Databricks.Databricks-Certified-Professional-Data-Engineer.v2026-11-02.q167

Exam Code:	Databricks-Certified-Professional-Data-Engineer
Exam Name:	Databricks Certified Professional Data Engineer Exam
Certification Provider:	Databricks
Free Question Number:	167
Version:	v2026-11-02
# of views:	153
# of Questions views:	1670
https://www.exam-tests.com/Databricks-Certified-Professional-Data-Engineer-exam/Databricks.Databricks-Certified-Professional-Data-Engineer.v2026-11-02.q167.html	

NEW QUESTION: 1

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



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.option("readChangeFeed", "true").table ("bronze")

D.

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

Answer: D (LEAVE A REPLY)

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

NEW QUESTION: 2

You noticed a colleague is manually copying the data to the backup folder prior to running an up-date command, incase if the update command did not provide the expected outcome so he can use the backup copy to replace table, which Delta Lake feature would you recommend simplifying the process?

A. Use time travel feature to refer old data instead of manually copying
B. Use DEEP CLONE to clone the table prior to update to make a backup copy
C. Use SHADOW copy of the table as preferred backup choice
D. Cloud object storage retains previous version of the file
E. Cloud object storage automatically backups the data
Answer: A (LEAVE A REPLY)
Explanation
The answer is, Use time travel feature to refer old data instead of manually copying.
https://databricks.com/blog/2019/02/04/introducing-delta-time-travel-for-large-scale-data-lakes.html
1.SELECT count(*) FROM my_table TIMESTAMP AS OF "2019-01-01"
2.SELECT count(*) FROM my_table TIMESTAMP AS OF date_sub(current_date(), 1)
3.SELECT count(*) FROM my_table TIMESTAMP AS OF "2019-01-01 01:30:00.000"

NEW QUESTION: 3
A data engineer needs to dynamically create a table name string using three Python varia-bles: region, store, and year. An example of a table name is below when region = "nyc", store = "100", and year = "2021":
nyc100_sales_2021
Which of the following commands should the data engineer use to construct the table name in Py-thon?
A. "{region}+{store}+_sales_+{year}"
B. "{region}+{store}+"_sales_"+{year}"
C. "{region}{store}_sales_{year}"
D. f"{region}{store}_sales_{year}"
E. f"{region}+{store}+_sales_+{year}"
Answer: D (LEAVE A REPLY)

NEW QUESTION: 4
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 namedusers.

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

Assuming thatuser_idis a unique identifying key and thatdelete_requestscontains 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: (SHOW ANSWER)
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: 5
The downstream consumers of a Delta Lake table have been complaining about data quality issues impacting performance in their applications. Specifically, they have complained that invalid latitude and longitude values in the activity_details table have been breaking their ability to use other geolocation processes.
A junior engineer has written the following code to add CHECK constraints to the Delta Lake table:

```
ALTER TABLE activity_details
ADD CONSTRAINT valid_coordinates
CHECK (
  latitude >= -90 AND
  latitude <= 90 AND
  longitude >= -180 AND
  longitude <= 180);
```

A senior engineer has confirmed the above logic is correct and the valid ranges for latitude and longitude are provided, but the code fails when executed.
Which statement explains the cause of this failure?
A. Because another team uses this table to support a frequently running application, two-phase locking is preventing the operation from committing.
B. The activity details table already exists; CHECK constraints can only be added during initial table creation.
C. The activity details table already contains records that violate the constraints; all existing data must pass CHECK constraints in order to add them to an existing table.
D. The activity details table already contains records; CHECK constraints can only be added prior to inserting values into a table.
E. The current table schema does not contain the field valid coordinates; schema evolution will need to be enabled before altering the table to add a constraint.
Answer: (SHOW ANSWER)
The failure is that the code to add CHECK constraints to the Delta Lake table fails when executed. The code uses ALTER TABLE ADD CONSTRAINT commands to add two CHECK constraints to a table named activity_details. The first constraint checks if the latitude value is between -90 and 90, and the second constraint checks if the longitude value is between -180 and 180. The cause of this failure is that the activity_details table already contains records that violate these constraints, meaning that they have invalid latitude or longitude values outside of these ranges. When adding CHECK constraints to an existing table, Delta Lake verifies that all existing data satisfies the constraints before adding them to the table. If any record violates the constraints, Delta Lake throws an exception and aborts the operation.
Verified Reference: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Add a CHECK constraint to an existing table" section.
https://docs.databricks.com/en/sql/language-manual/sql-ref-syntax-ddl-alter-table.html#add-constraint

NEW QUESTION: 6
A SQL Dashboard was built for the supply chain team to monitor the inventory and product orders, but all of the timestamps displayed on the dashboards are showing in UTC format, so they requested to change the time zone to the location of New York. How would you approach resolving this issue?
A. Move the workspace from Central US zone to East US Zone
B. Change the timestamp on the delta tables to America/New_York format
C. Change the spark configuration of SQL endpoint to format the timestamp to Ameri-ca/New_York
D. Under SQL Admin Console, set the SQL configuration parameter time zone to Ameri-ca/New_York
E. Add SET Timezone = America/New_York on every of the SQL queries in the dashboard.

Answer: D (LEAVE A REPLY)

Explanation

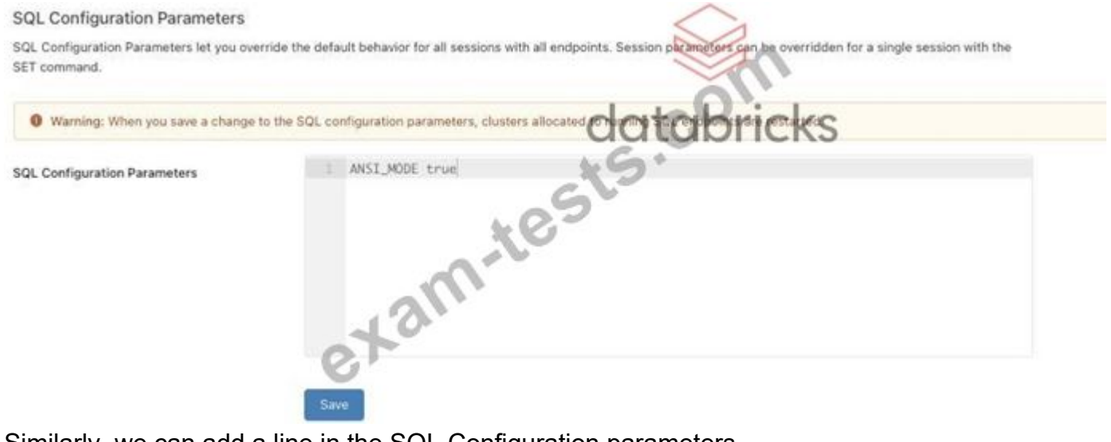
The answer is, Under SQL Admin Console, set the SQL configuration parameter time zone to America/New_York Here are steps you can take this to configure, so the entire dashboard is changed without changing individual queries Configure SQL parameters To configure all warehouses with SQL parameters:

1.Click Settings at the bottom of the sidebar and select SQL Admin Console.

2.Click the SQL Warehouse Settings tab.

3.In the SQL Configuration Parameters textbox, specify one key-value pair per line. Sepa-rate the name of the parameter from its value using a space. For example, to ena-ble ANSI_MODE:

Graphical user interface, text, application Description automatically generated



Similarly, we can add a line in the SQL Configuration parameters

timezone America/New_York

SQL configuration parameters | Databricks on AWS

NEW QUESTION: 7

A data engineer is creating a daily reporting job. There are two reporting notebooks-one for weekdays and one for weekends. An "if/else condition" task is configured as {{job.start_time.is_weekday}} == true to route the job to either the weekday or weekend notebook tasks. The same job would be used across multiple time zones.

Which action should a senior data engineer take upon reviewing the job to merge or reject the pull request?

A. Reject, as the {{job.start_time.is_weekday}} is for the UTC timezone.

B. Reject, as the {{job.start_time.is_weekday}} is not a valid value reference.

C. Merge, as the job configuration looks good.

D. Reject, as they should use {{job.trigger_time.is_weekday}} instead.

Answer: A (LEAVE A REPLY)

Comprehensive and Detailed Explanation From Exact Extract of Databricks Data Engineer Documents:

Databricks parameter templates like {{job.start_time.is_weekday}} evaluate in UTC time by default, not in local workspace or regional time zones. Therefore, when jobs are configured to run across different time zones, relying on is_weekday using UTC may cause scheduling and task routing mismatches (for example, triggering the weekday notebook in one region while it's still the weekend locally).

Databricks recommends adjusting conditional logic or pipeline parameters explicitly to handle time zone conversions if business requirements depend on local times. Because the engineer's configuration does not account for this behavior, a senior data engineer should reject the pull request and suggest time-zone-aware logic before merging.

NEW QUESTION: 8

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"))
```

exam-tests.com



databricks

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: (SHOW ANSWER)

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 Reference: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Write to Delta tables" section.

NEW QUESTION: 9
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 storage. leverage Databricks for all ELT.
C. Whenever a table is being created, make sure that the location keyword is used.
D. When tables are created, make sure that the external keyword is used in the create table statement.
E. When the workspace is being configured, make sure that external cloud object storage has been mounted.
Answer: C (LEAVE A REPLY)
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: 10
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 *
```

databricks
exam-tests.com

The viewnew_eventscontains a batch of records with the same schema as theeventsDelta table.
Theevent_idfield serves as a unique key for this table.
When this query is executed, what will happen with new records that have the sameevent_idas 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 (LEAVE A REPLY)
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%20o

NEW QUESTION: 11
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 allowed. "Can Attach To" privileges on the required cluster
C. Cluster creation allowed. "Can Attach To" privileges on the required cluster
D. "Can Restart" privileges on the required cluster
E. Cluster creation allowed. "Can Restart" privileges on the required cluster
Answer: (SHOW ANSWER)
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: 12
You have noticed the Data scientist team is using the notebook versioning feature with git integra-tion, you have recommended them to switch to using Databricks Repos, which of the below reasons could be the reason the why the team needs to switch to Databricks Repos.
A. Databricks Repos allows multiple users to make changes
B. Databricks Repos allows merge and conflict resolution
C. Databricks Repos has a built-in version control system
D. Databricks Repos automatically saves changes
E. Databricks Repos allow you to add comments and select the changes you want to commit.
Answer: E (LEAVE A REPLY)
Explanation
The answer is Databricks Repos allow you to add comments and select the changes you want to commit.

NEW QUESTION: 13
A data team ' s Structured Streaming job is configured to calculate running aggregates for item sales to update a downstream marketing dashboard. The marketing team has introduced a new field to track the number of times this promotion code is used for each item. A junior data engineer suggests updating the existing query as follows: Note that proposed changes are in bold.



exam-tests.com
databricks

Which step must also be completed to put the proposed query into production?

A. Increase the shuffle partitions to account for additional aggregates

B. Specify a new checkpointlocation

C. Run REFRESH TABLE delta, /item_agg '

D. Remove .option (mergeSchema ' , true ') from the streaming write

Answer: B (LEAVE A REPLY)

When introducing a new aggregation or a change in the logic of a Structured Streaming query, it is generally necessary to specify a new checkpoint location. This is because the checkpoint directory contains metadata about the offsets and the state of the aggregations of a streaming query. If the logic of the query changes, such as including a new aggregation field, the state information saved in the current checkpoint would not be compatible with the new logic, potentially leading to incorrect results or failures. Therefore, to accommodate the new field and ensure the streaming job has the correct starting point and state information for aggregations, a new checkpoint location should be specified.

:

Databricks documentation on Structured Streaming: https://docs.databricks.com/spark/latest/structured- streaming/index.html Databricks documentation on streaming checkpoints: https://docs.databricks.com/spark/latest/structured- streaming/production.html#checkpointing

NEW QUESTION: 14

A data architect has determined that a table of the following format is necessary:

Which of the following code blocks uses SQL DDL commands to create an empty Delta table in the above format regardless of whether a table already exists with this name?

A. 1. CREATE OR REPLACE TABLE table_name AS
2. SELECT id STRING, birthDate DATE, avgRating FLOAT USING DELTA

B. 1. CREATE OR REPLACE TABLE table_name (id STRING, birthDate DATE, avgRating FLOAT)

C. 1. CREATE TABLE table_name AS
2. SELECT id STRING, birthDate DATE, avgRating FLOAT

D. 1. CREATE OR REPLACE TABLE table_name
2. WITH COLUMNS (id STRING, birthDate DATE, avgRating FLOAT) USING DELTA

E. 1. CREATE TABLE IF NOT EXISTS table_name (id STRING, birthDate DATE, avgRating FLOAT)

Answer: B (LEAVE A REPLY)

NEW QUESTION: 15
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 (LEAVE A REPLY)
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.
Reference:
Databricks Documentation on Delta Live Tables: Delta Live Tables Guide

NEW QUESTION: 16
A CHECK constraint has been successfully added to the Delta table named activity_details using the following logic:

```
ALTER TABLE activity_details
ADD CONSTRAINT valid_coordinates
CHECK (
  latitude >= -90 AND
  latitude <= 90 AND
  longitude >= -180 AND
  longitude <= 180);
```

exam-tests.com
databricks

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 (LEAVE A REPLY)
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. Reference:
Constraints: https://docs.delta.io/latest/delta-constraints.html
ACID Transactions: https://docs.delta.io/latest/delta-intro.html#acid-transactions

Valid Databricks-Certified-Professional-Data-Engineer Dumps shared by BraindumpsPass.com for Helping Passing Databricks-Certified-Professional-Data-Engineer Exam! BraindumpsPass.com now offer the **newest Databricks-Certified-Professional-Data-Engineer exam dumps**, the BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer exam **questions have been updated** and **answers have been corrected** get the **newest** BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer dumps with Test Engine here:

https://www.braindumpspass.com/Databricks/Databricks-Certified-Professional-Data-Engineer-practice-exam-dumps.html (**217** Q&As Dumps, **40%OFF** **Special Discount:** **Exam-Tests**)

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: A (LEAVE A REPLY)

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: 18

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

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

```
CREATE DATABASE finance_eda_db
LOCATION '/mnt/finance_eda_bucket';
GRANT USAGE ON DATABASE finance_eda_db TO finance;
GRANT CREATE ON DATABASE finance_eda_db TO finance;
```

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

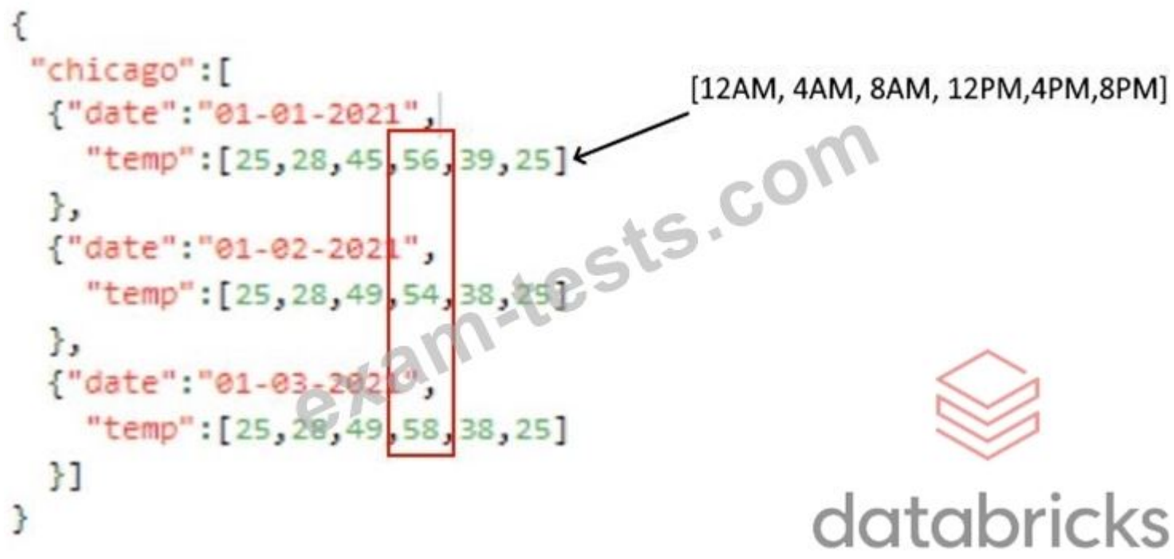
```
CREATE TABLE finance_eda_db.tx_sales AS
SELECT *
FROM sales
WHERE state = "TX";
```

