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

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

Answer: A

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

NEW QUESTION 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. pyspsark.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
A Databricks job has been configured with 3 tasks, each of which is a Databricks notebook. Task A does not depend on other tasks. Tasks B and C run in parallel, with each having a serial dependency on task A.
If tasks A and B complete successfully but task C fails during a scheduled run, which statement describes the resulting state?

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

Answer: A

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

NEW QUESTION 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 constrains 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
The Databricks CLI is use to trigger a run of an existing job by passing the job_id parameter. The response that the job run request has been submitted successfully includes a filed run_id.
Which statement describes what the number alongside this field represents?

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

Answer: D

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

NEW QUESTION 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 15
A Structured Streaming job deployed to production has been experiencing delays during peak hours of the day. At present, during normal execution, each microbatch of data is processed in less than 3 seconds. During peak hours of the day, execution time for each microbatch becomes very inconsistent, sometimes exceeding 30 seconds. The streaming write is currently configured with a trigger interval of 10 seconds.
Holding all other variables constant and assuming records need to be processed in less than 10 seconds, which adjustment will meet the requirement?

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

Answer: E

Explanation:
The adjustment that will meet the requirement of processing records in less than 10 seconds is to decrease the trigger interval to 5 seconds. This is because triggering batches more frequently may prevent records from backing up and large batches from causing spill. Spill is a phenomenon where the data in memory exceeds the available capacity and has to be written to disk, which can slow down the processing and increase the execution time1. By reducing the trigger interval, the streaming query can process smaller batches of data more quickly and avoid spill. This can also improve the latency and throughput of the streaming job2.
The other options are not correct, because:
? Option A is incorrect because triggering batches more frequently does not allow idle executors to begin processing the next batch while longer running tasks from previous batches finish. In fact, the opposite is true. Triggering batches more frequently may cause concurrent batches to compete for the same resources and cause contention and backpressure2. This can degrade the performance and stability of the streaming job.
? Option B is incorrect because increasing the trigger interval to 30 seconds is not a good practice to ensure no records are dropped. Increasing the trigger interval means that the streaming query will process larger batches of data less frequently, which can increase the risk of spill, memory pressure, and timeouts12. This can also increase the latency and reduce the throughput of the streaming job.
? Option C is incorrect because the trigger interval can be modified without modifying the checkpoint directory. The checkpoint directory stores the metadata and state of the streaming query, such as the offsets, schema, and configuration3. Changing the trigger interval does not affect the state of the streaming query, and does not require a new checkpoint directory. However, changing the number of shuffle partitions may affect the state of the streaming query, and may require a new checkpoint directory4.
? Option D is incorrect because using the trigger once option and configuring a Databricks job to execute the query every 10 seconds does not ensure that all backlogged records are processed with each batch. The trigger once option means that the streaming query will process all the available data in the source and then stop5. However, this does not guarantee that the query will finish processing within 10 seconds, especially if there are a lot of records in the source. Moreover, configuring a Databricks job to execute the query every 10 seconds may cause overlapping or missed batches, depending on the execution time of the query.
References: Memory Management Overview, Structured Streaming Performance Tuning Guide, Checkpointing, Recovery Semantics after Changes in a Streaming Query, Triggers

NEW QUESTION 17
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 19
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 22
What statement is true regarding the retention of job run history?

