Databricks

Exam Questions Databricks-Certified-Professional-Data-Engineer

Databricks Certified Data Engineer Professional Exam

NEW QUESTION 1
An upstream source writes Parquet data as hourly batches to directories named with the current date. A nightly batch job runs the following code to ingest all data from the previous day as indicated by the date variable:

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

Assume that the fields customer_id and order_id serve as a composite key to uniquely identify each order.
If the upstream system is known to occasionally produce duplicate entries for a single order
hours apart, which statement is correct?

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

Answer: B

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

NEW QUESTION 2
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 3
A user new to Databricks is trying to troubleshoot long execution times for some pipeline logic they are working on. Presently, the user is executing code cell-by-cell, using display() calls to confirm code is producing the logically correct results as new transformations are added to an operation. To get a measure of average time to execute, the user is running each cell multiple times interactively.
Which of the following adjustments will get a more accurate measure of how code is likely to perform in production?

A. Scala is the only language that can be accurately tested using interactive notebooks; because the best performance is achieved by using Scala code compiled to JAR
B. all PySpark and Spark SQL logic should be refactored.
C. The only way to meaningfully troubleshoot code execution times in development notebooks Is to use production-sized data and production-sized clusters with Run All execution.
D. Production code development should only be done using an IDE; executing code against a local build of open source Spark and Delta Lake will provide the most accurate benchmarks for how code will perform in production.
E. Calling display () forces a job to trigger, while many transformations will only add to the logical query plan; because of caching, repeated execution of the same logic does not provide meaningful results.
F. The Jobs Ul should be leveraged to occasionally run the notebook as a job and track execution time during incremental code development because Photon can only be enabled on clusters launched for scheduled jobs.

Answer: D

Explanation:
In Databricks notebooks, using the display() function triggers an action that forces Spark to execute the code and produce a result. However, Spark operations are generally divided into transformations and actions. Transformations create a new dataset from an existing one and are lazy, meaning they are not computed immediately but added to a logical plan. Actions, like display(), trigger the execution of this logical plan. Repeatedly running the same code cell can lead to misleading performance measurements due to caching. When a dataset is used multiple times, Spark's optimization mechanism caches it in memory, making

subsequent executions faster. This behavior does not accurately represent the first-time execution performance in a production environment where data might not be cached yet.
To get a more realistic measure of performance, it is recommended to:
? Clear the cache or restart the cluster to avoid the effects of caching.
? Test the entire workflow end-to-end rather than cell-by-cell to understand the cumulative performance.
? Consider using a representative sample of the production data, ensuring it includes various cases the code will encounter in production.
References:
? Databricks Documentation on Performance Optimization: Databricks Performance Tuning
? Apache Spark Documentation: RDD Programming Guide - Understanding transformations and actions

NEW QUESTION 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
The data architect has mandated that all tables in the Lakehouse should be configured as external Delta Lake tables.
Which approach will ensure that this requirement is met?

A. Whenever a database is being created, make sure that the location keyword is used
B. When configuring an external data warehouse for all table storag
C. leverage Databricks for all ELT.
D. Whenever a table is being created, make sure that the location keyword is used.
E. When tables are created, make sure that the external keyword is used in the create table statement.
F. When the workspace is being configured, make sure that external cloud object storage has been mounted.

Answer: C

Explanation:
This is the correct answer because it ensures that this requirement is met. The requirement is that all tables in the Lakehouse should be configured as external Delta Lake tables. An external table 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 by using the location keyword to specify the path to an existing directory in a cloud storage system, such as DBFS or S3. By creating external tables, the data engineering team can avoid losing data if they drop or overwrite the table, as well as leverage existing data without moving or copying it.
Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Create an external table” section.

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

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

Answer: A

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

NEW QUESTION 8
A Delta Lake table was created with the below query:
Realizing that the original query had a typographical error, the below code was executed: ALTER TABLE prod.sales_by_stor RENAME TO prod.sales_by_store
Which result will occur after running the second command?