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

A. A logical table will persist the query plan to the Hive Metastore in the Databricks control plane.

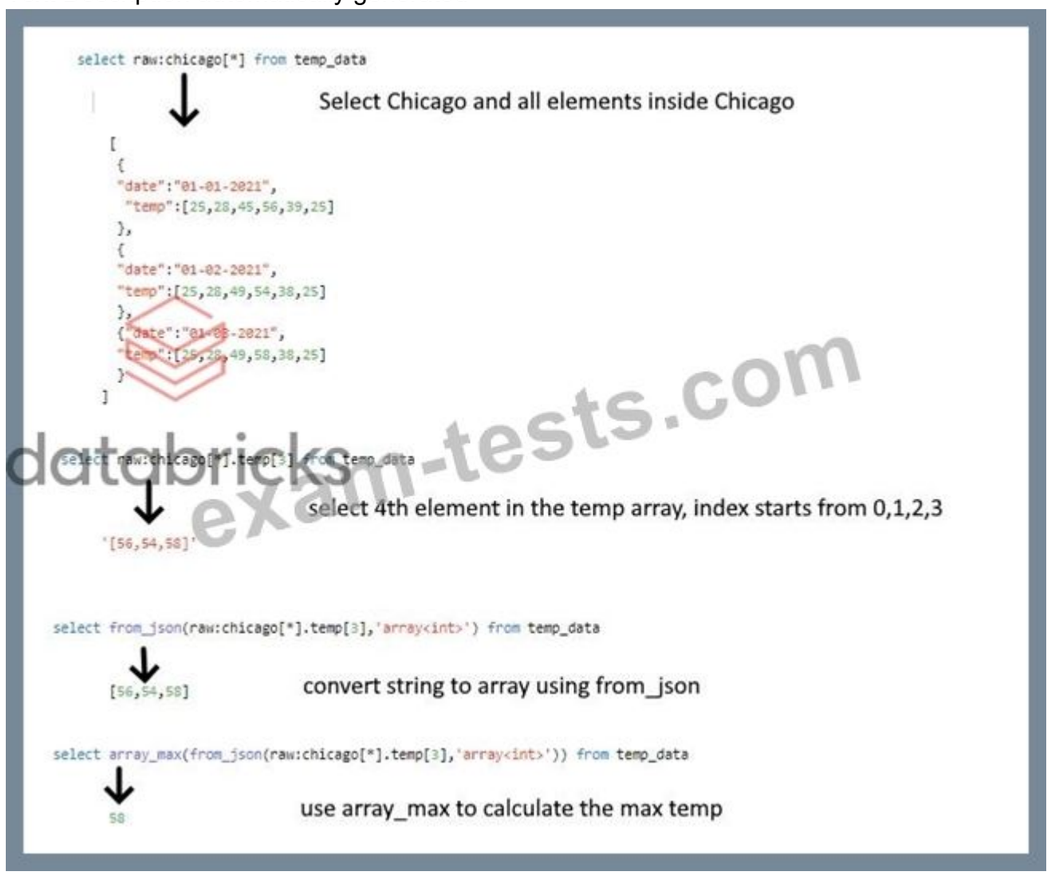
B. An external table will be created in the storage container mounted to /mnt/finance eda bucket.
C. A logical table will persist the physical plan to the Hive Metastore in the Databricks control plane.
D. An managed table will be created in the storage container mounted to /mnt/finance eda bucket.
E. A managed table will be created in the DBFS root storage container.
Answer: D (LEAVE A REPLY)
https://docs.databricks.com/en/lakehouse/data-objects.html

NEW QUESTION: 19
Below table temp_data has one column called raw contains JSON data that records temperature for every four hours in the day for the city of Chicago, you are asked to calculate the maximum temperature that was ever recorded for 12:00 PM hour across all the days. Parse the JSON data and use the necessary array function to calculate the max temp.
Table: temp_date
Column: raw
Datatype: string



Expected output: 58
A. 1.select max(raw.chicago.temp[3]) from temp_data
B. 1.select array_max(raw.chicago[*].temp[3]) from temp_data
C. 1.select array_max(from_json(raw['chicago'].temp[3],'array<int>')) from temp_data
D. 1.select array_max(from_json(raw:chicago[*].temp[3],'array<int>')) from temp_data
E. 1.select max(from_json(raw:chicago[3].temp[3],'array<int>')) from temp_data
Answer: D (LEAVE A REPLY)
Explanation
Note: This is a difficult question, more likely you may see easier questions similar to this but the more you are prepared for the exam easier it is to pass the exam.
Use this below link to look for more examples, this will definitely help you,
https://docs.databricks.com/optimizations/semi-structured.html
Here is the solution, step by step

Text Description automatically generated



databricks exam-tests.com

Use this below link to look for more examples, this will definitely help you,
https://docs.databricks.com/optimizations/semi-structured.html
If you want to try this solution use below DDL,
1.create or replace table temp_data
2. as select ' {
3. "chicago":[
4.{"date":"01-01-2021",
5."temp":[25,28,45,56,39,25]
6.},
7.{"date":"01-02-2021",
8."temp":[25,28,49,54,38,25]
9.},
10.{"date":"01-03-2021",
11."temp":[25,28,49,58,38,25]
12. }]
13. }
14. ' as raw

15.
16.select array_max(from_json(raw:chicago[*].temp[3],'array<int>')) from temp_data
17.

NEW QUESTION: 20
The data architect has decided that once data has been ingested from external sources into the Databricks Lakehouse, table access controls will be leveraged to manage permissions for all production tables and views.
The following logic was executed to grant privileges for interactive queries on a production database to the core engineering group.
GRANT USAGE ON DATABASE prod TO eng;
GRANT SELECT ON DATABASE prod TO eng;
Assuming these are the only privileges that have been granted to the eng group and that these users are not workspace administrators, which statement describes their privileges?
A. Group members have full permissions on the prod database and can also assign permissions to other users or groups.
B. Group members are able to list all tables in the prod database but are not able to see the results of any queries on those tables.
C. Group members are able to query and modify all tables and views in the prod database, but cannot create new tables or views.
D. Group members are able to query all tables and views in the prod database, but cannot create or edit anything in the database.
E. Group members are able to create, query, and modify all tables and views in the prod database, but cannot define custom functions.
Answer: D (LEAVE A REPLY)
The GRANT USAGE ON DATABASE prod TO eng command grants the eng group the permission to use the prod database, which means they can list and access the tables and views in the database. The GRANT SELECT ON DATABASE prod TO eng command grants the eng group the permission to select data from the tables and views in the prod database, which means they can query the data using SQL or DataFrame API. However, these commands do not grant the eng group any other permissions, such as creating, modifying, or deleting tables and views, or defining custom functions. Therefore, the eng group members are able to query all tables and views in the prod database, but cannot create or edit anything in the database. Reference:
Grant privileges on a database: https://docs.databricks.com/en/security/auth-authz/table-acls/grant-privileges-database.html Privileges you can grant on Hive metastore objects: https://docs.databricks.com/en/security/auth-authz/table-acls/privileges.html

NEW QUESTION: 21
A table is registered with the following code:

```
CREATE TABLE recent_orders AS (
  SELECT a.user_id, a.email, b.order_id, b.order_date
  FROM
    (SELECT user_id, email
    FROM users) a
    INNER JOIN
    (SELECT user_id, order_id, order_date
    FROM orders
    WHERE order_date >= (current_date() - 7)) b
    ON a.user_id = b.user_id
)
```

databricks exam-tests.com
Both users and orders are Delta Lake tables. Which statement describes the results of querying recent_orders?
A. 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.
B. 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.
C. 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.
D. The versions of each source table will be stored in the table transaction log; query results will be saved to DBFS with each query.

E. 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.
Answer: A (LEAVE A REPLY)

NEW QUESTION: 22
Which of the below commands can be used to drop a DELTA table?
A. DROP table_name
B. DROP DELTA table_name
C. DROP TABLE table_name
D. DROP TABLE table_name FORMAT DELTA
Answer: (SHOW ANSWER)

NEW QUESTION: 23
A data engineer is tasked with ensuring that a Delta table in Databricks continuously retains deleted files for
15 days instead of the default 7 days, in order to comply with the organization's data retention policy. Which code snippet correctly sets this retention period for deleted files?
A. spark.sql(" " "
ALTER TABLE my_table
SET TBLPROPERTIES (' delta.deletedFileRetentionDuration ' = ' interval 15 days ')
" " ")
B. from delta.tables import *
deltaTable = DeltaTable.forPath(spark, " /mnt/data/my_table ")
deltaTable.deletedFileRetentionDuration = " interval 15 days "
C. spark.sql(" VACUUM my_table RETAIN 15 HOURS ")
D. spark.conf.set(" spark.databricks.delta.deletedFileRetentionDuration " , " 15 days ")
Answer: (SHOW ANSWER)
Databricks documents delta.deletedFileRetentionDuration as a Delta table property and shows that Delta table properties are modified with SET TBLPROPERTIES . The documented value format is an interval expression such as ' interval 7 days ' , so ' interval 15 days ' is the correct way to set the retention window on the table itself. (Databricks Documentation) Option C controls a specific VACUUM run and is shown in hours here, not as a persistent table-level retention setting. Option D sets a Spark configuration rather than the table property required by the question.
Option B is not the documented API for setting this Delta retention property. (Databricks Documentation)
======

NEW QUESTION: 24
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 thegeo_lookuptable.
Before executing the code, runningSHOWTABLESon the current database indicates the database contains only two tables:geo_lookupandsales.



Which statement correctly describes the outcome of executing these command cells in order in an interactive notebook?

A. Both commands will succeed. Executing show tables will show that countries at and sales at have been registered as views.

B. Cmd 1 will succeed. Cmd 2 will search all accessible databases for a table or view named countries af:
if this entity exists, Cmd 2 will succeed.

C. Cmd 1 will succeed and Cmd 2 will fail, countries at will be a Python variable representing a PySpark DataFrame.

D. Both commands will fail. No new variables, tables, or views will be created.

E. Cmd 1 will succeed and Cmd 2 will fail, countries at will be a Python variable containing a list of strings.

Answer: E (LEAVE A REPLY)

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: 25

Which method can be used to determine the total wall-clock time it took to execute a query?

A. In the Spark UI, take the job duration of the longest-running job associated with that query.

B. In the Spark UI, take the sum of all task durations that ran across all stages for all jobs associated with that query.

C. Open the Query Profiler associated with that query and use the Total wall-clock duration metric.

D. Open the Query Profiler associated with that query and use the Aggregated task time metric.

Answer: C (LEAVE A REPLY)

The Query Profiler in Databricks SQL and notebooks provides a detailed breakdown of query performance metrics. The "Total wall-clock duration" metric directly represents the total elapsed time from query start to completion, including all execution, planning, and waiting stages. In contrast, "Aggregated task time" reflects the cumulative duration across all parallel tasks, which does not equal the total elapsed wall time since tasks often run concurrently. Using job duration from Spark UI can underestimate or overestimate runtime when queries span multiple jobs. Therefore, the Query Profiler's total wall-clock duration is the officially documented method to determine actual query execution time.

NEW QUESTION: 26

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 (LEAVE A REPLY)
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.
Reference:
Databricks Documentation on Testing: Testing and Validation of Data and Notebooks

NEW QUESTION: 27
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 (LEAVE A REPLY)
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.
Reference:
Databricks documentation on using SQL views in Delta Lake: https://docs.databricks.com/delta/quick-start.html#sql-views

NEW QUESTION: 28
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 Git. Most of the latency is related to cluster startup time.
B. 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.
C. %sh does not distribute file moving operations; the final line of code should be updated to use %fs instead.
D. Python will always execute slower than Scala on Databricks. The run.py script should be refactored to Scala.
E. %sh executes shell code on the driver node. The code does not take advantage of the worker nodes or Databricks optimized Spark.
Answer: E (LEAVE A REPLY)
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: 29
A data engineer is attempting to execute the following PySpark code:
df = spark.read.table("sales")
result = df.groupBy("region").agg(sum("revenue"))
However, upon inspecting the execution plan and profiling the Spark job, they observe excessive data shuffling during the aggregation phase.
Which technique should be applied to reduce shuffling during the groupBy aggregation operation?
A. Caching the DataFrame df.
B. Repartition by region before aggregation.
C. Use coalesce() after the aggregation.
D. Use broadcast join.
Answer: B (LEAVE A REPLY)
Comprehensive and Detailed Explanation From Exact Extract of Databricks Data Engineer Documents:
Databricks documents that shuffle occurs when Spark redistributes data across partitions for grouping or joining. To optimize aggregation performance, repartitioning by the grouping key (region) ensures rows with the same key are co-located in the same partition, thus minimizing shuffle movement. Caching improves reuse of DataFrames but does not reduce shuffle volume. coalesce() reduces the number of partitions after computation and cannot prevent shuffle. Broadcast joins are unrelated to single-table aggregations. The recommended practice for reducing shuffle in aggregation is explicit repartitioning by the grouping column.

NEW QUESTION: 30
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: (SHOW ANSWER)
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: 31
A data ingestion task requires a one-TB JSON dataset to be written out to Parquet with a target part-file size of
512 MB. Because Parquet is being used instead of Delta Lake, built-in file-sizing features such as Auto-Optimize & Auto-Compaction cannot be used.
Which strategy will yield the best performance without shuffling data?
A. Set spark.sql.files.maxPartitionBytes to 512 MB, ingest the data, execute the narrow transformations, and then write to parquet.
B. Set spark.sql.shuffle.partitions to 2,048 partitions (1TB*1024*1024/512), ingest the data, execute the narrow transformations, optimize the data by sorting it (which automatically repartitions the data), and then write to parquet.
C. Set spark.sql.shuffle.partitions to 512, ingest the data, execute the narrow transformations, and then write to parquet.
D. Set spark.sql.adaptive.advisoryPartitionSizeInBytes to 512 MB bytes, ingest the data, execute the narrow transformations, coalesce to 2,048 partitions (1TB*1024*1024/512), and then write to parquet.
E. Ingest the data, execute the narrow transformations, repartition to 2,048 partitions (1TB*
1024*1024/512), and then write to parquet.
Answer: B (LEAVE A REPLY)

Valid Databricks-Certified-Professional-Data-Engineer Dumps shared by BraindumpsPass.com for Helping Passing Databricks-Certified-Professional-Data-Engineer Exam! BraindumpsPass.com now offer the **newest Databricks-Certified-Professional-Data-Engineer exam dumps**, the BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer exam **questions have been updated** and **answers have been corrected** get the **newest** BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer dumps with Test Engine here:
https://www.braindumpspass.com/Databricks/Databricks-Certified-Professional-Data-Engineer-practice-exam-dumps.html (**217** Q&As Dumps, **40%OFF** **Special Discount:** **Exam-Tests**)

NEW QUESTION: 32
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: (SHOW ANSWER)
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: 33

The business reporting tem requires that data for their dashboards be updated every hour. The total processing time for the pipeline that extracts transforms and load the data for their pipeline runs in 10 minutes.
Assuming normal operating conditions, which configuration will meet their service-level agreement requirements with the lowest cost?
A. Schedule a jo to execute the pipeline once and hour on a dedicated interactive cluster.
B. Schedule a Structured Streaming job with a trigger interval of 60 minutes.
C. Schedule a job to execute the pipeline once hour on a new job cluster.
D. Configure a job that executes every time new data lands in a given directory.
Answer: (SHOW ANSWER)
Scheduling a job to execute the data processing pipeline once an hour on a new job cluster is the most cost- effective solution given the scenario. Job clusters are ephemeral in nature; they are spun up just before the job execution and terminated upon completion, which means you only incur costs for the time the cluster is active. Since the total processing time is only 10 minutes, a new job cluster created for each hourly execution minimizes the running time and thus the cost, while also fulfilling the requirement for hourly data updates for the business reporting team ' s dashboards.
:
Databricks documentation on jobs and job clusters: https://docs.databricks.com/jobs.html

NEW QUESTION: 34
Which of the following python statement can be used to replace the schema name and table name in the query statement?
A. 1.table_name = "sales"
2.schema_name = "bronze"
3.query = f"select * from schema_name.table_name"
B. 1.table_name = "sales"
2.schema_name = "bronze"
3.query = "select * from {schema_name}.{table_name}"
C. 1.table_name = "sales"
2.schema_name = "bronze"
3.query = f"select * from { schema_name}.{table_name}"
D. 1.table_name = "sales"
2.schema_name = "bronze"
3.query = f"select * from + schema_name +"."+table_name"
Answer: C (LEAVE A REPLY)
Explanation
Answer is
table_name = "sales"
query = f"select * from {schema_name}.{table_name}"
f strings can be used to format a string. f" This is string {python variable}"
https://realpython.com/python-f-strings/

NEW QUESTION: 35
A data engineer has a Delta table orders with deletion vectors enabled. The engineer executes the following command:
DELETE FROM orders WHERE status = ' cancelled ' ;
What should be the behavior of deletion vectors when the command is executed?
A. Rows are marked as deleted both in metadata and in files.

B. Delta automatically removes all cancelled orders permanently.
C. Files are physically rewritten without the deleted rows.
D. Rows are marked as deleted in metadata, not in files.
Answer: D (LEAVE A REPLY)
Deletion vectors (DVs) in Delta Lake optimize delete operations by marking deleted rows logically in metadata rather than rewriting Parquet files. When a DELETE statement is executed, affected rows are tracked by DVs in the transaction log. The data remains in the underlying files but is filtered out during query reads. This improves performance for frequent deletes and updates since file rewrites are deferred. Physical data removal only occurs when a VACUUM command is later executed. The Databricks documentation confirms: "With deletion vectors, deleted rows are marked in metadata and skipped at read time, avoiding file rewrites." Thus, rows are marked as deleted in metadata-not in files.

NEW QUESTION: 36
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: (SHOW ANSWER)
Explanation
This is the correct way to get a parameter passed to a notebook by the Databricks Jobs API. The dbutils.notebooks.getParam method returns the value of a parameter passed to a notebook as a string. If no parameter with that name is passed, it returns None by default. You can also specify a default value as a second argument. Verified References: Databricks Certified Data Engineer Professional, under "Databricks Tooling" section; Databricks Documentation, under "Pass parameters to a notebook" section.