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

Answer: C

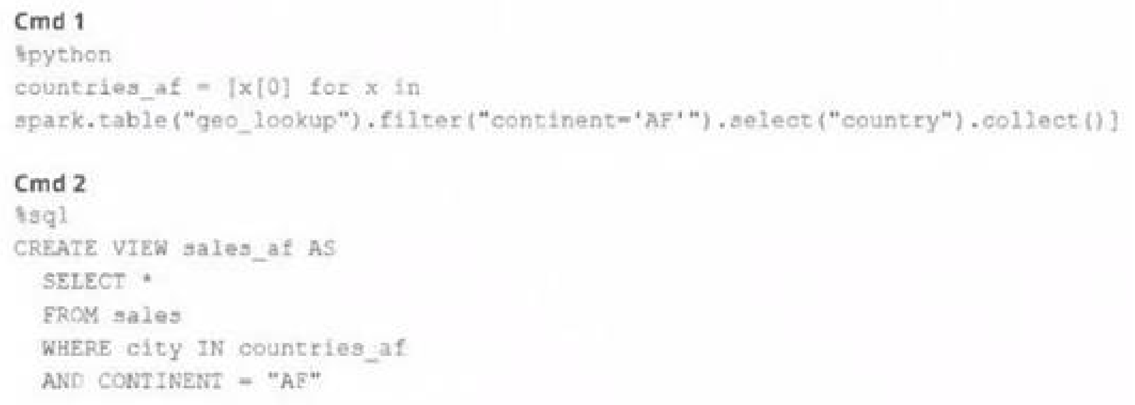
NEW QUESTION 27
A junior data engineer is working to implement logic for a Lakehouse table named silver_device_recordings. The source data contains 100 unique fields in a highly nested JSON structure.
The silver_device_recordings table will be used downstream for highly selective joins on a number of 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 30
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 succee
B. Executing show tables will show that countries at and sales at have been registered as views.
C. Cmd 1 will succee
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 fai
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 35
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.

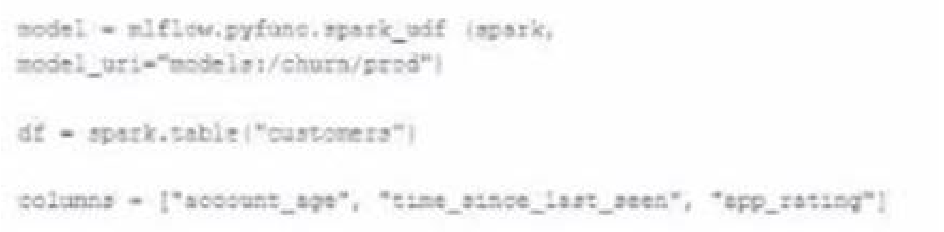
NEW QUESTION 40
The data governance team has instituted a requirement that all tables containing Personal Identifiable Information (PH) must be clearly annotated. This includes adding column comments, table comments, and setting the custom table property "contains_pii" = true.
The following SQL DDL statement 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 45
The data science team has created and logged a production using MLFlow. The model accepts a list of column names and returns a new column of type DOUBLE.
The following code correctly imports the production model, load the customer table containing the customer_id key column into a Dataframe, and defines the feature columns needed for the model.



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

df = spark.table("customers")

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

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

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

Answer: A

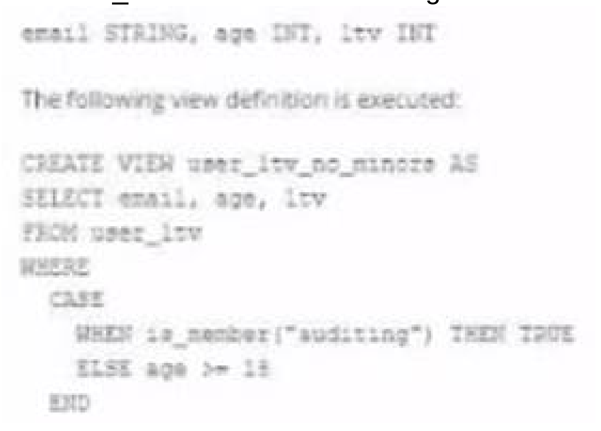
Explanation:
Given the information that the model is registered with MLflow and assuming predict is the method used to apply the model to a set of columns, we use the model.predict() function to apply the model to the DataFrame df using the specified columns. The model.predict() function is designed to take in a DataFrame and a list of column names as arguments, applying the trained model to these features to produce a predictions column. When working with PySpark, this predictions column needs to be selected alongside the customer_id to create a new DataFrame with the schema customer_id LONG, predictions DOUBLE.
References:
? MLflow documentation on using Python function models: https://www.mlflow.org/docs/latest/models.html#python-function-python
? PySpark MLlib documentation on model prediction: https://spark.apache.org/docs/latest/ml-pipeline.html#pipeline

NEW QUESTION 48
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 53
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 55
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.