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

Answer: A

Explanation:
The query uses the CREATE TABLE USING DELTA syntax to create a Delta Lake table from an existing Parquet file stored in DBFS. The query also uses the LOCATION keyword to specify the path to the Parquet file as /mnt/finance_eda_bucket/tx_sales.parquet. By using the LOCATION keyword, the query creates an external table, which is a table that is stored outside of the default warehouse directory and whose metadata is not managed by Databricks. An external table can be created from an existing directory in a cloud storage system, such as DBFS or S3, that contains data files in a supported format, such as Parquet or CSV.
The result that will occur after running the second command is that the table reference in the metastore is updated and no data is changed. The metastore is a service that stores metadata about tables, such as their schema, location, properties, and partitions. The metastore allows users to access tables using SQL commands or Spark APIs without knowing their physical location or format. When renaming an external table using the ALTER TABLE RENAME TO command, only the table reference in the metastore is updated with the new name; no data files or directories are moved or changed in the storage system. The table will still point to the same location and use the same format as before. However, if renaming a managed table, which is a table whose metadata and data are both managed by Databricks, both the table reference in the metastore and the data files in the default warehouse directory are moved and renamed accordingly.
Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “ALTER TABLE RENAME TO” section; Databricks Documentation, under “Metastore” section; Databricks Documentation, under “Managed and external tables” section.

NEW QUESTION 9
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
An hourly batch job is configured to ingest data files from a cloud object storage container where each batch represent all records produced by the source system in a given hour. The batch job to process these records into the Lakehouse is sufficiently delayed to ensure no late-arriving data is missed. The user_id field represents a unique key for the data, which has the following schema:
user_id BIGINT, username STRING, user_utc STRING, user_region STRING, last_login BIGINT, auto_pay BOOLEAN, last_updated BIGINT
New records are all ingested into a table named account_history which maintains a full record of all data in the same schema as the source. The next table in the system is named account_current and is implemented as a Type 1 table representing the most recent value for each unique user_id.
Assuming there are millions of user accounts and tens of thousands of records processed hourly, which implementation can be used to efficiently update the described account_current table as part of each hourly batch job?

A. Use Auto Loader to subscribe to new files in the account history directory; configure a Structured Streaminq trigger once job to batch update newly detected files into the account current table.
B. Overwrite the account current table with each batch using the results of a query against the account history table grouping by user id and filtering for the max value of last updated.
C. Filter records in account history using the last updated field and the most recent hour processed, as well as the max last iogin by user id write a merge statement to update or insert the most recent value for each user id.
D. Use Delta Lake version history to get the difference between the latest version of account history and one version prior, then write these records to account current.
E. Filter records in account history using the last updated field and the most recent hour processed, making sure to deduplicate on username; write a merge statement to update orinsert themost recent value for each username.

Answer: C

Explanation:

This is the correct answer because it efficiently updates the account current table with only the most recent value for each user id. The code filters records in account history using the last updated field and the most recent hour processed, which means it will only process the latest batch of data. It also filters by the max last login by user id, which means it will only keep the most recent record for each user id within that batch. Then, it writes a merge statement to update or insert the most recent value for each user id into account current, which means it will perform an upsert operation based on the user id column. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Upsert into a table using merge” section.

NEW QUESTION 10
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 11
When evaluating the Ganglia Metrics for a given cluster with 3 executor nodes, which indicator would signal proper utilization of the VM's resources?

A. The five Minute Load Average remains consistent/flat
B. Bytes Received never exceeds 80 million bytes per second
C. Network I/O never spikes
D. Total Disk Space remains constant
E. CPU Utilization is around 75%

Answer: E

Explanation:
In the context of cluster performance and resource utilization, a CPU utilization rate of around 75% is generally considered a good indicator of efficient resource usage. This level of CPU utilization suggests that the cluster is being effectively used without being overburdened or underutilized.
? A consistent 75% CPU utilization indicates that the cluster's processing power is being effectively employed while leaving some headroom to handle spikes in workload or additional tasks without maxing out the CPU, which could lead to performance degradation.
? A five Minute Load Average that remains consistent/flat (Option A) might indicate underutilization or a bottleneck elsewhere.
? Monitoring network I/O (Options B and C) is important, but these metrics alone don't provide a complete picture of resource utilization efficiency.
? Total Disk Space (Option D) remaining constant is not necessarily an indicator of proper resource utilization, as it's more related to storage rather than computational efficiency.
References:
? Ganglia Monitoring System: Ganglia Documentation
? Databricks Documentation on Monitoring: Databricks Cluster Monitoring

NEW QUESTION 12
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 15
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 20
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 22
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 24
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 29
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 30
A distributed team of data analysts share computing resources on an interactive cluster with autoscaling configured. In order to better manage costs and query throughput, the workspace administrator is hoping to evaluate whether cluster upscaling is caused by many concurrent users or resource-intensive queries.
In which location can one review the timeline for cluster resizing events?

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

Answer: C

NEW QUESTION 34
A CHECK constraint has been successfully added to the Delta table named activity_details using the following logic:
A batch job is attempting to insert new records to the table, including a record where latitude = 45.50 and longitude = 212.67.
Which statement describes the outcome of this batch insert?