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
SELECT email, age, ltv
FROM user_ltv
WHERE
  CASE
    WHEN is_member("auditing") THEN TRUE
    ELSE age >= 18
  END
```

exam-tests.com databricks

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: (SHOW ANSWER)
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: 38
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: (SHOW ANSWER)
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: 39

The data engineering team maintains the following code:

```
import pyspark.sql.functions as F

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

examtests.com



databricks

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

A. The silver_customer_sales table will be overwritten by aggregated values calculated from all records in the gold_customer_lifetime_sales_summary table as a batch job.

B. A batch job will update the gold_customer_lifetime_sales_summary table, replacing only those rows that have different values than the current version of the table, using customer_id as the primary key.

C. The gold_customer_lifetime_sales_summary table will be overwritten by aggregated values calculated from all records in the silver_customer_sales table as a batch job.

D. An incremental job will leverage running information in the state store to update aggregate values in the gold_customer_lifetime_sales_summary table.

E. An incremental job will detect if new rows have been written to the silver_customer_sales table; if new rows are detected, all aggregates will be recalculated and used to overwrite the gold_customer_lifetime_sales_summary table.

Answer: (SHOW ANSWER)

This code is using the pyspark.sql.functions library to group the silver_customer_sales table by customer_id and then aggregate the data using the minimum sale date, maximum sale total, and sum of distinct order ids.

The resulting aggregated data is then written to the gold_customer_lifetime_sales_summary table, overwriting any existing data in that table. This is a batch job that does not use any incremental or streaming logic, and does not perform any merge or update operations. Therefore, the code will overwrite the gold table with the aggregated values from the silver table every time it is executed. References:

* https://docs.databricks.com/spark/latest/dataframes-datasets/introduction-to-dataframes-python.html
* https://docs.databricks.com/spark/latest/dataframes-datasets/transforming-data-with-dataframes.html
* https://docs.databricks.com/spark/latest/dataframes-datasets/aggregating-data-with-dataframes.html

NEW QUESTION: 40

When working with AUTO LOADER you noticed that most of the columns that were inferred as part of loading are string data types including columns that were supposed to be integers, how can we fix this?

A. Provide the schema of the source table in the cloudfiles.schemalocation

B. Provide the schema of the target table in the cloudfiles.schemalocation

C. Provide schema hints

D. Update the checkpoint location

E. Correct the incoming data by explicitly casting the data types
Answer: C (LEAVE A REPLY)
Explanation
The answer is, Provide schema hints.
1.spark.readStream \
2.format("cloudFiles") \
3.option("cloudFiles.format", "csv") \
4.option("header", "true") \
5.option("cloudFiles.schemaLocation", schema_location) \
6.option("cloudFiles.schemaHints", "id int, description string")
7.load(raw_data_location)
8.writeStream \
9.option("checkpointLocation", checkpoint_location) \
10.start(target_delta_table_location)option("cloudFiles.schemaHints", "id int, description string")
# Here we are providing a hint that id column is int and the description is a string When cloudfiles.schemalocation is used to store the output of the schema inference during the load process, with schema hints you can enforce data types for known columns ahead of time.

NEW QUESTION: 41
Which of the following statements are true about a lakehouse?
A. Lakehouse only supports Machine learning workloads and Data warehouses support BI workloads
B. Lakehouse only supports end-to-end streaming workloads and Data warehouses support Batch workloads
C. Lakehouse does not support ACID
D. Lakehouse do not support SQL
E. Lakehouse supports Transactions
Answer: E (LEAVE A REPLY)
Explanation
What Is a Lakehouse? - The Databricks Blog
Text Description automatically generated

A lakehouse has the following key features:

- **Transaction support:** In an enterprise lakehouse many data pipelines will often be reading and writing data concurrently. Support for ACID transactions ensures consistency as multiple parties concurrently read or write data, typically using SQL.
- **Schema enforcement and governance:** The Lakehouse should have a way to support schema enforcement and evolution, supporting DW schema architectures such as star/snowflake-schemas. The system should be able to reason about data integrity, and it should have robust governance and auditing mechanisms.
- **BI support:** Lakehouses enable using BI tools directly on the source data. This reduces staleness and improves recency, reduces latency, and lowers the cost of having to operationalize two copies of the data in both a data lake and a warehouse.
- **Storage is decoupled from compute:** In practice this means storage and compute use separate clusters, thus these systems are able to scale to many more concurrent users and larger data sizes. Some modern data warehouses also have this property.
- **Openness:** The storage formats they use are open and standardized, such as Parquet, and they provide an API so a variety of tools and engines, including machine learning and Python/R libraries, can efficiently access the data **directly**.
- **Support for diverse data types ranging from unstructured to structured data**: The lakehouse can be used to store, refine, analyze, and access data types needed for many new data applications, including images, video, audio, semi-structured data, and text.
- **Support for diverse workloads:** including data science, machine learning, and SQL and analytics. Multiple tools might be needed to support all these workloads but they all rely on the same data repository.
- **End-to-end streaming:** Real-time reports are the norm in many enterprises. Support for streaming eliminates the need for separate systems dedicated to serving real-time data applications.

exam-tests.com
databricks

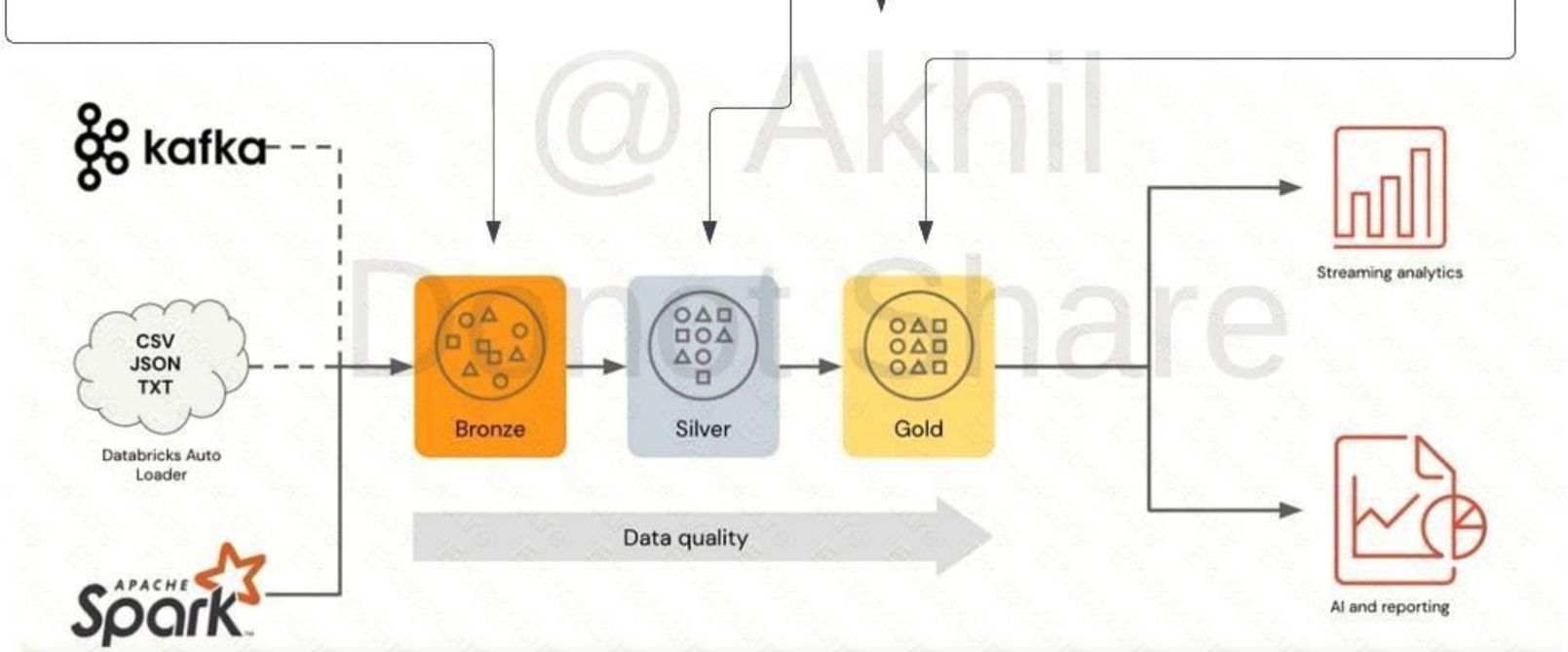
NEW QUESTION: 42
The DevOps team has configured a production workload as a collection of notebooks scheduled to run daily using the Jobs UI. A new data engineering hire is onboarding to the team and has requested access to one of these notebooks to review the production logic.
What are the maximum notebook permissions that can be granted to the user without allowing accidental changes to production code or data?
A. Can Manage
B. Can Edit
C. No permissions
D. Can Read
E. Can Run
Answer: (SHOW ANSWER)
Explanation
This is the correct answer because it is the maximum notebook permissions that can be granted to the user without allowing accidental changes to production code or data. Notebook permissions are used to control access to notebooks in Databricks workspaces. There are four types of notebook permissions: Can Manage, Can Edit, Can Run, and Can Read. Can Manage allows full control over the notebook, including editing, running, deleting, exporting, and changing permissions. Can Edit allows modifying and running the notebook, but not changing permissions or deleting it. Can Run allows executing commands in an existing cluster attached to the notebook, but not modifying or exporting it. Can Read allows viewing the notebook content, but not running or modifying it. In this case, granting Can Read permission to the user will allow them to review the production logic in the notebook without allowing them to makeany changes to it or run any commands that may affect production data. Verified References: [Databricks Certified Data Engineer Professional], under "Databricks Workspace" section; Databricks Documentation, under "Notebook permissions" section.

NEW QUESTION: 43
Which of the following data workloads will utilize a gold table as its source?
A. A job that enriches data by parsing its timestamps into a human-readable format
B. A job that queries aggregated data that already feeds into a dashboard
C. A job that ingests raw data from a streaming source into the Lakehouse
D. A job that aggregates cleaned data to create standard summary statistics
E. A job that cleans data by removing malformatted records
Answer: B (LEAVE A REPLY)
Explanation
The answer is, A job that queries aggregated data that already feeds into a dashboard The gold layer is used to store aggregated data, which are typically used for dashboards and reporting.
Review the below link for more info,
Medallion Architecture - Databricks
Gold Layer:
1. Powers Ml applications, reporting, dashboards, ad hoc analytics
2. Refined views of data, typically with aggregations
3. Reduces strain on production systems
4. Optimizes query performance for business-critical data
Exam focus: Please review the below image and understand the role of each layer(bronze, silver, gold) in medallion architecture, you will see varying questions targeting each layer and its purpose.
Sorry I had to add the watermark some people in Udemy are copying my content.
Purpose of each layer in medallion architecture

Bronze Layer:
1.Raw copy of ingested data
2.Replaces traditional data lake
3.Provides efficient storage and querying of full, unprocessed history of data
4.No schema is applied at this layer

Silver Layer:
1.Reduces data storage complexity, latency, and redundency
2.Optimizes ETL throughput and analytic query performance
3.Preserves grain of original data (without aggregation)
4.Eliminates duplicate records
5.Production schema enforced
6.Data quality checks, quarantine corrupt data

Gold Layer:
1.Powers Ml applications, reporting, dashboards, ad hoc analytics
2.Refined views of data, typically with aggregations
3.Reduces strain on production systems
4.Optimizes query performance for business-critical data



@ Akhil
Don't Share

NEW QUESTION: 44

The Databricks CLI is used to trigger a run of an existing job by passing the job_id parameter. The response indicating the job run request was submitted successfully includes a field run_id. Which statement describes what the number alongside this field represents?

A. The job_id and number of times the job has been run are concatenated and returned.

B. The globally unique ID of the newly triggered run.

C. The job_id is returned in this field.

D. The number of times the job definition has been run in this workspace.

Answer: (SHOW ANSWER)

Comprehensive and Detailed Explanation From Exact Extract:

Exact extract: "run_id: The canonical identifier of a run."

Exact extract: "Each job run has a unique run_id."

When a run is created via the Jobs API/CLI, Databricks returns a unique run_id for that specific run instance, distinct from the job's job_id.

Reference:

NEW QUESTION: 45

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 namedusers.

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

Assuming thatuser_idis a unique identifying key and that 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 mergeinto 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 (LEAVE A REPLY)

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: 46

A facilities-monitoring team is building a near-real-time PowerBI dashboard off the Delta table device_readings:

Columns:

* device_id (STRING, unique sensor ID)

* event_ts (TIMESTAMP, ingestion timestamp UTC)

* temperature_c (DOUBLE, temperature in °C)
Requirement:
* For each sensor, generate one row per non-overlapping 5-minute interval , offset by 2 minutes (e.g.,
00:02-00:07, 00:07-00:12, ...).
* Each row must include interval start, interval end, and average temperature in that slice.
* Downstream BI tools (e.g., Power BI) must use the interval timestamps to plot time-series bars.
Options:
A. WITH buckets AS (
SELECT device_id,
window(event_ts, ' 5 minutes ' , ' 2 minutes ' , ' 5 minutes ') AS win, temperature_c FROM device_readings) SELECT device_id, win.start AS bucket_start, win.end AS bucket_end, AVG(temperature_c) AS avg_temp_5m FROM buckets GROUP BY device_id, win ORDER BY device_id, bucket_start;
B. SELECT device_id,
event_ts,
AVG(temperature_c) OVER (
PARTITION BY device_id
ORDER BY event_ts
RANGE BETWEEN INTERVAL 5 MINUTES PRECEDING AND CURRENT ROW
) AS avg_temp_5m
FROM device_readings
WINDOW w AS (window(event_ts, ' 5 minutes ' , ' 2 minutes '));
C. SELECT device_id,
date_trunc(' minute ' , event_ts - INTERVAL 2 MINUTES) + INTERVAL 2 MINUTES AS bucket_start, date_trunc(' minute ' , event_ts - INTERVAL 2 MINUTES) + INTERVAL 7 MINUTES AS bucket_end,
AVG(temperature_c) AS avg_temp_5m FROM device_readings GROUP BY device_id, date_trunc(' minute ' , event_ts - INTERVAL 2 MINUTES) ORDER BY device_id, bucket_start;
D. SELECT device_id,
window.start AS bucket_start,
window.end AS bucket_end,
AVG(temperature_c) AS avg_temp_5m
FROM device_readings
GROUP BY device_id, window(event_ts, ' 5 minutes ' , ' 5 minutes ' , ' 2 minutes ') ORDER BY device_id, bucket_start;
Answer: A (LEAVE A REPLY)
The correct way to satisfy non-overlapping windows with an offset in Databricks SQL is to use the window function with three parameters: window duration, slide duration, and start offset .
* In option A , the function call:
* window(event_ts, ' 5 minutes ' , ' 2 minutes ' , ' 5 minutes ')
creates 5-minute windows that slide every 5 minutes , with a 2-minute offset , which exactly matches the requirement (intervals like 00:02-00:07, 00:07-00:12, ...).
* Option B is incorrect because it uses a windowed aggregation with RANGE , which produces overlapping sliding averages , not discrete non-overlapping buckets.
* Option C manually constructs bucket boundaries with date_trunc and offsets, but this is brittle and less efficient than the built-in window function.
* Option D incorrectly passes four parameters to window but with the wrong ordering (5 minutes, 5 minutes, 2 minutes). This creates a sliding window every 5 minutes with overlap , rather than true non-overlapping shifted windows.
Reference (Databricks SQL Windowing Functions):
Databricks documentation specifies that:
window(time_col, windowDuration, slideDuration, startTime)

produces tumbling or sliding windows . When slideDuration = windowDuration, it produces non- overlapping tumbling windows . The startTime argument allows for offset windows , which is why ' 2 minutes ' ensures alignment at 00:02, 00:07, etc.
Thus, A is the only correct solution as it directly implements non-overlapping, offset-based tumbling windows .

Valid Databricks-Certified-Professional-Data-Engineer Dumps shared by BraindumpsPass.com for Helping Passing Databricks-Certified-Professional-Data-Engineer Exam! BraindumpsPass.com now offer the **newest Databricks-Certified-Professional-Data-Engineer exam dumps**, the BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer exam **questions have been updated** and **answers have been corrected** get the **newest** BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer dumps with Test Engine here:
https://www.braindumpspass.com/Databricks/Databricks-Certified-Professional-Data-Engineer-practice-exam-dumps.html (**217** Q&As Dumps, **40%OFF** **Special Discount:** **Exam-Tests**)

NEW QUESTION: 47
A data engineer is testing a collection of mathematical functions, one of which calculates the area under a curve as described by another function.
Which kind of the test does the above line exemplify?
A. Integration
B. Unit
C. Manual
D. functional
Answer: (SHOW ANSWER)
A unit test is designed to verify the correctness of a small, isolated piece of code, typically a single function. Testing a mathematical function that calculates the area under a curve is an example of a unit test because it is testing a specific, individual function to ensure it operates as expected.
Reference:
Software Testing Fundamentals: Unit Testing

NEW QUESTION: 48
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 (LEAVE A REPLY)
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: 49
When building a DLT s pipeline you have two options to create a live tables, what is the main dif-ference between CREATE STREAMING LIVE TABLE vs CREATE LIVE TABLE?
A. CREATE STREAMING LIVE table is used in MULTI HOP Architecture
B. CREATE LIVE TABLE is used when working with Streaming data sources and Incremental data
C. CREATE STREAMING LIVE TABLE is used when working with Streaming data sources and Incremental data
D. There is no difference both are the same, CREATE STRAMING LIVE will be deprecated soon
E. CREATE LIVE TABLE is used in DELTA LIVE TABLES, CREATE STREAMING LIVE can only used in Structured Streaming applications
Answer: (SHOW ANSWER)
Explanation
The answer is, CREATE STREAMING LIVE TABLE is used when working with Streaming data sources and Incremental data

NEW QUESTION: 50
Once a cluster is deleted, below additional actions need to performed by the administrator
A. Remove virtual machines but storage and networking are automatically dropped
B. Drop storage disks but Virtual machines and networking are automatically dropped
C. Remove networking but Virtual machines and storage disks are automatically dropped
D. Remove logs
E. No action needs to be performed. All resources are automatically removed.
Answer: E (LEAVE A REPLY)
Explanation
What is Delta?
Delta lake is
* Open source
* Builds up on standard data format
* Optimized for cloud object storage
* Built for scalable metadata handling
Delta lake is not
* Proprietary technology
* Storage format
* Storage medium
* Database service or data warehouse

NEW QUESTION: 51
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 (LEAVE A REPLY)

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 Reference: [Databricks Certified Data Engineer Professional], under "Structured Streaming" section; Databricks Documentation, under "Checkpointing" section.