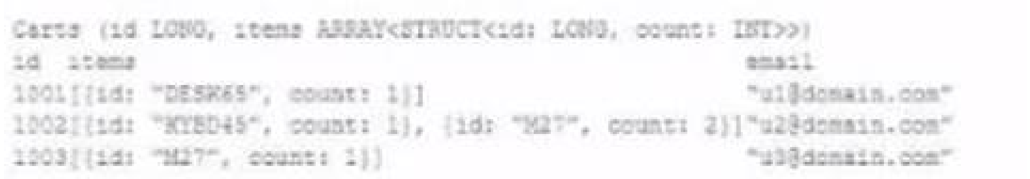
NEW QUESTION 57
A Data engineer wants to run unit’s tests using common Python testing frameworks on python functions defined across several Databricks notebooks currently used in production.
How can the data engineer run unit tests against function that work with data in production?

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

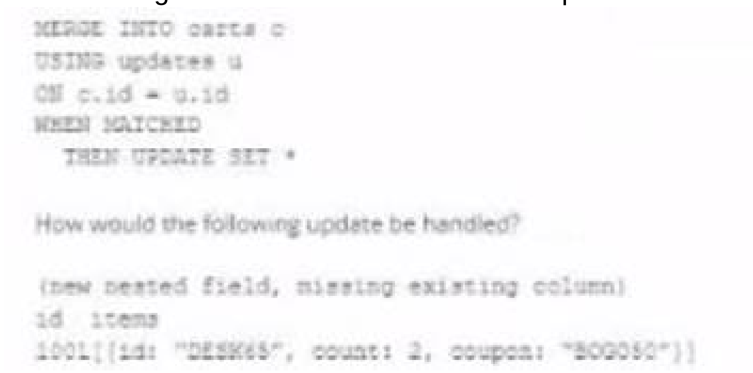
Explanation:
The best practice for running unit tests on functions that interact with data is to use a dataset that closely mirrors the production data. This approach allows data engineers to validate the logic of their functions without the risk of affecting the actual production data. It's important 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 61
The following table consists of items found in user carts within an e-commerce website.



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

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



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

How would the following update be handled?

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

How would the following update be handled?

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

Answer: D

Explanation:
With schema evolution enabled in Databricks Delta tables, when a new field is added to a record through a MERGE operation, Databricks automatically modifies the table schema to include the new field. In existing records where this new field is not present, Databricks will insert NULL values for that field. This ensures that the schema remains consistent across all records in the table, with the new field being present in every record, even if it is NULL for records that did not originally include it.
References:
? Databricks documentation on schema evolution in Delta Lake: https://docs.databricks.com/delta/delta-batch.html#schema-evolution

NEW QUESTION 65
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 67
A team of data engineer are adding tables to a DLT pipeline that contain repetitive expectations for many of the same data quality checks.
One member of the team suggests reusing these data quality rules across all tables defined for this pipeline.
What approach would allow them to do this?

A. Maintain data quality rules in a Delta table outside of this pipeline’s target schema, providing the schema name as a pipeline parameter.
B. Use global Python variables to make expectations visible across DLT notebooks included in the same pipeline.
C. Add data quality constraints to tables in this pipeline using an external job with access to pipeline configuration files.
D. Maintain data quality rules in a separate Databricks notebook that each DLT notebook of file.

Answer: A

Explanation:
Maintaining data quality rules in a centralized Delta table allows for the reuse of these rules across multiple DLT (Delta Live Tables) pipelines. By storing these rules outside the pipeline's target schema and referencing the schema name as a pipeline parameter, the team can apply the same set of data quality checks to different tables within the pipeline. This approach ensures consistency in data quality validations and reduces redundancy in code by not having to replicate the same rules in each DLT notebook or file. References:
? Databricks Documentation on Delta Live Tables: Delta Live Tables Guide