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

Answer: B

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

NEW QUESTION 37
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 39
A production workload incrementally applies updates from an external Change Data Capture feed to a Delta Lake table as an always-on Structured Stream job. When data was initially migrated for this table, OPTIMIZE was executed and most data files were resized to 1 GB. Auto Optimize and Auto Compaction were both turned on for the streaming production job. Recent review of data files shows that most data files are under 64 MB, although each partition in the table contains at least 1 GB of data and the total table size is over 10 TB.
Which of the following likely explains these smaller file sizes?

A. Databricks has autotuned to a smaller target file size to reduce duration of MERGE operations
B. Z-order indices calculated on the table are preventing file compactionC Bloom filler indices calculated on the table are preventing file compaction
C. Databricks has autotuned to a smaller target file size based on the overall size of data in the table
D. Databricks has autotuned to a smaller target file size based on the amount of data in each partition

Answer: A

Explanation:
This is the correct answer because Databricks has a feature called Auto Optimize, which automatically optimizes the layout of Delta Lake tables by coalescing small files into larger ones and sorting data within each file by a specified column. However, Auto Optimize also considers the trade-off between file size and merge performance, and may choose a smaller target file size to reduce the duration of merge operations, especially for streaming workloads that frequently update existing records. Therefore, it is possible that Auto Optimize has autotuned to a smaller target file size based on the characteristics of the streaming production job. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Auto Optimize” section. https://docs.databricks.com/en/delta/tune-file-size.html#autotune-table 'Autotune file size based on workload'

NEW QUESTION 44
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 45
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 49
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 54
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: "KYBD49", 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 55
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 59
The DevOps team has configured a production workload as a collection of notebooks scheduled to run daily using the Jobs UI. A new data engineering hire is onboarding to the team and has requested access to one of these notebooks to review the production logic.
What are the maximum notebook permissions that can be granted to the user without allowing accidental changes to production code or data?

A. Can manage
B. Can edit
C. Can run
D. Can Read

Answer: D

Explanation:
Granting a user 'Can Read' permissions on a notebook within Databricks allows them to view the notebook's content without the ability to execute or edit it. This level of permission ensures that the new team member can review the production logic for learning or auditing purposes without the risk of altering the notebook's code or affecting production data and workflows. This approach aligns with best practices for maintaining security and integrity in production environments, where strict access controls are essential to prevent unintended modifications.References: Databricks documentation on access control and permissions for notebooks within the workspace (https://docs.databricks.com/security/access-control/workspace-acl.html).

NEW QUESTION 64
Which statement regarding spark configuration on the Databricks platform is true?

A. Spark configuration properties set for an interactive cluster with the Clusters UI will impact all notebooks attached to that cluster.
B. When the same spar configuration property is set for an interactive to the same interactive cluster.
C. Spark configuration set within an notebook will affect all SparkSession attached to the same interactive cluster
D. The Databricks REST API can be used to modify the Spark configuration properties for an interactive cluster without interrupting jobs.

Answer: A

Explanation:
When Spark configuration properties are set for an interactive cluster using the Clusters UI in Databricks, those configurations are applied at the cluster level. This means that all notebooks attached to that cluster will inherit and be affected by these configurations. This approach ensures consistency across all executions within that cluster, as the Spark configuration properties dictate aspects such as memory allocation, number of executors, and other vital execution parameters. This centralized configuration management helps maintain standardized execution environments across different notebooks, aiding in debugging and performance optimization.
References:
? Databricks documentation on configuring clusters: https://docs.databricks.com/clusters/configure.html

NEW QUESTION 66
A member of the data engineering team has submitted a short notebook that they wish to schedule as part of a larger data pipeline. Assume that the commands provided below produce the logically correct results when run as presented.

```
Cmd 1

rawDF = spark.table("raw_data")

Cmd 2

rawDF.printSchema()

Cmd 3

flattenedDF = rawDF.select("*", "values.*")

Cmd 4

finalDF = flattenedDF.drop("values")

Cmd 5

display(finalDF)

Cmd 6

finalDF.write.mode("append").saveAsTable("flat_data")
```

Which command should be removed from the notebook before scheduling it as a job?

