

NEW QUESTION 35

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 40

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 44

The data engineering team maintains the following code:

```
import pyspark.sql.functions as F

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

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

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

Answer: C

Explanation:

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

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

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

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

NEW QUESTION 47

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 50

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 identity 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 53

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 54

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

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

Answer: B

Explanation:

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

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

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

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

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

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

NEW QUESTION 59

The marketing team is looking to share data in an aggregate table with the sales organization, but the field names used by the teams do not match, and a number of marketing specific fields have not been approved for the sales org.

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

- A. Create a view on the marketing table selecting only these fields approved for the sales team alias the names of any fields that should be standardized to the sales naming conventions.
- B. Use a CTAS statement to create a derivative table from the marketing table configure a production job to propagate changes.
- C. Add a parallel table write to the current production pipeline, updating a new sales table that varies as required from marketing table.
- D. Create a new table with the required schema and use Delta Lake's DEEP CLONE functionality to sync up changes committed to one table to the corresponding

table.

Answer: A

Explanation:

Creating a view is a straightforward solution that can address the need for field name standardization and selective field sharing between departments. A view allows for presenting a transformed version of the underlying data without duplicating it. In this scenario, the view would only include the approved fields for the sales team and rename any fields as per their naming conventions.

References:

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

NEW QUESTION 61

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 65

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 68

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

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

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

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

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

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

Answer: A

Explanation:

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

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

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

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

NEW QUESTION 73

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

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

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

Answer: A

Explanation:

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

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

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

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

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

References:

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

NEW QUESTION 75

A table is registered with the following code:

Both users and orders are Delta Lake tables. Which statement describes the results of querying recent_orders?

- A. All logic will execute at query time and return the result of joining the valid versions of the source tables at the time the query finishes.
- B. All logic will execute when the table is defined and store the result of joining tables to the DBFS; this stored data will be returned when the table is queried.
- C. Results will be computed and cached when the table is defined; these cached results will incrementally update as new records are inserted into source tables.
- D. All logic will execute at query time and return the result of joining the valid versions of the source tables at the time the query began.
- E. The versions of each source table will be stored in the table transaction log; query results will be saved to DBFS with each query.

Answer: B

NEW QUESTION 80

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 81

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 86

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 89

The data engineering team has configured a Databricks SQL query and alert to monitor the values in a Delta Lake table. The recent_sensor_recordings table contains an identifying sensor_id alongside the timestamp and temperature for the most recent 5 minutes of recordings.

The below query is used to create the alert:

```
SELECT MEAN(temperature), MAX(temperature), MIN(temperature)
FROM recent_sensor_recordings
GROUP BY sensor_id
```

The query is set to refresh each minute and always completes in less than 10 seconds. The alert is set to trigger when mean (temperature) > 120. Notifications are triggered to be sent at most every 1 minute.

If this alert raises notifications for 3 consecutive minutes and then stops, which statement must be true?

- A. The total average temperature across all sensors exceeded 120 on three consecutive executions of the query
- B. The recent_sensor_recordingstable was unresponsive for three consecutive runs of the query
- C. The source query failed to update properly for three consecutive minutes and then restarted
- D. The maximum temperature recording for at least one sensor exceeded 120 on three consecutive executions of the query
- E. The average temperature recordings for at least one sensor exceeded 120 on three consecutive executions of the query

Answer: E

Explanation:

This is the correct answer because the query is using a GROUP BY clause on the sensor_id column, which means it will calculate the mean temperature for each sensor separately. The alert will trigger when the mean temperature for any sensor is greater than 120, which means at least one sensor had an average temperature above 120 for three consecutive minutes. The alert will stop when the mean temperature for all sensors drops below 120. Verified References: [Databricks Certified Data Engineer Professional], under "SQL Analytics" section; Databricks Documentation, under "Alerts" section.

NEW QUESTION 91

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 93

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

street addresses as they appear in the customers table.

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

Which piece of information is critical to this decision?

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

Answer: A

Explanation:

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

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

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

NEW QUESTION 94

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

Which statement describes a limitation of Databricks Secrets?

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

Answer: E

Explanation:

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

NEW QUESTION 99

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

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

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

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

Answer: B

Explanation:

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

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

NEW QUESTION 102

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 104

.....

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](#)