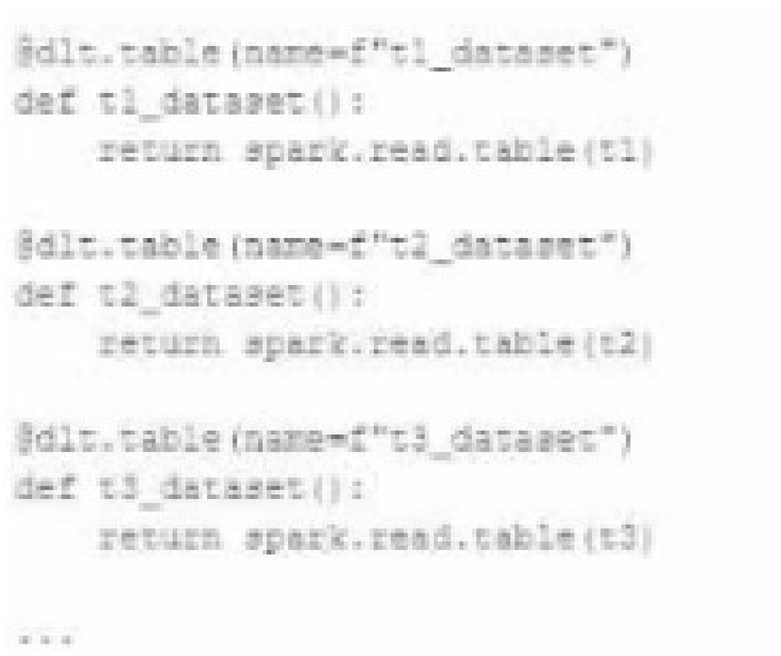
NEW QUESTION 68
Spill occurs as a result of executing various wide transformations. However, diagnosing spill requires one to proactively look for key indicators.
Where in the Spark UI are two of the primary indicators that a partition is spilling to disk?

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 73
A data engineer wants to reflector the following DLT code, which includes multiple definition with very similar code:



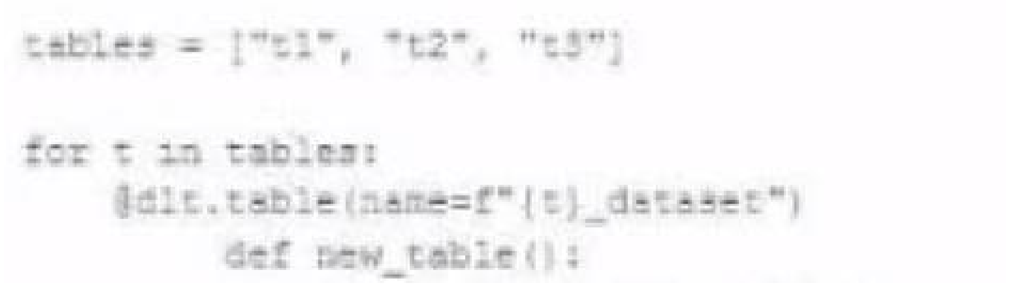
```
@dlt.table(name=f"t1_dataset")
def t1_dataset():
    return spark.read.table(t1)

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

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

...
```

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



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

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

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

A. Convert the list of configuration values to a dictionary of table settings, using table names as keys.
B. Convert the list of configuration values to a dictionary of table settings, using different input the for loop.
C. Load the configuration values for these tables from a separate file, located at a path provided by a pipeline parameter.
D. Wrap the loop inside another table definition, using generalized names and properties to replace with those from the inner table

Answer: A

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

NEW QUESTION 76
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 80
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 approval for the sales org.
Which of the following solutions addresses the situation while emphasizing simplicity?

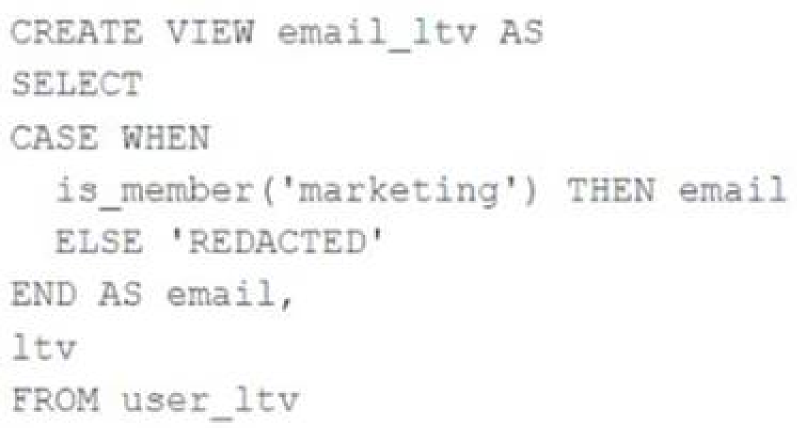
A. Create a view on the marketing table selecting only these fields approved for the sales team alias the names of any fields that should be standardized to the sales naming conventions.
B. Use a CTAS statement to create a derivative table from the marketing table configure a production jon to propagation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