NEW QUESTION: 52

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 ofVACUUMcommands 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 (LEAVE A REPLY)

Explanation

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

NEW QUESTION: 53

A data engineer wants to create a cluster using the Databricks CLI for a big ETL pipeline. The cluster should havefive workers,one driverof type i3.xlarge, and should use the '14.3.x-scala2.12' runtime.

Which command should the data engineer use?

A. databricks clusters create 14.3.x-scala2.12 --num-workers 5 --node-type-id i3.xlarge --cluster-name DataEngineer_cluster

B. databricks clusters add 14.3.x-scala2.12 --num-workers 5 --node-type-id i3.xlarge --cluster-name Data Engineer_cluster

C. databricks compute add 14.3.x-scala2.12 --num-workers 5 --node-type-id i3.xlarge --cluster-name Data Engineer_cluster

D. databricks compute create 14.3.x-scala2.12 --num-workers 5 --node-type-id i3.xlarge --cluster-name Data Engineer_cluster

Answer: (SHOW ANSWER)
Comprehensive and Detailed In-Depth Explanation:
TheDatabricks CLIallows users to manage clusters using command-line commands. The correct command for creating a cluster follows a specific format.
Key Components in the Command:
* Command Type:databricks compute create is the correct syntax for creating a new compute resource (cluster).
* Runtime Version:'14.3.x-scala2.12' specifies the Databricks runtime to use.
* Workers:--num-workers 5 sets the number of worker nodes to 5.
* Node Type:--node-type-id i3.xlarge defines the hardware configuration.
* Cluster Name:--cluster-name DataEngineer_cluster assigns a recognizable name to the cluster.
Evaluation of Options:
* Option A (databricks clusters create ...)
* Incorrect:databricks clusters createis not a valid commandin the Databricks CLI v0.205.
* The correct CLI command for cluster creation is databricks compute create.
* Option B (databricks clusters add ...)
* Incorrect:databricks clusters addis not a valid CLI command.
* Option C (databricks compute add ...)
* Incorrect:databricks compute addis not a valid CLI command.
* Option D (databricks compute create ...)
* Correct:databricks compute create is the correct command for creating a cluster.
Conclusion:
The correct command to create a cluster with five workers, an i3.xlarge node type, and Databricks runtime
14.3.x-scala2.12 is:
databricks compute create 14.3.x-scala2.12 --num-workers 5 --node-type-id i3.xlarge --cluster-name Data Engineer_cluster Thus, the correct answer isD.
References:
* Databricks CLI Documentation

NEW QUESTION: 54
A table in the Lakehouse namedcustomer_churn_paramsis 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: (SHOW ANSWER)

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: 55
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. Df. Select ("customer_id".
Model ("columns) alias ("predictions")
D. Df.apply(model, columns). Select ("customer_id, prediction"
Answer: A (LEAVE A REPLY)
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: 56
A streaming video analytics team ingests billions of events daily into a Unity Catalog-managed Delta table video_events . Analysts run ad-hoc point-lookup queries on columns like user_id, campaign_id, and region.
The team manually runs OPTIMIZE video_events ZORDER BY (user_id, campaign_id, region), but still sees poor performance on recent data and dislikes the operational overhead. The team wants a hands-off way to keep hot columns co-located as query patterns evolve.
A. Schedule OPTIMIZE/ZORDER to run after each job to improve recent file performance.
B. Enable Delta caching.
C. Utilize Liquid Clustering (CLUSTER BY AUTO) and Predictive Optimization.
D. Enable auto-compaction (optimizeWrite and autoCompact).
Answer: C (LEAVE A REPLY)
According to Databricks Delta Lake optimization documentation, Liquid Clustering is a next-generation file organization capability that automatically manages file co-location without requiring explicit partitioning or manual Z-ORDERing. When combined with Predictive Optimization , Databricks automatically maintains clustering across frequently filtered or queried columns, adapting dynamically as query workloads evolve.
This approach eliminates the need for manual maintenance (such as periodic OPTIMIZE or Z-ORDER commands) while improving query performance on large tables-particularly for high-ingest streaming workloads.
Delta caching (B) only improves performance for cached queries and does not address file layout issues, and (D) handles file size optimization but not clustering. Thus, C is the most efficient, modern, and low- maintenance solution recommended by Databricks.

NEW QUESTION: 57
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 storage. leverage Databricks for all ELT.
C. Whenever a table is being created, make sure that the location keyword is used.
D. When tables are created, make sure that the external keyword is used in the create table statement.
E. When the workspace is being configured, make sure that external cloud object storage has been mounted.
Answer: (SHOW ANSWER)
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 Reference: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Create an external table" section.

NEW QUESTION: 58
The data engineering team maintains the following code:

```
import pyspark.sql.functions as F

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

databricks
exam-tests.com

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

A. The silver_customer_sales table will be overwritten by aggregated values calculated from all records in the gold_customer_lifetime_sales_summary table as a batch job.

B. A batch job will update the gold_customer_lifetime_sales_summary table, replacing only those rows that have different values than the current version of the table, using customer_id as the primary key.

C. The gold_customer_lifetime_sales_summary table will be overwritten by aggregated values calculated from all records in the silver_customer_sales table as a batch job.

D. An incremental job will leverage running information in the state store to update aggregate values in the gold_customer_lifetime_sales_summary table.

E. An incremental job will detect if new rows have been written to the silver_customer_sales table; if new rows are detected, all aggregates will be recalculated and used to overwrite the gold_customer_lifetime_sales_summary table.

Answer: C (LEAVE A REPLY)

This code is using the pyspark.sql.functions library to group the silver_customer_sales table by customer_id and then aggregate the data using the minimum sale date, maximum sale total, and sum of distinct order ids.

The resulting aggregated data is then written to the gold_customer_lifetime_sales_summary table, overwriting any existing data in that table. This is a batch job that does not use any incremental or streaming logic, and does not perform any merge or update operations. Therefore, the code will overwrite the gold table with the aggregated values from the silver table every time it is executed. References:

* https://docs.databricks.com/spark/latest/dataframes-datasets/introduction-to-dataframes-python.html
* https://docs.databricks.com/spark/latest/dataframes-datasets/transforming-data-with-dataframes.html
* https://docs.databricks.com/spark/latest/dataframes-datasets/aggregating-data-with-dataframes.html

NEW QUESTION: 59

Create a sales database using the DBFS location 'dbfs:/mnt/delta/databases/sales.db/'

A. CREATE DATABASE sales FORMAT DELTA LOCATION 'dbfs:/mnt/delta/databases/sales.db/"

B. CREATE DATABASE sales USING LOCATION 'dbfs:/mnt/delta/databases/sales.db/'

C. CREATE DATABASE sales LOCATION 'dbfs:/mnt/delta/databases/sales.db/'

D. The sales database can only be created in Delta lake

E. CREATE DELTA DATABASE sales LOCATION 'dbfs:/mnt/delta/databases/sales.db/'

Answer: (SHOW ANSWER)

Explanation

The answer is

CREATE DATABASE sales LOCATION 'dbfs:/mnt/delta/databases/sales.db/'

Note: with the introduction of the Unity catalog and three-layer namespace usage of SCHEMA and DATABASE is interchangeable

NEW QUESTION: 60
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. Df. Select ("customer_id".
Model ("columns) alias ("predictions")
D. Df.apply(model, columns). Select ("customer_id, prediction"
Answer: (SHOW ANSWER)
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.
Reference:
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: 61
Which of the following tool provides Data Access control, Access Audit, Data Lineage, and Data discovery?
A. DELTA LIVE Pipelines
B. Unity Catalog
C. Data Governance
D. DELTA lake
E. Lakehouse
Answer: B (LEAVE A REPLY)

Valid Databricks-Certified-Professional-Data-Engineer Dumps shared by BraindumpsPass.com for Helping Passing Databricks-Certified-Professional-Data-Engineer Exam! BraindumpsPass.com now offer the **newest Databricks-Certified-Professional-Data-Engineer exam dumps**, the BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer exam **questions have been updated** and **answers have been corrected** get the **newest** BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer dumps with Test Engine here:
https://www.braindumpspass.com/Databricks/Databricks-Certified-Professional-Data-Engineer-practice-exam-dumps.html (**217** Q&As Dumps, **40%OFF** **Special Discount:** **Exam-Tests**)

NEW QUESTION: 62

You are asked to create a model to predict the total number of monthly subscribers for a specific magazine.
You are provided with 1 year's worth of subscription and payment data, user demographic data, and 10 years worth of content of the magazine (articles and pictures). Which algorithm is the most appropriate for building a predictive model for subscribers?
A. TF-IDF
B. Decision trees
C. Logistic regression
D. Linear regression
Answer: D (LEAVE A REPLY)

NEW QUESTION: 63
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: (SHOW ANSWER)
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.
Reference:
Databricks Documentation on Delta Lake: Delta Lake Guide
Databricks Documentation on ACID Transactions in Delta Lake: ACID Transactions in Delta Lake

NEW QUESTION: 64
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 (LEAVE A REPLY)
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: 65
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 (LEAVE A REPLY)
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: 66

Which of the following SQL commands are used to append rows to an existing delta table?
A. APPEND INTO DELTA table_name
B. APPEND INTO table_name
C. COPY DELTA INTO table_name
D. INSERT INTO table_name
E. UPDATE table_name
Answer: D (LEAVE A REPLY)
Explanation
The answer is INSERT INTO table_name
Insert adds rows to an existing table, this is very similar to add rows a traditional Database or Da-tawarehouse.

NEW QUESTION: 67
What is a method of installing a Python package scoped at the notebook level to all nodes in the currently active cluster?
A. Use &Pip install in a notebook cell
B. Run source env/bin/activate in a notebook setup script
C. Install libraries from PyPi using the cluster UI
D. Use &sh install in a notebook cell
Answer: (SHOW ANSWER)
In Databricks notebooks, you can use the %pip install command in a notebook cell to install a Python package. This will install the package on all nodes in the currently active cluster at the notebook level. It is a feature provided by Databricks to facilitate the installation of Python libraries for the notebook environment specifically.

NEW QUESTION: 68
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")
    )
```

exam-tests.com

databricks

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. eturn 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")

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

Answer: E (LEAVE A REPLY)

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

NEW QUESTION: 69

A data engineer is configuring Delta Sharing for a Databricks-to-Databricks scenario to optimize read performance. The recipient needs to perform time travel queries and streaming reads on shared sales data.
Which configuration will provide the optimal performance while enabling these capabilities?
A. Share tables WITH HISTORY, ensure tables don't have partitioning enabled, and enable CDF before sharing.
B. Share tables WITHOUT HISTORY and enable partitioning for better query performance.
C. Share the entire schema WITHOUT HISTORY and rely on recipient-side caching for performance.
D. Use the open sharing protocol instead of Databricks-to-Databricks sharing for better performance.
Answer: A (LEAVE A REPLY)
Comprehensive and Detailed Explanation From Exact Extract of Databricks Data Engineer Documents:
The official Delta Sharing guidance specifies that in order for recipients to use time travel queries and streaming reads, providers must share Delta tables WITH HISTORY. Sharing history ensures the Delta log is included, which enables efficient access to table snapshots and incremental data streams. Additionally, Change Data Feed (CDF) must be enabled prior to sharing if downstream consumers require streaming CDC queries. Without history, recipients cannot perform time travel or streaming queries. Open sharing supports static Delta tables but lacks streaming support. Therefore, sharing tables WITH HISTORY and enabling CDF is the required configuration for both performance and functionality.

NEW QUESTION: 70
Direct query on external files limited options, create external tables for CSV files with header and pipe delimited CSV files, fill in the blanks to complete the create table statement CREATE TABLE sales (id int, unitsSold int, price FLOAT, items STRING)

_____

_____
LOCATION "dbfs:/mnt/sales/*.csv"
A. FORMAT CSV
OPTIONS ("true","|")
B. USING CSV
TYPE ("true","|")
C. USING CSV
OPTIONS (header ="true", delimiter = "|")
(Correct)
D. FORMAT CSV
FORMAT TYPE (header ="true", delimiter = "|")
E. FORMAT CSV
TYPE (header ="true", delimiter = "|")
Answer: C (LEAVE A REPLY)
Explanation
Answer is
USING CSV
OPTIONS (header ="true", delimiter = "|")
Here is the syntax to create an external table with additional options
CREATE TABLE table_name (col_name1 col_typ1,..)
USING data_source
OPTIONS (key='value', key2=vla2)
LOCATION = "/location"

NEW QUESTION: 71
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: (SHOW ANSWER)
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: 72
Two data engineers are working on the same Databricks notebook in separate branches. Both have edited the same section of code. When one tries to merge the other's branch into their own using the Databricks Git folders UI, a merge conflict occurs on that notebook file. The UI highlights the conflict and presents options for resolution.
How should the data engineers resolve this merge conflict using Databricks Git folders?
A. Abort the merge, discard all local changes, and try the merge operation again without reviewing the conflicting code.
B. Delete the conflicted notebook file via the Databricks workspace UI, commit the deletion, and recreate the notebook from scratch in a new commit to bypass the conflict entirely.
C. Use the Git CLI in the cluster's web terminal to force-push the conflicted merge (git push -force), overriding the remote branch with the local version and discarding changes.
D. Use the Git folders UI to manually edit the notebook file, selecting the desired lines from both versions and removing the conflict markers, then mark the conflict as resolved.
Answer: (SHOW ANSWER)
Comprehensive and Detailed Explanation From Exact Extract of Databricks Data Engineer Documents:
In the Databricks Git folders integration, when merge conflicts arise in notebooks, the UI provides a visual diff editor that highlights conflicting code segments. Users can manually choose which changes to keep from each branch, edit directly in the notebook UI, and remove conflict markers.
After resolving, the engineer must mark the conflict as resolved, save, and commit the final version.
This process ensures that both contributors' valid code segments are merged correctly and version history is maintained.
Forcing a push (C) or deleting notebooks (B) introduces data loss or versioning issues. Aborting without review (A) violates collaborative best practices. Therefore, D is the only correct and Databricks-approved way to resolve notebook merge conflicts.

NEW QUESTION: 73
Which statement regarding stream-static joins and static Delta tables is correct?
A. Each microbatch of a stream-static join will use the most recent version of the static Delta table as of each microbatch.
B. Each microbatch of a stream-static join will use the most recent version of the static Delta table as of the job's initialization.
C. The checkpoint directory will be used to track state information for the unique keys present in the join.
D. Stream-static joins cannot use static Delta tables because of consistency issues.
E. The checkpoint directory will be used to track updates to the static Delta table.

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

NEW QUESTION: 74
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.



exam-tests.com
databricks

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 (LEAVE A REPLY)

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: 75
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 (LEAVE A REPLY)
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: 76
The data analyst team had put together queries that identify items that are out of stock based on orders and replenishment but when they run all together for final output the team noticed it takes a really long time, you were asked to look at the reason why queries are running slow and identify steps to improve the performance and when you looked at it you noticed all the code queries are running sequentially and using a SQL endpoint cluster. Which of the following steps can be taken to resolve the issue?
Here is the example query
1.--- Get order summary
2.create or replace table orders_summary
3.as
4.select product_id, sum(order_count) order_count
5.from
6. (
7. select product_id,order_count from orders_instore

```
8. union all
9. select product_id,order_count from orders_online
10. )
11.group by product_id
12.-- get supply summary
13.create or repalce tabe supply_summary
14.as
15.select product_id, sum(supply_count) supply_count
16.from supply
17.group by product_id
18.
19.-- get on hand based on orders summary and supply summary
20.
21.with stock_cte
22.as (
23.select nvl(s.product_id,o.product_id) as product_id,
24. nvl(supply_count,0) - nvl(order_count,0) as on_hand
25.from supply_summary s
26.full outer join orders_summary o
27. on s.product_id = o.product_id
28.)
29.select *
30.from
31.stock_cte
32.where on_hand = 0
```

A. Turn on the Serverless feature for the SQL endpoint.

B. Increase the maximum bound of the SQL endpoint's scaling range.

C. Increase the cluster size of the SQL endpoint.

D. Turn on the Auto Stop feature for the SQL endpoint.

E. Turn on the Serverless feature for the SQL endpoint and change the Spot Instance Pol-icy to "Reliability Optimized."

Answer: C (LEAVE A REPLY)

Explanation

The answer is to increase the cluster size of the SQL Endpoint, here queries are running sequentially and since the single query can not span more than one cluster adding more clusters won't improve the query but rather increasing the cluster size will improve performance so it can use additional compute in a warehouse.