A. Cmd 2
B. Cmd 3
C. Cmd 4
D. Cmd 5
E. Cmd 6

Answer: E

Explanation:
Cmd 6 is the command that should be removed from the notebook before scheduling it as a job. This command is selecting all the columns from the finalDF

dataframe and displaying them in the notebook. This is not necessary for the job, as the finalDF dataframe is already written to a table in Cmd 7. Displaying the dataframe in the notebook will only consume resources and time, and it will not affect the output of the job. Therefore, Cmd 6 is redundant and should be removed.
The other commands are essential for the job, as they perform the following tasks:
? Cmd 1: Reads the raw_data table into a Spark dataframe called rawDF.
? Cmd 2: Prints the schema of the rawDF dataframe, which is useful for debugging and understanding the data structure.
? Cmd 3: Selects all the columns from the rawDF dataframe, as well as the nested columns from the values struct column, and creates a new dataframe called flattenedDF.
? Cmd 4: Drops the values column from the flattenedDF dataframe, as it is no longer needed after flattening, and creates a new dataframe called finalDF.
? Cmd 5: Explains the physical plan of the finalDF dataframe, which is useful for optimizing and tuning the performance of the job.
? Cmd 7: Writes the finalDF dataframe to a table called flat_data, using the append mode to add new data to the existing table.

NEW QUESTION 67
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 72
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 74
A data pipeline uses Structured Streaming to ingest data from kafka to Delta Lake. Data is being stored in a bronze table, and includes the Kafka_generated timesamp, key, and value. Three months after the pipeline is deployed the data engineering team has noticed some latency issued during certain times of the day.
A senior data engineer updates the Delta Table's schema and ingestion logic to include the current timestamp (as recoded by Apache Spark) as well the Kafka topic and partition. The team plans to use the additional metadata fields to diagnose the transient processing delays:
Which limitation will the team face while diagnosing this problem?

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

Answer: A

Explanation:
When adding new fields to a Delta table's schema, these fields will not be retrospectively applied to historical records that were ingested before the schema change. Consequently, while the team can use the new metadata fields to investigate transient processing delays moving forward, they will be unable to apply this diagnostic approach to past data that lacks these fields.
References:
? Databricks documentation on Delta Lake schema management: https://docs.databricks.com/delta/delta-batch.html#schema-management

NEW QUESTION 79
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 84
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 88
The data governance team is reviewing code used for deleting records for compliance with GDPR. They note the following logic is used to delete records from the Delta Lake table named users.

```
DELETE FROM users
WHERE user_id IN
   (SELECT user_id FROM delete_requests)
```

Assuming that user_id is a unique identifying key and that delete_requests contains all users that have requested deletion, which statement describes whether successfully executing the above logic guarantees that the records to be deleted are no longer accessible and why?

A. Yes; Delta Lake ACID guarantees provide assurance that the delete command succeeded fully and permanently purged these records.
B. No; the Delta cache may return records from previous versions of the table until the cluster is restarted.
C. Yes; the Delta cache immediately updates to reflect the latest data files recorded to disk.
D. No; the Delta Lake delete command only provides ACID guarantees when combined with the merge into command.
E. No; files containing deleted records may still be accessible with time travel until a vacuum command is used to remove invalidated data files.

Answer: E

Explanation:
The code uses the DELETE FROM command to delete records from the users table that match a condition based on a join with another table called delete_requests, which contains all users that have requested deletion. The DELETE FROM command deletes records from a Delta Lake table by creating a new version of the table that does not contain the deleted records. However, this does not guarantee that the records to be deleted are no longer accessible, because Delta Lake supports time travel, which allows querying previous versions of the table using a timestamp or version number. Therefore, files containing deleted records may still be accessible with time travel until a vacuum command is used to remove invalidated data files from physical storage. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Delete from a table” section; Databricks Documentation, under “Remove files no longer referenced by a Delta table” section.

NEW QUESTION 91
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 95
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 99
Which distribution does Databricks support for installing custom Python code packages?

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

Answer: D

NEW QUESTION 102
A nightly job ingests data into a Delta Lake table using the following code:

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

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

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

A. return spark.readStream.table("bronze")
B. return spark.readStream.load("bronze")C.
```
return (spark.read
        .table("bronze")
        .filter(col("ingest_time") == current_timestamp())
)
```
D.return spark.read.option("readChangeFeed", "true").table ("bronze")
C.

```
return (spark.read
    .table("bronze")
    .filter(col("source_file") == f"/mnt/daily_batch/{year}/{month}/{day}")
)
```

Answer: E

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

NEW QUESTION 107
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 111
A junior data engineer on your team has implemented the following code block.