Answer: A

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

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

NEW QUESTION 83
A table named user_ltv is being used to create a view that will be used by data analysts on various teams. Users in the workspace are configured into groups, which are used for setting up data access using ACLs.
The user_ltv table has the following schema:
email STRING, age INT, ltv INT
The following view definition is executed:



```
CREATE VIEW email_ltv AS
SELECT
CASE WHEN
  is_member('marketing') THEN email
  ELSE 'REDACTED'
END AS email,
ltv
FROM user_ltv
```

An analyst who is not a member of the marketing group executes the following query: SELECT * FROM email_ltv
Which statement describes the results returned by this query?

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

Answer: E

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

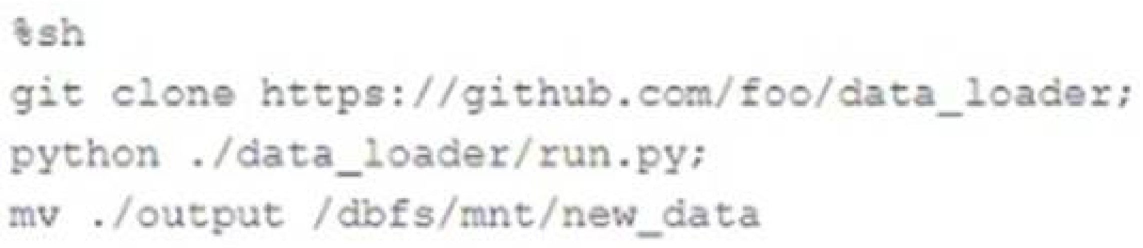
NEW QUESTION 88
An upstream system is emitting change data capture (CDC) logs that are being written to a cloud object storage directory. Each record in the log indicates the change type (insert, update, or delete) and the values for each field after the change. The source table has a primary key identified 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 93
The following code has been migrated to a Databricks notebook from a legacy workload:



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

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

A. %sh triggers a cluster restart to collect and install Gi

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

Answer: E

Explanation:
https://www.databricks.com/blog/2020/08/31/introducing-the-databricks-web- terminal.html
The code is using %sh to execute shell code on the driver node. This means that the code is not taking advantage of the worker nodes or Databricks optimized Spark. This is why the code is taking longer to execute. A better approach would be to use Databricks libraries and APIs to read and write data from Git and DBFS, and to leverage the parallelism and performance of Spark. For example, you can use the Databricks Connect feature to run your Python code on a remote Databricks cluster, or you can use the Spark Git Connector to read data from Git repositories as Spark DataFrames.

NEW QUESTION 97
A table in the Lakehouse named customer_churn_params 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.
The churn prediction model used by the ML team is fairly stable in production. The team is only interested in making predictions on records that have changed in the past 24 hours.
Which approach would simplify the identification of these changed records?

A. Apply the churn model to all rows in the customer_churn_params table, but implement logic to perform an upsert into the predictions table that ignores rows where predictions have not changed.
B. Convert the batch job to a Structured Streaming job using the complete output mode; configure a Structured Streaming job to read from the customer_churn_params table and incrementally predict against the churn model.
C. Calculate the difference between the previous model predictions and the current customer_churn_params on a key identifying unique customers before making new predictions; only make predictions on those customers not in the previous predictions.
D. Modify the overwrite logic to include a field populated by calling spark.sql.functions.current_timestamp() as data are being written; use this field to identify records written on a particular date.
E. Replace the current overwrite logic with a merge statement to modify only those records that have changed; write logic to make predictions on the changed records identified by the change data feed.