In the exam please note that additional context will not be given instead you have to look for cue words or need to understand if the queries are running sequentially or concurrently. if the que-ries are running sequentially then scale up(more nodes) if the queries are running concurrently (more users) then scale out(more clusters).

Below is the snippet from Azure, as you can see by increasing the cluster size you are able to add more worker nodes.

Cluster size	Instance type for driver(applies only to Classic Warehouses/SQL Endpoints)	Worker count
2X-Small	Standard_E8ds_v4	1
X-Small		2
Small	Standard_E16ds_v4	4
Medium	Standard_E32ds_v4	8
Large		16
X-Large	Standard_E64ds_v4	32
2X-Large		64
3X-Large		128
4X-Large		256

*The instance size of all workers is Standard_E8ds_v4.

exam-tests.com

databricks

SQL endpoint scales horizontally(scale-out) and vertically (scale-up), you have to understand when to use what.

Scale-up-> Increase the size of the cluster from x-small to small, to medium, X Large....

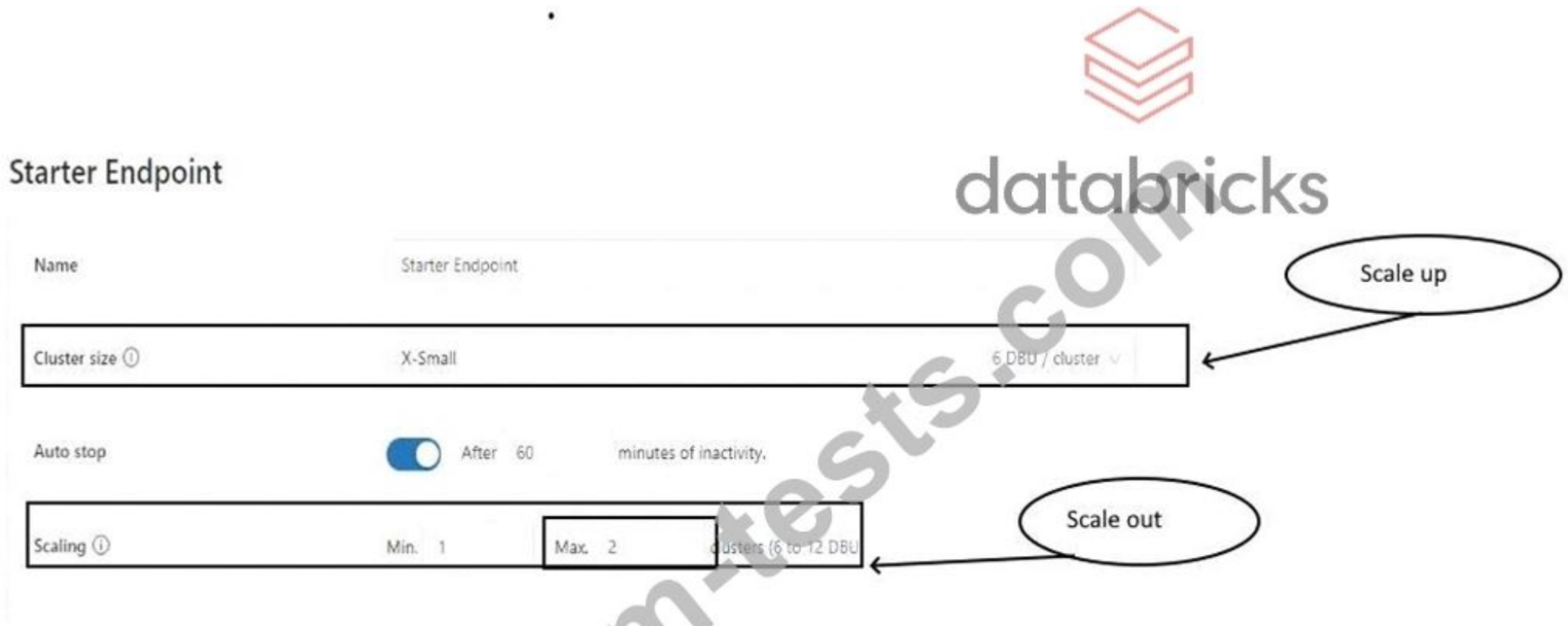
If you are trying to improve the performance of a single query having additional memory, additional nodes and cpu in the cluster will improve the performance.

Scale-out -> Add more clusters, change max number of clusters

If you are trying to improve the throughput, being able to run as many queries as possible then having an additional cluster(s) will improve the performance.

SQL endpoint

A picture containing diagram Description automatically generated



Valid Databricks-Certified-Professional-Data-Engineer Dumps shared by BraindumpsPass.com for Helping Passing Databricks-Certified-Professional-Data-Engineer Exam! BraindumpsPass.com now offer the **newest Databricks-Certified-Professional-Data-Engineer exam dumps**, the BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer exam **questions have been updated** and **answers have been corrected** get the **newest** BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer dumps with Test Engine here:

https://www.braindumpspass.com/Databricks/Databricks-Certified-Professional-Data-Engineer-practice-exam-dumps.html (**217** Q&As Dumps, **40%OFF** **Special Discount:** **Exam-Tests**)

NEW QUESTION: 77
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"))
```

exam-tests.com



databricks

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 (LEAVE A REPLY)

The provided PySpark code performs the following operations:

* Reads Data from silver_customer_sales Table:

* The code starts by accessing the silver_customer_sales table using the spark.table method.
* Groups Data by customer_id:
* The .groupBy(" customer_id ") function groups the data based on the customer_id column.
* Aggregates Data:
* The .agg() function computes several aggregate metrics for each customer_id:
* F.min(" sale_date ").alias(" first_transaction_date "): Determines the earliest sale date for the customer.
* F.max(" sale_date ").alias(" last_transaction_date "): Determines the latest sale date for the customer.
* F.mean(" sale_total ").alias(" average_sales "): Calculates the average sale amount for the customer.
* F.countDistinct(" order_id ").alias(" total_orders "): Counts the number of unique orders placed by the customer.
* F.sum(" sale_total ").alias(" lifetime_value "): Calculates the total sales amount (lifetime value) for the customer.
* Writes Data to gold_customer_lifetime_sales_summary Table:
* The .write.mode(" overwrite ").table(" gold_customer_lifetime_sales_summary ") command writes the aggregated data to the gold_customer_lifetime_sales_summary table.
* The mode(" overwrite ") specifies that the existing data in the
gold_customer_lifetime_sales_summary table will be completely replaced by the new aggregated data.
Conclusion:
When this code is executed, it reads all records from the silver_customer_sales table, performs the specified aggregations grouped by customer_id, and then overwrites the entire gold_customer_lifetime_sales_summary table with the aggregated results. Therefore, option D accurately describes this process: " The gold_customer_lifetime_sales_summary table will be overwritten by aggregated values calculated from all records in the silver_customer_sales table as a batch job. " References:
PySpark DataFrame groupBy
PySpark Basics

NEW QUESTION: 78
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 (LEAVE A REPLY)
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 Reference: [Databricks Certified Data Engineer Professional], under "Databricks Workspace" section; Databricks Documentation, under "List secrets" section.

NEW QUESTION: 79
A table nameduser_ltvis 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.
Theuser_ltvtable 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
```

examtests.com



databricks

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, age. and ltv columns will be returned with the values in user ltv.
E. Only the email and ltv columns will be returned; the email column will contain the string "REDACTED" in each row.
Answer: E (LEAVE A REPLY)
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: 80
A transactions table has been liquid clustered on the columns product_id, user_id, and event_date.
Which operation lacks support for cluster on write?
A. spark.writestream.format('delta').mode('append')
B. CTAS and RTAS statements
C. INSERT INTO operations
D. spark.write.format('delta').mode('append')
Answer: (SHOW ANSWER)
Delta Lake'sLiquid Clusteringis an advanced feature that improves query performance by dynamically clustering data without requiring costly compaction steps like traditional Z-ordering.
When performing writes to aLiquid Clusteredtable, some write operations automatically maintain clustering, while othersdo not.
Explanation of Each Option:
* (A) spark.writestream.format('delta').mode('append') (Correct Answer)
* Reason:Streaming writes (writestream) donotsupportLiquid Clusteringbecause streaming data arrives in micro-batches.
* Since Liquid Clustering needs efficient global reorganization of files, streaming append operations don't provide sufficient data volume at a time to be effectively clustered.
* Delta Lake documentation states that Liquid Clustering is only supported for batch writes.

* (B) CTAS and RTAS statements
* Reason:CREATE TABLE AS SELECT (CTAS) and REPLACE TABLE AS SELECT (RTAS) are batch operationsand can enforce Liquid Clustering.
* These operations create or replace a table based on a query result, and since they are batch-based, Liquid Clustering applies.
* (C) INSERT INTO operations
* Reason:INSERT INTOis supportedfor Liquid Clustering because it is a batch operation.
* While it may not be as efficient as MERGE or COPY INTO, clustering is applied upon execution.
* (D) spark.write.format('delta').mode('append')
* Reason:Batch append operationsare supportedfor Liquid Clustering.
* Unlike streaming append, batch writes allow the optimizer to re-cluster data efficiently.
Conclusion:
Sincestreaming append operations do not support Liquid Clustering, option(A)is the correct answer.
References:
* Liquid Clustering in Delta Lake - Databricks Documentation

NEW QUESTION: 81
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 compaction C 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 (LEAVE A REPLY)
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.

NEW QUESTION: 82
How does Lakehouse replace the dependency on using Data lakes and Data warehouses in a Data and Analytics solution?
A. Open, direct access to data stored in standard data formats.
B. Supports ACID transactions.
C. Supports BI and Machine learning workloads
D. Support for end-to-end streaming and batch workloads
E. All the above
Answer: E (LEAVE A REPLY)
Explanation
Lakehouse combines the benefits of a data warehouse and data lakes,
Lakehouse = Data Lake + DataWarehouse

Here are some of the major benefits of a lakehouse

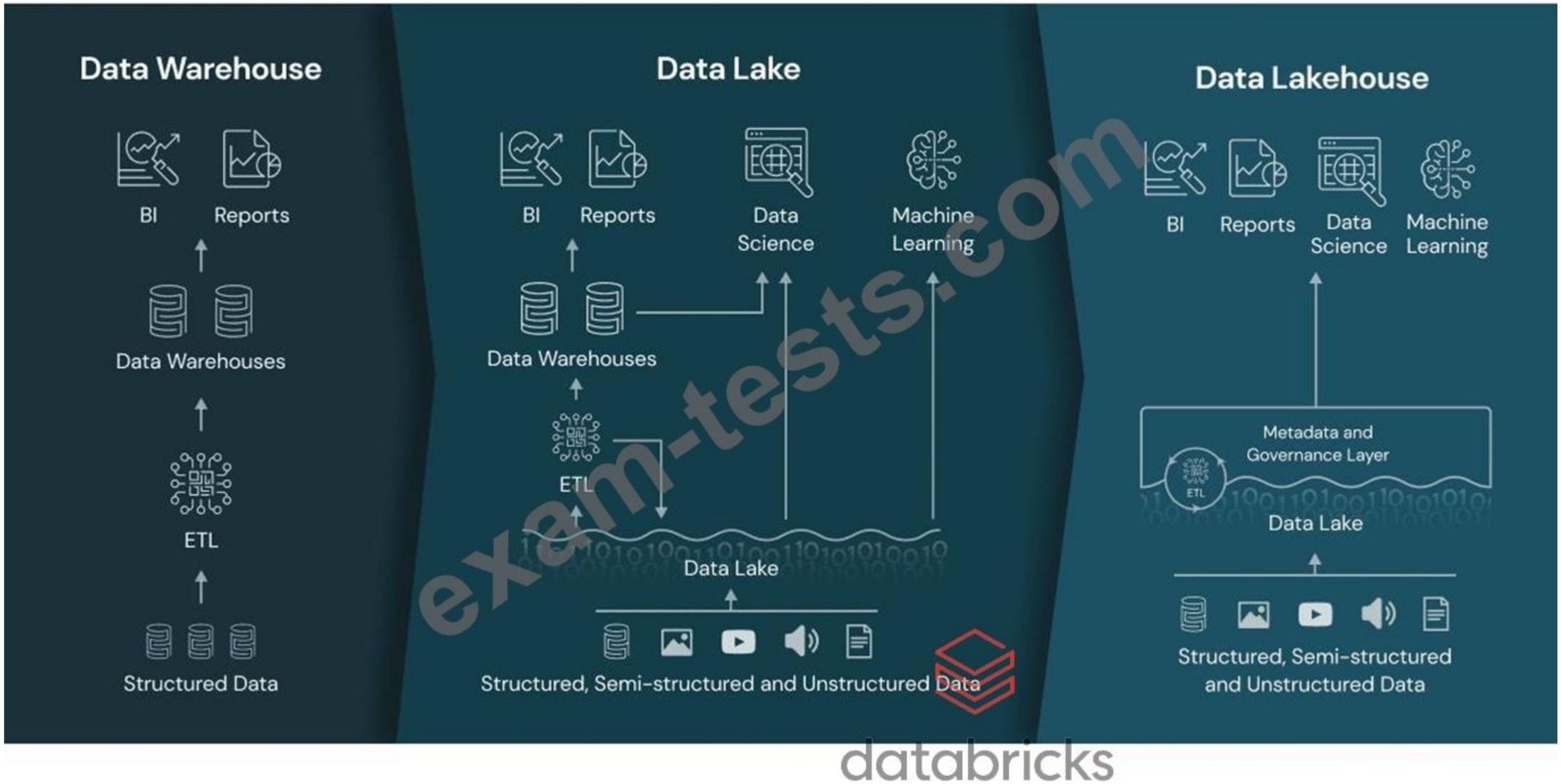
Text, letter Description automatically generated

A lakehouse has the following key features:

- **Transaction support:** In an enterprise lakehouse many data pipelines will often be reading and writing data concurrently. Support for ACID transactions ensures consistency as multiple parties concurrently read or write data, typically using SQL.
- **Schema enforcement and governance:** The Lakehouse should have a way to support schema enforcement and evolution, supporting DW schema architectures such as star/snowflake-schemas. The system should be able to reason about data integrity, and it should have robust governance and auditing mechanisms.
- **BI support:** Lakehouses enable using BI tools directly on the source data. This reduces staleness and improves recency, reduces latency, and lowers the cost of having to operationalize two copies of the data in both a data lake and a warehouse.
- **Storage is decoupled from compute:** In practice this means storage and compute use separate clusters, thus these systems are able to scale to many more concurrent users and larger data sizes. Some modern data warehouses also have this property.
- **Openness:** The storage formats they use are open and standardized, such as Parquet, and they provide an API so a variety of tools and engines, including machine learning and Python/R libraries, can efficiently access the data **directly**.
- **Support for diverse data types ranging from unstructured to structured data**: The lakehouse can be used to store, refine, analyze, and access data types needed for many new data applications, including images, video, audio, semi-structured data, and text.
- **Support for diverse workloads:** including data science, machine learning, and SQL and analytics. Multiple tools might be needed to support all these workloads but they all rely on the same data repository.
- **End-to-end streaming:** Real-time reports are the norm in many enterprises. Support for streaming eliminates the need for separate systems dedicated to serving real-time data applications.

databricks
Exam-tests.com

Lakehouse = Data Lake + DataWarehouse
A picture containing text, blackboard Description automatically generated



NEW QUESTION: 83
A data engineer is optimizing a managed Delta table that suffers from data skew and frequently changing query filter columns. The engineer wants to avoid costly data rewrites when query patterns evolve. The table size is under 1 TB.
How should the data engineer meet this requirement?
A. Apply Z-ordering, since it allows flexible reorganization of data layout without rewriting existing files and adapts easily to new filter columns.
B. Use Hive-style partitioning, as it provides efficient data skipping and is easy to change partition columns at any time.
C. Enable liquid clustering, as it efficiently handles data skew, allows clustering keys to be changed without rewriting existing data, and adapts to evolving query patterns.
D. Combine partitioning and Z-ordering to maximize flexibility and minimize maintenance as query patterns change.

Answer: C (LEAVE A REPLY)
Comprehensive and Detailed Explanation From Exact Extract of Databricks Data Engineer Documents:
The Databricks documentation describes Liquid Clustering as the recommended data layout optimization for evolving workloads. Unlike traditional partitioning or Z-ordering, Liquid Clustering dynamically maintains data organization without rewriting existing files when clustering keys change. It handles data skew automatically and supports flexible re-clustering based on query patterns. Partitioning and Z-ordering require full data rewrites whenever key structures change, making them expensive for tables with frequently evolving access patterns. For tables under a few terabytes, Liquid Clustering offers the best balance between scalability, adaptability, and maintenance efficiency. Thus, option C aligns with Databricks' best practice for modern adaptive layout optimization.