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

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

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

Answer: B

Explanation:
This is the correct answer because it describes what will happen with new records that have the same event_id as an existing record when the query is executed. The query uses the INSERT INTO command to append new records from the view new_events to the table events. However, the INSERT INTO command does not check for duplicate values in the primary key column (event_id) and does not perform any update or delete operations on existing records. Therefore, if there are new records that have the same event_id as an existing record, they will be ignored and not inserted into the table events. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Append data using INSERT INTO” section.
"If none of the WHEN MATCHED conditions evaluate to true for a source and target row pair that matches the merge_condition, then the target row is left unchanged." https://docs.databricks.com/en/sql/language-manual/delta-merge-into.html#:~:text=If%20none%20of%20the%20WHEN%20MATCHED%20conditions%20evaluate%20to%20true%20for%20a%20source%20and%20target%20row%20pair%20that%20matches%20the%20merge_condition%2C%20then%20the%20target%20row%20is%20l eft%20unchanged.

NEW QUESTION 113
Which is a key benefit of an end-to-end test?

A. It closely simulates real world usage of your application.
B. It pinpoint errors in the building blocks of your application.
C. It provides testing coverage for all code paths and branches.
D. It makes it easier to automate your test suite

Answer: A

Explanation:

End-to-end testing is a methodology used to test whether the flow of an application, from start to finish, behaves as expected. The key benefit of an end-to-end test is that it closely simulates real-world, user behavior, ensuring that the system as a whole operates correctly.
References:
? Software Testing: End-to-End Testing

NEW QUESTION 116
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 lo
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 120
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 122
Which statement describes Delta Lake Auto Compaction?

A. An asynchronous job runs after the write completes to detect if files could be further compacted; if yes, an optimize job is executed toward a default of 1 GB.
B. Before a Jobs cluster terminates, optimize is executed on all tables modified during the most recent job.
C. Optimized writes use logical partitions instead of directory partitions; because partition boundaries are only represented in metadata, fewer small files are written.
D. Data is queued in a messaging bus instead of committing data directly to memory; all data is committed from the messaging bus in one batch once the job is complete.
E. An asynchronous job runs after the write completes to detect if files could be further compacted; if yes, an optimize job is executed toward a default of 128 MB.

Answer: E

Explanation:
This is the correct answer because it describes the behavior of Delta Lake Auto Compaction, which is a feature that automatically optimizes the layout of Delta Lake tables by coalescing small files into larger ones. Auto Compaction runs as an asynchronous job after a write to a table has succeeded and checks if files within a partition can be further compacted. If yes, it runs an optimize job with a default target file size of 128 MB. Auto Compaction only compacts files that have not been compacted previously. Verified References: [Databricks Certified Data Engineer Professional], under “Delta Lake” section; Databricks Documentation, under “Auto Compaction for Delta Lake on Databricks” section.
"Auto compaction occurs after a write to a table has succeeded and runs synchronously on the cluster that has performed the write. Auto compaction only compacts files that haven’t been compacted previously."
https://learn.microsoft.com/en-us/azure/databricks/delta/tune-file-size

NEW QUESTION 123
All records from an Apache Kafka producer are being ingested into a single Delta Lake table with the following schema:

key BINARY, value BINARY, topic STRING, partition LONG, offset LONG, timestamp LONG
There are 5 unique topics being ingested. Only the "registration" topic contains Personal Identifiable Information (PII). The company wishes to restrict access to PII. The company also wishes to only retain records containing PII in this table for 14 days after initial ingestion. However, for non-PII information, it would like to retain these records indefinitely.
Which of the following solutions meets the requirements?

A. All data should be deleted biweekly; Delta Lake's time travel functionality should be leveraged to maintain a history of non-PII information.
B. Data should be partitioned by the registration field, allowing ACLs and delete statements to be set for the PII directory.
C. Because the value field is stored as binary data, this information is not considered PII and no special precautions should be taken.
D. Separate object storage containers should be specified based on the partition field, allowing isolation at the storage level.
E. Data should be partitioned by the topic field, allowing ACLs and delete statements to leverage partition boundaries.

Answer: B

Explanation:
Partitioning the data by the topic field allows the company to apply different access control policies and retention policies for different topics. For example, the company can use the Table Access Control feature to grant or revoke permissions to the registration topic based on user roles or groups. The company can also use the DELETE command to remove records from the registration topic that are older than 14 days, while keeping the records from other topics indefinitely. Partitioning by the topic field also improves the performance of queries that filter by the topic field, as they can skip reading irrelevant partitions. References:
? Table Access Control: https://docs.databricks.com/security/access-control/table-acls/index.html
? DELETE: https://docs.databricks.com/delta/delta-update.html#delete-from-a-table

NEW QUESTION 127
.....

Thank You for Trying Our Product

We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questons 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