Answer: E

Explanation:
The approach that would simplify the identification of the changed records is to replace the current overwrite logic with a merge statement to modify only those records that have changed, and write logic to make predictions on the changed records identified by the change data feed. This approach leverages the Delta Lake features of merge and change data feed, which are designed to handle upserts and track row-level changes in a Delta table12. By using merge, the data engineering team can avoid overwriting the entire table every night, and only update or insert the records that have changed in the source data. By using change data feed, the ML team can easily access the change events that have occurred in the customer_churn_params table, and filter them by operation type (update or insert) and timestamp. This way, they can only make predictions on the records that have changed in the past 24 hours, and avoid re-processing the unchanged records. The other options are not as simple or efficient as the proposed approach, because:
? Option A would require applying the churn model to all rows in the customer_churn_params table, which would be wasteful and redundant. It would also require implementing logic to perform an upsert into the predictions table, which would be more complex than using the merge statement.
? Option B would require converting the batch job to a Structured Streaming job, which would involve changing the data ingestion and processing logic. It would also require using the complete output mode, which would output the entire result table every time there is a change in the source data, which would be inefficient and costly.
? Option C would require calculating the difference between the previous model predictions and the current customer_churn_params on a key identifying unique customers, which would be computationally expensive and prone to errors. It would also require storing and accessing the previous predictions, which would add extra storage and I/O costs.
? Option D would require modifying the overwrite logic to include a field populated by calling spark.sql.functions.current_timestamp() as data are being written, which would add extra complexity and overhead to the data engineering job. It would also require using this field to identify records written on a particular date, which would be less accurate and reliable than using the change data feed.
References: Merge, Change data feed

NEW QUESTION 101
The data science team has requested assistance in accelerating queries on free form text from user reviews. The data is currently stored in Parquet with the below schema:
item_id INT, user_id INT, review_id INT, rating FLOAT, review STRING
The review column contains the full text of the review left by the user. Specifically, the data science team is looking to identify if any of 30 key words exist in this field.
A junior data engineer suggests converting this data to Delta Lake will improve query performance.
Which response to the junior data engineer s suggestion is correct?

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

Answer: A

Explanation:
Converting the data to Delta Lake may not improve query performance on free text fields with high cardinality, such as the review column. This is because Delta Lake collects statistics on the minimum and maximum values of each column, which are not very useful for filtering or skipping data on free text fields. Moreover, Delta Lake collects statistics on the first 32 columns by default, which may not include the review column if the table has more columns. Therefore, the junior data engineer’s suggestion is not correct. A better approach would be to use a full-text search engine, such as Elasticsearch, to index and query the review column.

Alternatively, you can use natural language processing techniques, such as tokenization, stemming, and lemmatization, to preprocess the review column and create a new column with normalized terms that can be used for filtering or skipping data. References:
? Optimizations: https://docs.delta.io/latest/optimizations-oss.html
? Full-text search with Elasticsearch: https://docs.databricks.com/data/data- sources/elasticsearch.html
? Natural language processing: https://docs.databricks.com/applications/nlp/index.html

NEW QUESTION 103
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 105
Which Python variable contains a list of directories to be searched when trying to locate required modules?

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

Answer: B

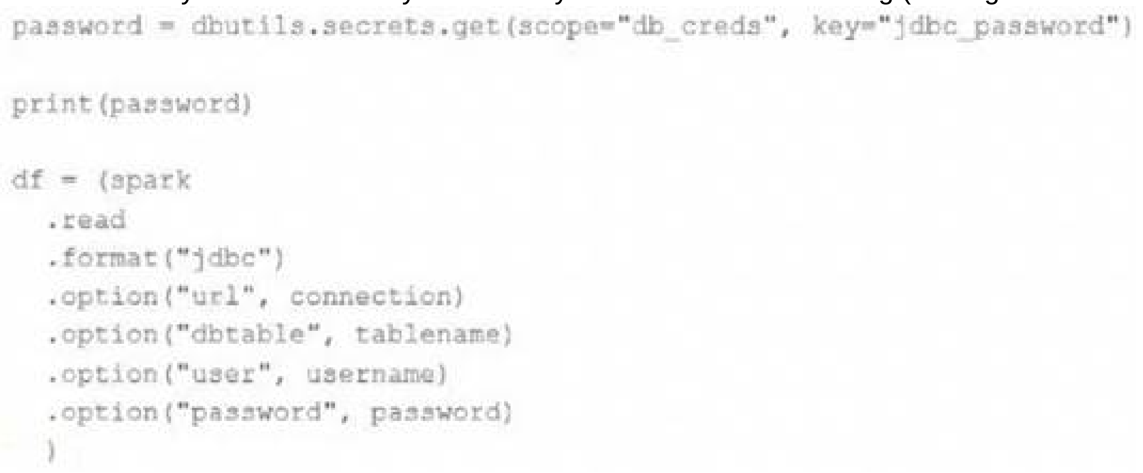
NEW QUESTION 107
Which statement describes integration testing?