NEW QUESTION: 84
What is the best way to query external csv files located on DBFS Storage to inspect the data using SQL?
A. SELECT * FROM 'dbfs:/location/csv_files/' FORMAT = 'CSV'
B. SELECT CSV. * from 'dbfs:/location/csv_files/'
C. SELECT * FROM CSV. 'dbfs:/location/csv_files/'
D. You can not query external files directly, us COPY INTO to load the data into a table first
E. SELECT * FROM 'dbfs:/location/csv_files/' USING CSV
Answer: C (LEAVE A REPLY)
Explanation
Answer is, SELECT * FROM CSV. 'dbfs:/location/csv_files/'
you can query external files stored on the storage using below syntax
SELECT * FROM format.`/Location`
format - CSV, JSON, PARQUET, TEXT

NEW QUESTION: 85
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: "DESW69", count: 1}]             "u1@domain.com"
1002[{id: "KYBD48", 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.



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: (SHOW ANSWER)
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.
Reference:

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

NEW QUESTION: 86
How are the operational aspects of Lakeflow Declarative Pipelines different from Spark Structured Streaming?
A. Lakeflow Declarative Pipelines manage the orchestration of multi-stage pipelines automatically, while Structured Streaming requires external orchestration for complex dependencies.
B. Structured Streaming can process continuous data streams, while Lakeflow Declarative Pipelines cannot.
C. Lakeflow Declarative Pipelines can write to Delta Lake format, while Structured Streaming cannot.
D. Lakeflow Declarative Pipelines automatically handle schema evolution, while Structured Streaming always requires manual schema management.
Answer: (SHOW ANSWER)
Comprehensive and Detailed Explanation From Exact Extract of Databricks Data Engineer Documents:
Databricks documentation explains that Lakeflow Declarative Pipelines build upon Structured Streaming but add higher-level orchestration and automation capabilities. They automatically manage dependencies, materialization, and recovery across multi-stage data flows without requiring external orchestration tools such as Airflow or Azure Data Factory. In contrast, Structured Streaming operates at a lower level, where developers must manually handle orchestration, retries, and dependencies between streaming jobs. Both support Delta Lake outputs and schema evolution; however, Lakeflow Declarative Pipelines simplify management by declaratively defining transformations and data quality expectations. Hence, the correct distinction is A - automated orchestration and management in Lakeflow Declarative Pipelines.

NEW QUESTION: 87
A newly joined team member John Smith in the Marketing team currently has access read access to sales tables but does not have access to update the table, which of the following commands help you accomplish this?
A. GRANT UPDATE ON TABLE table_name TO john.smith@marketing.com
B. GRANT USAGE ON TABLE table_name TO john.smith@marketing.com
C. GRANT MODIFY ON TABLE table_name TO john.smith@marketing.com
D. GRANT UPDATE TO TABLE table_name ON john.smith@marketing.com
E. GRANT MODIFY TO TABLE table_name ON john.smith@marketing.com
Answer: (SHOW ANSWER)
Explanation
The answer is GRANT MODIFY ON TABLE table_name TO john.smith@marketing.com
https://docs.microsoft.com/en-us/azure/databricks/security/access-control/table-acls/object-privileges#privileges

NEW QUESTION: 88
A Delta Lake table representing metadata about content posts from users has the following schema:
user_id LONG, post_text STRING, post_id STRING, longitude FLOAT, latitude FLOAT, post_time TIMESTAMP, date DATE This table is partitioned by the date column. A query is run with the following filter: longitude < 20 & longitude > -20
Which statement describes how data will be filtered?
A. Statistics in the Delta Log will be used to identify partitions that might Include files in the filtered range.
B. No file skipping will occur because the optimizer does not know the relationship between the partition column and the longitude.
C. The Delta Engine will use row-level statistics in the transaction log to identify the flies that meet the filter criteria.
D. Statistics in the Delta Log will be used to identify data files that might include records in the filtered range.
E. The Delta Engine will scan the parquet file footers to identify each row that meets the filter criteria.
Answer: D (LEAVE A REPLY)
Explanation

This is the correct answer because it describes how data will be filtered when a query is run with the following filter: longitude < 20 & longitude > -20. The query is run on a Delta Lake table that has the following schema:
user_id LONG, post_text STRING, post_id STRING, longitude FLOAT, latitude FLOAT, post_time TIMESTAMP, date DATE. This table is partitioned by the date column. When a query is run on a partitioned Delta Lake table, Delta Lake uses statistics in the Delta Log to identify data files that might include records in the filtered range. The statistics include information such as min and max values for each column in each data file. By using these statistics, Delta Lake can skip reading data files that do not match the filter condition, which can improve query performance and reduce I/O costs. Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Data skipping" section.

NEW QUESTION: 89
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 namedstore_saies_summaryand the schema is as follows:

```
store_id INT, total_sales_qtd FLOAT, avg_daily_sales_qtd FLOAT,  total_sales_ytd FLOAT,
avg_daily_sales_ytd FLOAT, previous_day_sales FLOAT, total_sales_7d FLOAT, avg_daily_sales_7d
FLOAT, updated TIMESTAMP
```

The tabledaily_store_salescontains all the information needed to updatestore_sales_summary. The schema for this table is:
store_id INT, sales_date DATE, total_sales FLOAT
Ifdaily_store_salesis implemented as a Type 1 table and thetotal_salescolumn might be adjusted after manual data auditing, which approach is the safest to generate accurate reports in thestore_sales_summarytable?
A. Implement the appropriate aggregate logic as a batch read against the daily_store_sales table 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 (LEAVE A REPLY)
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: 90
Which of the following statements describes Delta Lake?

A. Delta Lake is an open source platform to help manage the complete machine learning lifecycle
B. Delta Lake is an open format storage layer that delivers reliability, security, and per-formance
C. Delta Lake is an open source data storage format for distributed data
D. Delta Lake is an open source analytics engine used for big data workloads
E. Delta Lake is an open format storage layer that processes data
Answer: B (LEAVE A REPLY)
Explanation
Delta Lake

NEW QUESTION: 91
Which REST API call can be used to review the notebooks configured to run as tasks in a multi-task job?
A. /jobs/runs/list
B. /jobs/runs/get-output
C. /jobs/runs/get
D. /jobs/get
E. /jobs/list
Answer: D (LEAVE A REPLY)
Explanation
This is the correct answer because it is the REST API call that can be used to review the notebooks configured to run as tasks in a multi-task job. The REST API is an interface that allows programmatically interacting with Databricks resources, such as clusters, jobs, notebooks, or tables. The REST API uses HTTP methods, such as GET, POST, PUT, or DELETE, to perform operations on these resources. The /jobs/get endpoint is a GET method that returns information about a job given its job ID. The information includes the job settings, such as the name, schedule, timeout, retries, email notifications, and tasks. The tasks are the units of work that a job executes. A task can be a notebook task, which runs a notebook with specified parameters; a jar task, which runs a JAR uploaded to DBFS with specified main class and arguments; or a python task, which runs a Python file uploaded to DBFS with specified parameters. A multi-task job is a job that has more than one task configured to run in a specific order or in parallel. By using the /jobs/get endpoint, one can review the notebooks configured to run as tasks in a multi-task job. Verified References: [Databricks Certified Data Engineer Professional], under "Databricks Jobs" section; Databricks Documentation, under "Get" section; Databricks Documentation, under "JobSettings" section.

Valid Databricks-Certified-Professional-Data-Engineer Dumps shared by BraindumpsPass.com for Helping Passing Databricks-Certified-Professional-Data-Engineer Exam! BraindumpsPass.com now offer the **newest Databricks-Certified-Professional-Data-Engineer exam dumps**, the BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer exam **questions have been updated** and **answers have been corrected** get the **newest** BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer dumps with Test Engine here:
https://www.braindumpspass.com/Databricks/Databricks-Certified-Professional-Data-Engineer-practice-exam-dumps.html (**217** Q&As Dumps, **40%OFF** **Special Discount:** **Exam-Tests**)

NEW QUESTION: 92
Each configuration below is identical to the extent that each cluster has 400 GB total of RAM, 160 total cores and only one Executor per VM.
Given a job with at least one wide transformation, which of the following cluster configurations will result in maximum performance?
A. * Total VMs; 1
* 400 GB per Executor
* 160 Cores / Executor
B. * Total VMs: 8
* 50 GB per Executor

* 20 Cores / Executor
C. * Total VMs: 4
* 100 GB per Executor
* 40 Cores/Executor
D. * Total VMs:2
* 200 GB per Executor
* 80 Cores / Executor
Answer: B (LEAVE A REPLY)
Explanation
This is the correct answer because it is the cluster configuration that will result in maximum performance for a job with at least one wide transformation. A wide transformation is a type of transformation that requires shuffling data across partitions, such as join, groupBy, or orderBy. Shuffling can be expensive and time-consuming, especially if there are too many or too few partitions. Therefore, it is important to choose a cluster configuration that can balance the trade-off between parallelism and network overhead. In this case, having 8 VMs with 50 GB per executor and 20 cores per executor will create 8 partitions, each with enough memory and CPU resources to handle the shuffling efficiently. Having fewer VMs with more memory and cores per executor will create fewer partitions, which will reduce parallelism and increase the size of each shuffle block. Having more VMs with less memory and cores per executor will create more partitions, which will increase parallelism but also increase the network overhead and the number of shuffle files. Verified References: [Databricks Certified Data Engineer Professional], under "Performance Tuning" section; Databricks Documentation, under "Cluster configurations" section.

NEW QUESTION: 93
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. Events are recorded once per minute per device.
df has the following schema: device_id INT, event_time TIMESTAMP, temp FLOAT, humidity FLOAT Code block:

```
df.withWatermark("event_time", "10 minutes")
.groupBy(
_____,
"device_id"
)
.agg(
avg("temp").alias("avg_temp"),
avg("humidity").alias("avg_humidity")
)
.writeStream
.format("delta")
.saveAsTable("sensor_avg")
```

Which line of code correctly fills in the blank within the code block to complete this task?
A. window("event_time", "5 minutes").alias("time")
B. to_interval("event_time", "5 minutes").alias("time")
C. "event_time"
D. lag("event_time", "5 minutes").alias("time")
Answer: A (LEAVE A REPLY)
Comprehensive and Detailed Explanation From Exact Extract:
Exact extract: "window(timeColumn, windowDuration[, slideDuration]) returns a time window for grouping on event-time." Exact extract: "If slideDuration is not specified the windows are non-overlapping (tumbling) with length windowDuration." Exact extract: "Watermarking on the event-time column is required with aggregations to bound state and handle late data."

NEW QUESTION: 94
A data engineer is tasked with building a nightly batch ETL pipeline that processes very large volumes of raw JSON logs from a data lake into Delta tables for reporting. The data arrives in bulk once per day, and the pipeline takes several hours to complete. Cost efficiency is important, but performance and reliability of completing the pipeline are the highest priorities.
Which type of Databricks cluster should the data engineer configure?
A. A lightweight single-node cluster with low worker node count to reduce costs.
B. A high-concurrency cluster designed for interactive SQL workloads.
C. An all-purpose cluster always kept running to ensure low-latency job startup times.
D. A job cluster configured to autoscale across multiple workers during the pipeline run.
Answer: D (LEAVE A REPLY)
Job clusters are optimized for automated production workloads. They start when a job is triggered and terminate automatically once the task completes. This ensures cost control while maintaining performance and reliability for batch ETL. Autoscaling allows Databricks to add or remove workers dynamically based on workload size, ensuring large data volumes are processed efficiently.
All-purpose clusters are intended for development or ad-hoc workloads, not scheduled ETL.
Reference Source: Databricks Compute and Job Cluster Configuration Documentation - "Autoscaling and Job Clusters."

NEW QUESTION: 95
A data engineer is using a Databricks SQL query to monitor the performance of an ELT job. The ELT job is triggered by a specific number of input records being ready to process. The Databricks SQL query returns the number of minutes since the job's most recent runtime. Which of the following approaches can enable the data engineering team to be notified if the ELT job has not been run in an hour?
A. They can set up an Alert for the accompanying dashboard to notify them if the returned value is greater than 60.
B. They can set up an Alert for the query to notify when the ELT job fails.
C. They can set up an Alert for the accompanying dashboard to notify when it has not re-freshed in 60 minutes.
D. They can set up an Alert for the query to notify them if the returned value is greater than 60.
E. This type of alert is not possible in Databricks
Answer: D (LEAVE A REPLY)
Explanation
The answer is, They can set up an Alert for the query to notify them if the returned value is greater than 60.
The important thing to note here is that alert can only be setup on query not on the dashboard, query can return a value, which is used if alert can be triggered.

NEW QUESTION: 96
A data organization has adopted Delta Sharing to securely distribute curated datasets from a Unity Catalog-enabled workspace. The data engineering team shares large Delta tables internally via Databricks-to-Databricks and externally via Open Sharing for aggregated reports. While testing, they encounter challenges related to access control, data update visibility, and shareable object types.
What is a limitation of the Delta Sharing protocol or implementation when used with Databricks-to-Databricks or Open Sharing?
A. With Open Sharing, recipients cannot access Volumes, Models, or notebooks - only static Delta tables are supported.
B. Delta Sharing does not support Unity Catalog-enabled tables; only legacy Hive Metastore tables are shareable.
C. With Databricks-to-Databricks sharing, Unity Catalog recipients must re-ingest data manually using COPY INTO or REST APIs.
D. Delta Sharing (both Databricks-to-Databricks and Open Sharing) allows recipients to modify the source data if they have select privileges.
Answer: A (LEAVE A REPLY)
Comprehensive and Detailed Explanation From Exact Extract of Databricks Data Engineer Documents:
According to Databricks' documentation, Open Sharing allows secure sharing of Delta tables to any recipient via a REST-based protocol without requiring a Databricks account. However, the Open Sharing protocol is limited to static Delta tables-it does not support sharing of Unity Catalog objects like Volumes, Machine Learning models, or notebooks. Only Databricks-to-Databricks sharing supports dynamic

data sharing with update visibility and streaming reads. Thus, the inability to share non-table objects in Open Sharing represents a known limitation of the protocol. Option A accurately reflects this constraint as described in Delta Sharing design principles and documentation.

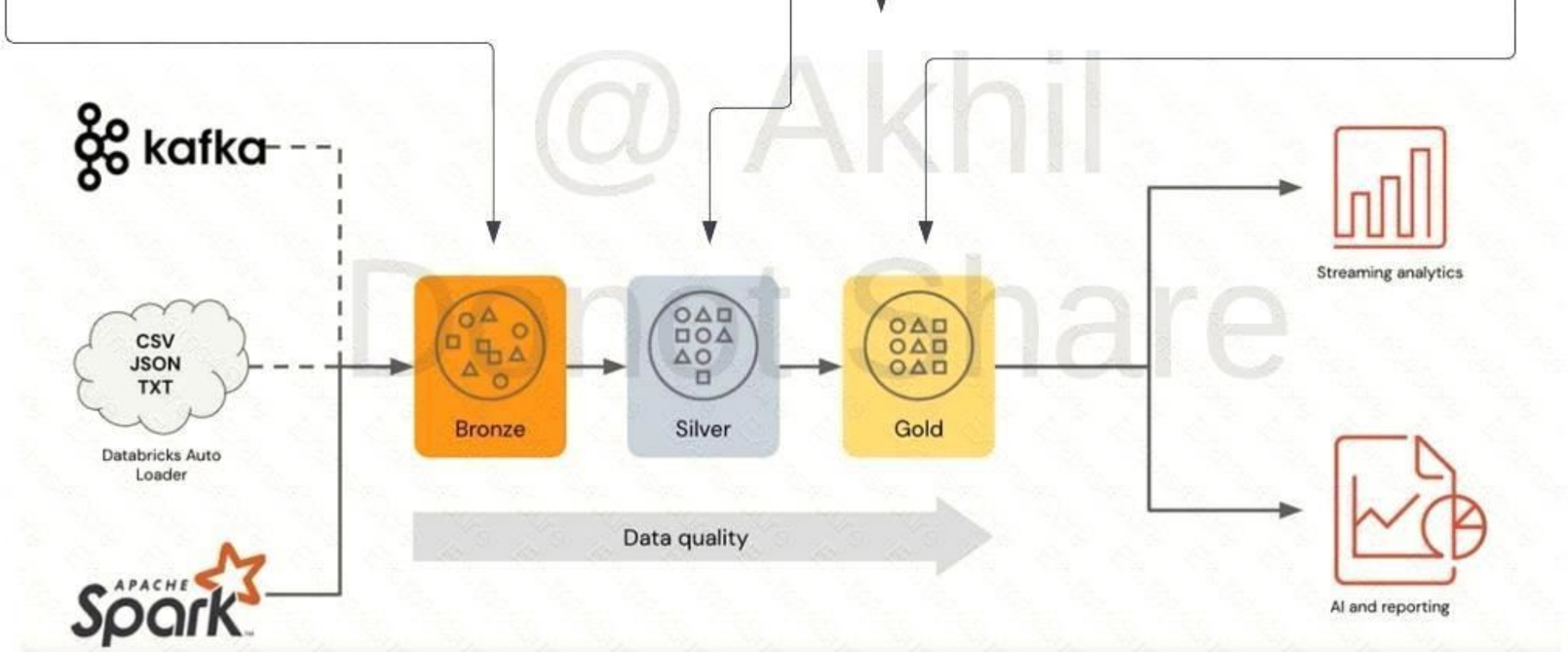
NEW QUESTION: 97
A Delta Lake table in the Lakehouse named customer_parsams is used in churn prediction by the machine learning team. The table contains information about customers derived from a number of upstream sources.
Currently, the data engineering team populates this table nightly by overwriting 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: (SHOW ANSWER)
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: 98
What is the main difference between the bronze layer and silver layer in a medallion architecture?
A. Duplicates are removed in bronze, schema is applied in silver
B. Silver may contain aggregated data
C. Bronze is raw copy of ingested data, silver contains data with production schema and optimized for ELT/ETL throughput
D. Bad data is filtered in Bronze, silver is a copy of bronze data
Answer: (SHOW ANSWER)
Explanation
Medallion Architecture - Databricks
Exam focus: Please review the below image and understand the role of each layer(bronze, silver, gold) in medallion architecture, you will see varying questions targeting each layer and its purpose.
Sorry I had to add the watermark some people in Udemy are copying my content.
A diagram of a house Description automatically generated with low confidence

Bronze Layer:
1.Raw copy of ingested data
2.Replaces traditional data lake
3.Provides efficient storage and querying of full, unprocessed history of data
4.No schema is applied at this layer

Silver Layer:
1.Reduces data storage complexity, latency, and redundency
2.Optimizes ETL throughput and analytic query performance
3.Preserves grain of original data (without aggregation)
4.Eliminates duplicate records
5.Production schema enforced
6.Data quality checks, quarantine corrupt data

Gold Layer:
1.Powers Ml applications, reporting, dashboards, ad hoc analytics
2.Refined views of data, typically with aggregations
3.Reduces strain on production systems
4.Optimizes query performance for business-critical data



@ Akhil Don't Share

NEW QUESTION: 99

To identify the top users consuming compute resources, a data engineering team needs to monitor usage within their Databricks workspace for better resource utilization and cost control. The team decided to use Databricks system tables, available under the System catalog in Unity Catalog, to gain detailed visibility into workspace activity.
Which SQL query should the team run from the System catalog to achieve this?
A. SELECT sku_name,
identity_metadata.created_by AS user_email,
COUNT(usage_quantity) AS total_dbus
FROM system.billing.usage
GROUP BY user_email, sku_name
ORDER BY total_dbus DESC
LIMIT 10
B. SELECT identity_metadata.run_as AS user_email,
SUM(usage_quantity) AS total_dbus
FROM system.billing.usage
GROUP BY user_email
ORDER BY total_dbus DESC
LIMIT 10
C. SELECT sku_name,
identity_metadata.created_by AS user_email,
SUM(usage_quantity * usage_unit) AS total_dbus
FROM system.billing.usage
GROUP BY user_email, sku_name
ORDER BY total_dbus DESC
LIMIT 10
D. SELECT sku_name,
usage_metadata.run_name AS user_email,
SUM(usage_quantity) AS total_dbus
FROM system.billing.usage
GROUP BY user_email, sku_name
ORDER BY total_dbus DESC
LIMIT 10
Answer: B (LEAVE A REPLY)
The system.billing.usage table in the Unity Catalog System schema provides detailed usage metrics for each workload in the workspace. The field identity_metadata.run_as identifies the user or service principal under which the job or query executed. Summing usage_quantity provides total DBU (Databricks Unit) consumption per user. According to Databricks documentation, this table is the authoritative source for monitoring workspace cost drivers, showing compute SKU, user, and DBU consumption over time. Grouping by identity_metadata.run_as and summing usage_quantity produces the correct aggregation to determine top users. Other queries use non-existent or incorrect fields (created_by, run_name, or multiplied usage quantities), which do not reflect actual billing metrics.

NEW QUESTION: 100
While reviewing a query's execution in the Databricks Query Profiler, a data engineer observes that the Top Operators panel shows a Sort operator with high Time Spent and Memory Peak metrics. The Spark UI also reports frequent data spilling.
How should the data engineer address this issue?
A. Switch to a broadcast join to reduce memory usage.

B. Repartition the DataFrame to a single partition before sorting.
C. Convert the sort operation to a filter operation.
D. Increase the number of shuffle partitions to better distribute data.
Answer: (SHOW ANSWER)
Comprehensive and Detailed Explanation From Exact Extract of Databricks Data Engineer Documents:
When Spark performs wide transformations such as sortBy or orderBy, large data volumes can exceed memory limits, causing disk spilling. The official Databricks performance tuning guide recommends increasing the shuffle partition count to distribute the data more evenly across executors. By default, Spark uses a fixed number of shuffle partitions (e.g., 200), which can lead to memory imbalance and spill if some partitions are too large. Increasing this number (via spark.sql.shuffle.partitions) results in smaller partitions, reduced in-memory pressure, and improved sort performance. Other options like broadcast joins or single partition sorts do not apply to single-table sorts, and converting to filters changes query logic. Thus, option D is the correct remedy.

NEW QUESTION: 101
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 nameddev-2.3.9is 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 (LEAVE A REPLY)
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: 102
A senior data engineer is planning large-scale data workflows. The task is to identify the considerations that form a foundation for creating scalable data models for managing large datasets. The team has listed Delta Lake capabilities and wants to determine which feature should not be considered as a core factor.
Which key feature can be ignored while evaluating Delta Lake?
A. Delta Lake's ability to process data in both batch and streaming modes seamlessly, providing flexibility in ingestion and processing.
B. Delta Lake works with various data formats (Parquet, JSON, CSV) and integrates well with Spark and Databricks tools.
C. Delta Lake optimizes metadata handling, efficiently managing billions of files and facilitating scalability to petabyte-scale datasets.
D. Delta Lake provides limited support for monitoring and troubleshooting data pipelines, so relevant partner tools have to be identified and set up for enhanced operational efficiency.
Answer: (SHOW ANSWER)
The Databricks documentation emphasizes that Delta Lake provides robust capabilities for batch and streaming unification, scalable metadata management, and seamless integration with Databricks and Apache Spark. These are core design principles for building modern, scalable data platforms. However, the statement claiming that Delta provides "limited support for monitoring and troubleshooting" is not valid . Delta Lake itself includes comprehensive logging, event history tracking, and data lineage via the transaction log, and integrates with Databricks' built-in monitoring tools. Hence, such a limitation is not a design consideration when evaluating Delta Lake's scalability. Therefore, this statement (D) can be ignored, as it does not reflect Delta Lake's documented capabilities.

NEW QUESTION: 103

A data engineer is running a groupBy aggregation on a massive user activity log grouped by user_id. A few users have millions of records, causing task skew and long runtimes.
Which technique will fix the skew in this aggregation?
A. Use salting by adding a random prefix to skewed keys before aggregation, then aggregate again after removing the prefix.
B. Increase the Spark driver memory and retry.
C. Use reduceByKey instead of groupBy to avoid shuffles.
D. Filter out the skewed users before the aggregation.
Answer: (SHOW ANSWER)
Task skew occurs when a small subset of keys holds a disproportionate amount of data, causing certain tasks to process significantly more records than others. Databricks documentation recommends salting as an effective mitigation technique.
Salting introduces a random or calculated prefix to skewed keys, distributing records across multiple partitions and balancing the workload during the shuffle stage. After aggregation, a second pass re-aggregates results by removing the prefix to restore key integrity.
Increasing memory (B) does not resolve distribution imbalance; reduceByKey (C) still triggers shuffles; and filtering (D) would remove valid business data. Hence, salting is the correct and officially recommended approach to address skew in Spark aggregations.

NEW QUESTION: 104
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: (SHOW ANSWER)
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.
Reference: Databricks documentation on Delta Lake optimization techniques, including data skipping and statistics collection (https://docs.databricks.com/delta/optimizations/index.html).

NEW QUESTION: 105
A Delta Lake table was created with the below query:

```
CREATE TABLE prod.sales_by_store
AS (
  SELECT *
  FROM prod.sales a
  INNER JOIN prod.store b
  ON a.store_id = b.store_id
)
```

examtests.com
databricks

Consider the following query:
DROP TABLE prod.sales_by_store -
If this statement is executed by a workspace admin, which result will occur?
A. Nothing will occur until a COMMIT command is executed.
B. The table will be removed from the catalog but the data will remain in storage.
C. The table will be removed from the catalog and the data will be deleted.
D. An error will occur because Delta Lake prevents the deletion of production data.
E. Data will be marked as deleted but still recoverable with Time Travel.
Answer: C (LEAVE A REPLY)
Explanation
When a table is dropped in Delta Lake, the table is removed from the catalog and the data is deleted. This is because Delta Lake is a transactional storage layer that provides ACID guarantees. When a table is dropped, the transaction log is updated to reflect the deletion of the table and the data is deleted from the underlying storage. References:
https://docs.databricks.com/delta/quick-start.html#drop-a-table
https://docs.databricks.com/delta/delta-batch.html#drop-table

NEW QUESTION: 106
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
```
finalDF.explain()
```
Cmd 6
```
display(finalDF)
```
Cmd 7
```
finalDF.write.mode("append").saveAsTable("flat_data")
```
exam-tests.com
databricks

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 (LEAVE A REPLY)
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.

Valid Databricks-Certified-Professional-Data-Engineer Dumps shared by BraindumpsPass.com for Helping Passing Databricks-Certified-Professional-Data-Engineer Exam! BraindumpsPass.com now offer the **newest Databricks-Certified-Professional-Data-Engineer exam dumps**, the BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer exam **questions have been updated** and **answers have been corrected** get the **newest** BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer dumps with Test Engine here:
https://www.braindumpspass.com/Databricks/Databricks-Certified-Professional-Data-Engineer-practice-exam-dumps.html (**217** Q&As Dumps, **40%OFF** **Special Discount:** **Exam-Tests**)

NEW QUESTION: 107
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. Stop the existing pipeline; use the returned settings in a reset command
B. 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
C. 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
D. 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
Answer: B (LEAVE A REPLY)
The Databricks CLI provides a way to automate interactions with Databricks services. When dealing with pipelines, you can use thedatabricks pipelines get --pipeline-idcommand to capture the settings of an existing pipeline in JSON format. This JSON can then be modified by removing thepipeline_idto prevent conflicts and renaming the pipeline to create a new pipeline. The modified JSON file can then be used with the databricks pipelines createcommand to create a new pipeline with those settings.
References:
* Databricks Documentation on CLI for Pipelines: Databricks CLI - Pipelines

NEW QUESTION: 108
You are still noticing slowness in query after performing optimize which helped you to resolve the small files problem, the column(transactionId) you are using to filter the data has high cardinality and auto incrementing number. Which delta optimization can you enable to filter data effectively based on this column?

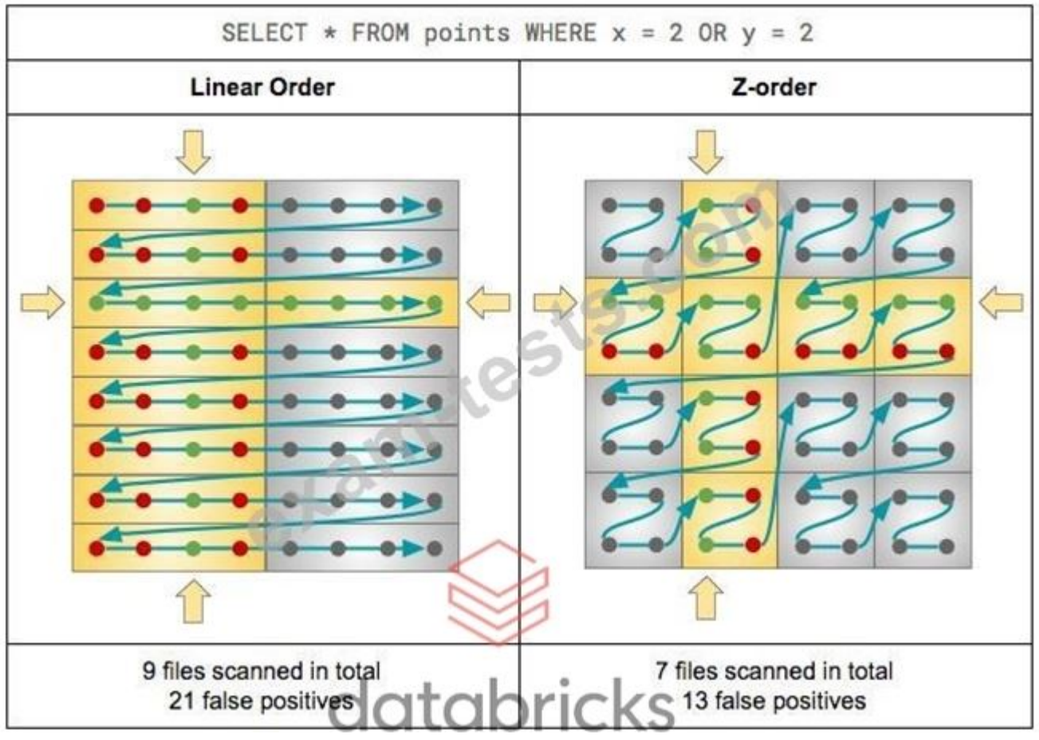
A. Create BLOOM FLTER index on the transactionId
B. Perform Optimize with Zorder on transactionId
(Correct)
C. transactionId has high cardinality, you cannot enable any optimization.
D. Increase the cluster size and enable delta optimization
E. Increase the driver size and enable delta optimization
Answer: B (LEAVE A REPLY)
Explanation
The answer is, perform Optimize with Z-order by transactionid
Here is a simple explanation of how Z-order works, once the data is naturally ordered, when a flle is scanned it only brings the data it needs into spark's memory Based on the column min and max it knows which data files needs to be scanned.
Table Description automatically generated

```
SELECT input_file_name() as "file_name",
       min(col) AS "col_min",
       max(col) AS "col_max"
FROM table
GROUP BY input_file_name()
```

file_name	col_min	col_max
data_file_1	6	8
data_file_2	3	10
data_file_3	1	4

databricks

Graphical user interface, diagram, application Description automatically generated



NEW QUESTION: 109
A junior data engineer has ingested a JSON file into a table raw_table with the following schema:
1. cart_id STRING,
2. items ARRAY<item_id:STRING>
The junior data engineer would like to unnest the items column in raw_table to result in a new table with the following schema:
1.cart_id STRING,
2.item_id STRING
Which of the following commands should the junior data engineer run to complete this task?
A. 1. SELECT cart_id, reduce(items) AS item_id
2. FROM raw_table;
B. 1. SELECT cart_id, slice(items) AS item_id
2. FROM raw_table;
C. 1. SELECT cart_id, filter(items) AS item_id
2. FROM raw_table;
D. 1. SELECT cart_id, flatten(items) AS item_id
2. FROM raw_table;
E. 1. SELECT cart_id, explode(items) AS item_id
2. FROM raw_table;
Answer: (SHOW ANSWER)

NEW QUESTION: 110
The data engineering team has configured a Databricks SQL query and alert to monitor the values in a Delta Lake table. The recent_sensor_recordings table contains an identifying sensor_id alongside the timestamp and temperature for the most recent 5 minutes of recordings.
The below query is used to create the alert:

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

The query is set to refresh each minute and always completes in less than 10 seconds. The alert is set to trigger when mean (temperature) > 120. Notifications are triggered to be sent at most every 1 minute.
If this alert raises notifications for 3 consecutive minutes and then stops, which statement must be true?
A. The total average temperature across all sensors exceeded 120 on three consecutive executions of the query
B. The recent_sensor_recordingstable was unresponsive for three consecutive runs of the query
C. The source query failed to update properly for three consecutive minutes and then restarted
D. The maximum temperature recording for at least one sensor exceeded 120 on three consecutive executions of the query
E. The average temperature recordings for at least one sensor exceeded 120 on three consecutive executions of the query
Answer: (SHOW ANSWER)
This is the correct answer because the query is using a GROUP BY clause on the sensor_id column, which means it will calculate the mean temperature for each sensor separately. The alert will trigger when the mean temperature for any sensor is greater than 120, which means at least one sensor had an average temperature above 120 for three consecutive minutes. The alert will stop when the mean temperature for all sensors drops below 120. Verified References: [Databricks Certified Data Engineer Professional], under "SQL Analytics" section; Databricks Documentation, under "Alerts" section.

NEW QUESTION: 111
A data engineer is overwriting data in a table by deleting the table and recreating the table. Another data engineer suggests that this is inefficient and the table should simply be overwritten instead.
Which of the following reasons to overwrite the table instead of deleting and recreating the table is incorrect?
A. Overwriting a table is an atomic operation and will not leave the table in an unfinished state
B. Overwriting a table allows for concurrent queries to be completed while in progress
C. Overwriting a table results in a clean table history for logging and audit purposes
D. Overwriting a table is efficient because no files need to be deleted
E. Overwriting a table maintains the old version of the table for Time Travel
Answer: (SHOW ANSWER)

NEW QUESTION: 112
An analytics team wants to run a short-term experiment in Databricks SQL on the customer transactions Delta table (about 20 billion records) created by the data engineering team. Which strategy should the data engineering team use to ensure minimal downtime and no impact on the ongoing ETL processes?
A. Create a new table for the analytics team using a CTAS statement.
B. Deep clone the table for the analytics team.
C. Give the analytics team direct access to the production table.
D. Shallow clone the table for the analytics team.
Answer: D (LEAVE A REPLY)
Comprehensive and Detailed Explanation From Exact Extract:

Exact extract: "A shallow clone creates a copy of the metadata that references the source data files; it is fast and inexpensive." Exact extract: "A deep clone copies the data." Exact extract: "Clones provide a point-in-time snapshot for experimentation without impacting the source." A shallow clone of the production Delta table creates an instantaneous snapshot that references the same data files, so it introduces virtually no downtime or storage overhead and avoids interfering with the ongoing ETL. A deep clone would copy all data (very expensive and slow for 20B rows). CTAS rewrites data and is unnecessary; direct access to prod risks contention and accidental changes.