A. Validates interactions between subsystems of your application
B. Requires an automated testing framework
C. Requires manual intervention
D. Validates an application use case
E. Validates behavior of individual elements of your application

Answer: D

Explanation:
This is the correct answer because it describes integration testing. Integration testing is a type of testing that validates interactions between subsystems of your application, such as modules, components, or services. Integration testing ensures that the subsystems work together as expected and produce the correct outputs or results. Integration testing can be done at different levels of granularity, such as component integration testing, system integration testing, or end-to-end testing. Integration testing can help detect errors or bugs that may not be found by unit testing, which only validates behavior of individual elements of your application. Verified References: [Databricks Certified Data Engineer Professional], under “Testing” section; Databricks Documentation, under “Integration testing” section.

NEW QUESTION 110
The security team is exploring whether or not the Databricks secrets module can be leveraged for connecting to an external database.
After testing the code with all Python variables being defined with strings, they upload the password to the secrets module and configure the correct permissions for the currently active user. They then modify their code to the following (leaving all other variables unchanged).



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

print(password)

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

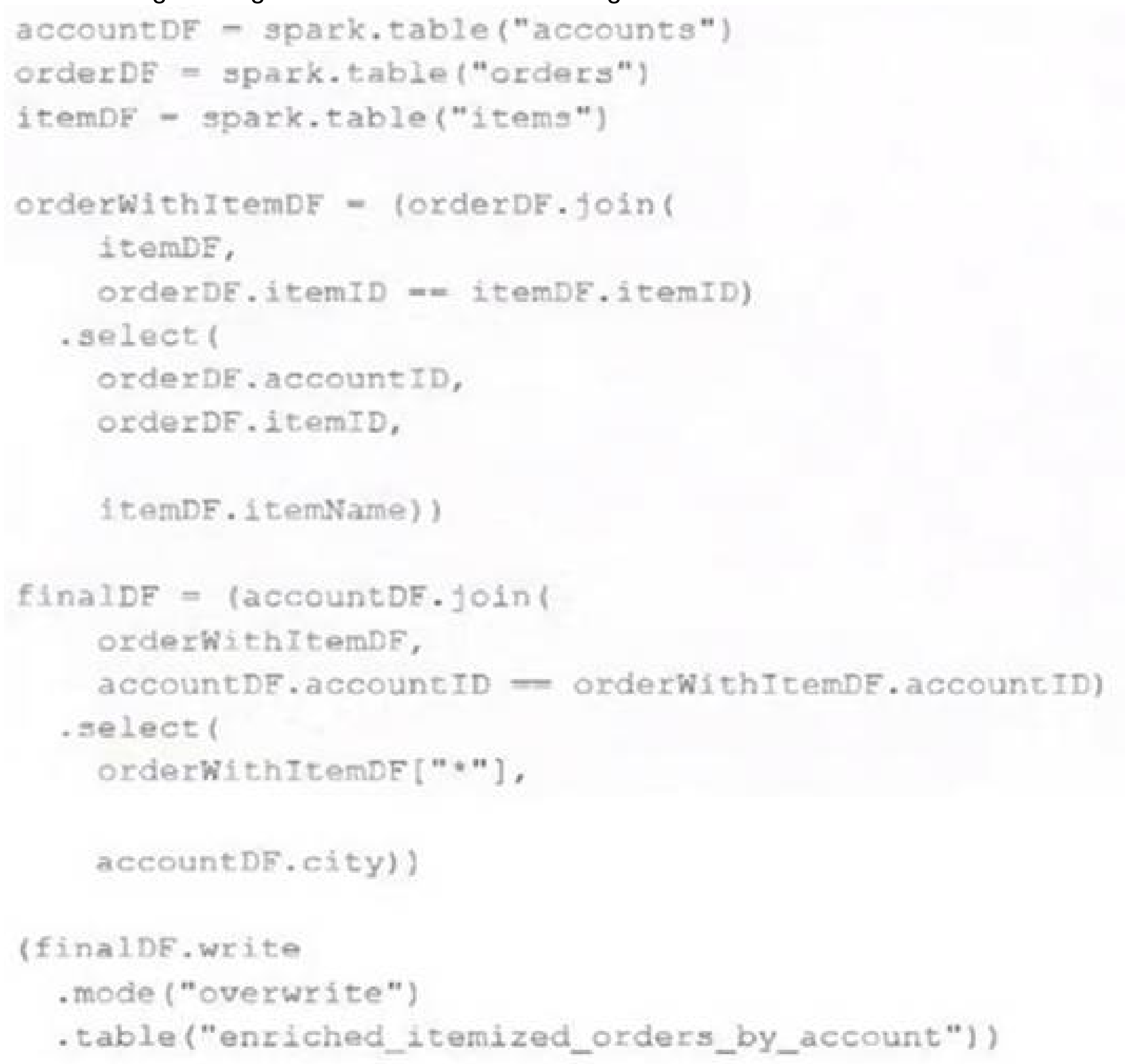
Which statement describes what will happen when the above code is executed?

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

Answer: E

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

NEW QUESTION 113
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 118
The data engineering team has configured a job to process customer requests to be forgotten (have their data deleted). All user data that needs to be deleted is stored in Delta Lake tables using default table settings.
The team has decided to process all deletions from the previous week as a batch job at 1am each Sunday. The total duration of this job is less than one hour.
Every Monday at 3am, a batch job executes a series of VACUUM commands on all Delta Lake tables throughout the organization.
The compliance officer has recently learned about Delta Lake's time travel functionality. They are concerned that this might allow continued access to deleted data.
Assuming all delete logic is correctly implemented, which statement correctly addresses this concern?

A. Because the vacuum command permanently deletes all files containing deleted records, deleted records may be accessible with time travel for around 24 hours.

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

Answer: E

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

NEW QUESTION 123
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 126
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 129
Which of the following is true of Delta Lake and the Lakehouse?

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

Answer: B

Explanation:
https://docs.delta.io/2.0.0/table-properties.html
Delta Lake automatically collects statistics on the first 32 columns of each table, which are leveraged in data skipping based on query filters1. Data skipping is a performance optimization technique that aims to avoid reading irrelevant data from the storage layer1. By collecting statistics such as min/max values, null counts, and bloom filters, Delta Lake can efficiently prune unnecessary files or partitions from the query plan1. This can significantly improve the query performance and reduce the I/O cost.
The other options are false because:
? Parquet compresses data column by column, not row by row2. This allows for better compression ratios, especially for repeated or similar values within a column2.
? Views in the Lakehouse do not maintain a valid cache of the most recent versions of source tables at all times3. Views are logical constructs that are defined by a SQL query on one or more base tables3. Views are not materialized by default, which means they do not store any data, but only the query definition3.

Therefore, views always reflect the latest state of the source tables when queried3. However, views can be cached manually using the CACHE TABLE or CREATE TABLE AS SELECT commands.
? Primary and foreign key constraints can not be leveraged to ensure duplicate values are never entered into a dimension table. Delta Lake does not support enforcing primary and foreign key constraints on tables. Constraints are logical rules that define the integrity and validity of the data in a table. Delta Lake relies on the application logic or the user to ensure the data quality and consistency.
? Z-order can be applied to any values stored in Delta Lake tables, not only numeric values. Z-order is a technique to optimize the layout of the data files by sorting them on one or more columns. Z-order can improve the query performance by clustering related values together and enabling more efficient data skipping. Z-order can be applied to any column that has a defined ordering, such as numeric, string, date, or boolean values.
References: Data Skipping, Parquet Format, Views, [Caching], [Constraints], [Z-Ordering]

NEW QUESTION 132
A small company based in the United States has recently contracted a consulting firm in India to implement several new data engineering pipelines to power artificial intelligence applications. All the company's data is stored in regional cloud storage in the United States.
The workspace administrator at the company is uncertain about where the Databricks workspace used by the contractors should be deployed.
Assuming that all data governance considerations are accounted for, which statement accurately informs this decision?

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

Answer: C

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

NEW QUESTION 137
......