NEW QUESTION: 113
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: (SHOW ANSWER)
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: 114
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:



databricks
exam-tests.com

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 (LEAVE A REPLY)

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: 115

Which of the following SQL statements can replace python variables in Databricks SQL code, when the notebook is set in SQL mode?

1.%python

2.table_name = "sales"

3.schema_name = "bronze"

4.

5.%sql

6.SELECT * FROM _____

A. SELECT * FROM f{schema_name.table_name}

B. SELECT * FROM {schem_name.table_name}

C. SELECT * FROM ${schema_name}.${table_name}

D. SELECT * FROM schema_name.table_name

Answer: C (LEAVE A REPLY)

Explanation

The answer is, SELECT * FROM ${schema_name}.${table_name}

%python
table_name = "sales"
schema_name = "bronze"
%sql
SELECT * FROM ${schema_name}.${table_name}
${python variable} -> Python variables in Databricks SQL code

NEW QUESTION: 116
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 thedatevariable:

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

exam-tests.com
databricks

Assume that the fieldscustomer_idandorder_idserve 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 (LEAVE A REPLY)
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: 117
A data engineer has ingested data from an external source into a PySpark DataFrame raw_df. They need to briefly make this data available in SQL for a data analyst to perform a quality assurance check on the data.
Which of the following commands should the data engineer run to make this data available in SQL for only the remainder of the Spark session?
A. raw_df.saveAsTable("raw_df")
B. raw_df.createOrReplaceTempView("raw_df")
C. raw_df.createTable("raw_df")
D. There is no way to share data between PySpark and SQL
E. raw_df.write.save("raw_df")
Answer: B (LEAVE A REPLY)

NEW QUESTION: 118
A data engineer has created a new cluster using shared access mode with default configurations. The data engineer needs to allow the development team access to view the driver logs if needed.
What are the minimal cluster permissions that allow the development team to accomplish this?
A. CAN ATTACH TO
B. CAN MANAGE
C. CAN VIEW
D. CAN RESTART
Answer: C (LEAVE A REPLY)
Comprehensive and Detailed In-Depth Explanation:
Databricks provides differentpermission levelsto control access to clusters. The correctminimal permission required for viewing driver logs isCAN VIEW.
Databricks Cluster Permission Levels:
* CAN ATTACH TO:
* Allows users toattach notebooks to a clusterbutdoes notallow them toview logs.
* Not sufficientfor viewing driver logs.
* CAN MANAGE:
* Grantsfull controlover the cluster, including starting, stopping, and editing configurations.
* Too broad for this requirement.
* CAN VIEW (Correct Answer):
* Allows users toview cluster details, logs, and statusbutnot modifyany configurations.
* Minimal required permission for viewing logs.
* CAN RESTART:
* Grants permission torestart the cluster, but doesnot include log access.
* Not sufficientfor viewing logs.
Conclusion:
Theminimalpermission needed to allow the development team toview driver logsisCAN VIEW.
References:
* Databricks Cluster Permissions Documentation

NEW QUESTION: 119
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 (LEAVE A REPLY)
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: 120
Which of the following programming languages can be used to build a Databricks SQL dashboard?
A. R
B. SQL
C. Scala
D. All of the above
E. Python
Answer: B (LEAVE A REPLY)

NEW QUESTION: 121
Which of the following is not a privilege in the Unity catalog?
A. SELECT
B. MODIFY
C. DELETE
D. CREATE TABLE
E. EXECUTE
Answer: C (LEAVE A REPLY)
Explanation
The Answer is DELETE and UPDATE permissions do not exit, you have to use MODIFY which provides both Update and Delete permissions.
Please note: TABLE ACL privilege types are different from Unity Catalog privilege types, please read the question carefully.
Here is the list of all privileges in Unity Catalog:

Privilege types

The following table shows which Unity Catalog privileges are associated with which Unity Catalog securable objects.

Securable	Privileges
Metastore	`CREATE CATALOG`, `CREATE EXTERNAL LOCATION`, `CREATE SHARE`, `CREATE RECIPIENT`
Catalog	`USE CATALOG`, `CREATE SCHEMA` Privileges for securable objects within a catalog can be granted at the catalog level.
Schema	`USE SCHEMA`, `CREATE TABLE`, `CREATE VIEW`, `CREATE FUNCTION` Privileges for securable objects within a schema can be granted at the schema level.
Table	`SELECT`, `MODIFY`
View	`SELECT`
External location	`CREATE EXTERNAL TABLE`, `READ FILES`, `WRITE FILES`
Storage credential	`CREATE EXTERNAL TABLE`, `READ FILES`, `WRITE FILES`, `CREATE EXTERNAL LOCATION`
Function	`EXECUTE`
Share	`SELECT` (can be granted to `RECIPIENT`)
Recipient	None.
Provider	None.

exam-tests.com



databricks

Unity Catalog Privileges
https://learn.microsoft.com/en-us/azure/databricks/spark/latest/spark-sql/language-manual/sql-ref-privileges#priv Table ACL privileges
https://learn.microsoft.com/en-us/azure/databricks/security/access-control/table-acls/object-privileges#privileges

Valid Databricks-Certified-Professional-Data-Engineer Dumps shared by BraindumpsPass.com for Helping Passing Databricks-Certified-Professional-Data-Engineer Exam! BraindumpsPass.com now offer the **newest Databricks-Certified-Professional-Data-Engineer exam dumps**, the BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer exam **questions have been updated** and **answers have been corrected** get the **newest** BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer dumps with Test Engine here:

https://www.braindumpspass.com/Databricks/Databricks-Certified-Professional-Data-Engineer-practice-exam-dumps.html (**217** Q&As Dumps, **40%OFF** **Special Discount:** **Exam-Tests**)

NEW QUESTION: 122
The data engineer team is configuring environment for development testing, and production before beginning migration on a new data pipeline. The team requires extensive testing on both the code and data resulting from code execution, and the team want to develop and test against similar production data as possible.
A junior data engineer suggests that production data can be mounted to the development testing environments, allowing pre production code to execute against production data. Because all users have Admin privileges in the development environment, the junior data engineer has offered to configure permissions and mount this data for the team.
Which statement captures best practices for this situation?
A. Because access to production data will always be verified using passthrough credentials it is safe to mount data to any Databricks development environment.
B. All developer, testing and production code and data should exist in a single unified workspace; creating separate environments for testing and development further reduces risks.

C. In environments where interactive code will be executed, production data should only be accessible with read permissions; creating isolated databases for each environment further reduces risks.
D. Because delta Lake versions all data and supports time travel, it is not possible for user error or malicious actors to permanently delete production data, as such it is generally safe to mount production data anywhere.
Answer: (SHOW ANSWER)
The best practice in such scenarios is to ensure that production data is handled securely and with proper access controls. By granting only read access to production data in development and testing environments, it mitigates the risk of unintended data modification. Additionally, maintaining isolated databases for different environments helps to avoid accidental impacts on production data and systems.
:
Databricks best practices for securing data: https://docs.databricks.com/security/index.html

NEW QUESTION: 123
A Spark job is taking longer than expected. Using the Spark UI, a data engineer notes that the Min, Median, and Max Durations for tasks in a particular stage show the minimum and median time to complete a task as roughly the same, but the max duration for a task to be roughly 100 times as long as the minimum.
Which situation is causing increased duration of the overall job?
A. Task queueing resulting from improper thread pool assignment.
B. Spill resulting from attached volume storage being too small.
C. Network latency due to some cluster nodes being in different regions from the source data
D. Skew caused by more data being assigned to a subset of spark-partitions.
E. Credential validation errors while pulling data from an external system.
Answer: D (LEAVE A REPLY)
This is the correct answer because skew is a common situation that causes increased duration of the overall job. Skew occurs when some partitions have more data than others, resulting in uneven distribution of work among tasks and executors. Skew can be caused by various factors, such as skewed data distribution, improper partitioning strategy, or join operations with skewed keys. Skew can lead to performance issues such as long-running tasks, wasted resources, or even task failures due to memory or disk spills. Verified References: [Databricks Certified Data Engineer Professional], under "Performance Tuning" section; Databricks Documentation, under "Skew" section.

NEW QUESTION: 124
You are working on a marketing team request to identify customers with the same information between two tables CUSTOMERS_2021 and CUSTOMERS_2020 each table contains 25 columns with the same schema, You are looking to identify rows that match between two tables across all columns, which of the following can be used to perform in SQL
A. 1.SELECT * FROM CUSTOMERS_2021
2. UNION
3.SELECT * FROM CUSTOMERS_2020
B. 1.SELECT * FROM CUSTOMERS_2021
2. UNION ALL
3.SELECT * FROM CUSTOMERS_2020
C. 1.SELECT * FROM CUSTOMERS_2021 C1
2.INNER JOIN CUSTOMERS_2020 C2
3.ON C1.CUSTOMER_ID = C2.CUSTOMER_ID
D. 1.SELECT * FROM CUSTOMERS_2021
2. INTERSECT
3.SELECT * FROM CUSTOMERS_2020
E. 1.SELECT * FROM CUSTOMERS_2021
2.EXCEPT

3.SELECT * FROM CUSTOMERS_2020
Answer: D (LEAVE A REPLY)
Explanation
Answer is,
1.SELECT * FROM CUSTOMERS_2021
2. INTERSECT
3.SELECT * FROM CUSTOMERS_2020
To compare all the rows between both the tables across all the columns using intersect will help us achieve that, an inner join is only going to check if the same column value exists across both the tables on a single column.
INTERSECT [ALL | DISTINCT]
*Returns the set of rows which are in both subqueries.
If ALL is specified a row that appears multiple times in the subquery1 as well as in subquery will be returned multiple times.
If DISTINCT is specified the result does not contain duplicate rows. This is the default.

NEW QUESTION: 125
A Delta Lake table was created with the below query:

```
CREATE TABLE prod.sales_by_store
AS (
  SELECT *
  FROM prod.sales a
  INNER JOIN prod.store b
  ON a.store_id = b.store_id
)
```

databricks exam-tests.com

Consider the following query:
DROP TABLE prod.sales_by_store -
If this statement is executed by a workspace admin, which result will occur?
A. Nothing will occur until a COMMIT command is executed.
B. The table will be removed from the catalog but the data will remain in storage.
C. The table will be removed from the catalog and the data will be deleted.
D. An error will occur because Delta Lake prevents the deletion of production data.
E. Data will be marked as deleted but still recoverable with Time Travel.
Answer: C (LEAVE A REPLY)
When a table is dropped in Delta Lake, the table is removed from the catalog and the data is deleted. This is because Delta Lake is a transactional storage layer that provides ACID guarantees. When a table is dropped, the transaction log is updated to reflect the deletion of the table and the data is deleted from the underlying storage. References :
* https://docs.databricks.com/delta/quick-start.html#drop-a-table
* https://docs.databricks.com/delta/delta-batch.html#drop-table

NEW QUESTION: 126
A Delta Lake table was created with the below query:

```
CREATE TABLE prod.sales_by_stor
USING DELTA
LOCATION "/mnt/prod/sales_by_store"
```

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 (LEAVE A REPLY)
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 "ALTERTABLE RENAME TO" section; Databricks Documentation, under "Metastore" section; Databricks Documentation, under "Managed and external tables" section.

NEW QUESTION: 127
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: A (LEAVE A REPLY)
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: 128
A data engineer is designing an append-only pipeline that needs to handle both batch and streaming data in Delta Lake. The team wants to ensure that the streaming component can efficiently track which data has already been processed.
Which configuration should be set to enable this?
A. overwriteSchema
B. partitionBy
C. checkpointLocation
D. mergeSchema
Answer: (SHOW ANSWER)
When working with Delta Lake streaming ingestion, checkpointing is critical for maintaining fault tolerance and ensuring exactly-once data processing semantics.
The checkpointLocation parameter defines the directory where Spark Structured Streaming stores progress information, offsets, and metadata. This allows the engine to resume processing from the last committed offset without reprocessing previously ingested data.
Without checkpointing, each stream restart would reprocess all data, leading to duplicates. Parameters like partitionBy or schema options (mergeSchema / overwriteSchema) affect table structure, not data lineage tracking. Therefore, the correct and required configuration for efficient streaming state management is checkpointLocation .

NEW QUESTION: 129
A data team's Structured Streaming job is configured to calculate running aggregates for item sales to update a downstream marketing dashboard. The marketing team has introduced a new field to track the number of times this promotion code is used for each item. A junior data engineer suggests updating the existing query as follows: Note that proposed changes are in bold.



Which step must also be completed to put the proposed query into production?
A. Increase the shuffle partitions to account for additional aggregates
B. Specify a new checkpointlocation
C. Run REFRESH TABLE delta, /item_agg'
D. Remove .option (mergeSchema', true') from the streaming write
Answer: B (LEAVE A REPLY)

When introducing a new aggregation or a change in the logic of a Structured Streaming query, it is generally necessary to specify a new checkpoint location. This is because the checkpoint directory contains metadata about the offsets and the state of the aggregations of a streaming query. If the logic of the query changes, such as including a new aggregation field, the state information saved in the current checkpoint would not be compatible with the new logic, potentially leading to incorrect results or failures. Therefore, to accommodate the new field and ensure the streaming job has the correct starting point and state information for aggregations, a new checkpoint location should be specified.
References:
* Databricks documentation on Structured Streaming: https://docs.databricks.com/spark/latest/structured- streaming/index.html
* Databricks documentation on streaming checkpoints: https://docs.databricks.com/spark/latest/structured- streaming/production.html#checkpointing

NEW QUESTION: 130
A data engineer wants to join a stream of advertisement impressions (when an ad was shown) with another stream of user clicks on advertisements to correlate when impression led to monitizable clicks.



exam-tests.com
databricks

Which solution would improve the performance?

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

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

NEW QUESTION: 131
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 (LEAVE A REPLY)
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: 132
A table is registered with the following code:

```
CREATE TABLE recent_orders AS (
  SELECT a.user_id, a.email, b.order_id, b.order_date
  FROM
    (SELECT user_id, email
    FROM users) a
    INNER JOIN
    (SELECT user_id, order_id, order_date
    FROM orders
    WHERE order_date >= (current_date() - 7)) b
    ON a.user_id = b.user_id
)
```

exam-tests.com
databricks

Bothusersandordersare Delta Lake tables. Which statement describes the results of queryingrecent_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 definedand store the result of joiningtables 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: D (LEAVE A REPLY)
Explanation
This is the correct answer because Delta Lake supports time travel, which allows users to query data as of a specific version or timestamp. The code uses the VERSION AS OF syntax to specify the version of each source table to be used in the join. The result of querying recent_orders will be the same as joining those versions of the source tables at query time. The query will use snapshot isolation, which means it will use a consistent snapshot of the table at the time the query began, regardless of any concurrent updates or deletes.
Verified References: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Query an older snapshot of a table (time travel)" section.

NEW QUESTION: 133
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 fieldpk_id.
For auditing purposes, the data governance team wishes to maintain a full record of all values that have ever been valid in the source system. For analytical purposes, only the most recent value for each record needs to be recorded. The Databricks job to ingest these records occurs once per hour, but each individual record may have changed multiple times over the course of an hour.
Which solution meets these requirements?

A. Create a separate history table for each pk_id resolve the current state of the table by running a union all filtering the history tables for the most recent state.
B. Use merge into to insert, update, or delete the most recent entry for each pk_id into a bronze table, then propagate all changes throughout the system.
C. Iterate through an ordered set of changes to the table, applying each in turn; rely on Delta Lake's versioning ability to create an audit log.
D. Use Delta Lake's change data feed to automatically process CDC data from an external system, propagating all changes to all dependent tables in the Lakehouse.
E. Ingest all log information into a bronze table; use merge into to insert, update, or delete the most recent entry for each pk_id into a silver table to recreate the current table state.
Answer: E (LEAVE A REPLY)
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 bronzetable, 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: 134
The data engineering team has configured a Databricks SQL query and alert to monitor the values in a Delta Lake table. Therecent_sensor_recordingstable contains an identifyingsensor_idalongside thetimestampandtemperaturefor the most recent 5 minutes of recordings.
The below query is used to create the alert:

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

The query is set to refresh each minute and always completes in less than 10 seconds. The alert is set to trigger whenmean (temperature) > 120. Notifications are triggered to be sent at most every 1 minute.
If this alert raises notifications for 3 consecutive minutes and then stops, which statement must be true?
A. The total average temperature across all sensors exceeded 120 on three consecutive executions of the query
B. Therecent_sensor_recordingstable was unresponsive for three consecutive runs of the query
C. The source query failed to update properly for three consecutive minutes and then restarted
D. The maximum temperature recording for at least one sensor exceeded 120 on three consecutive executions of the query
E. The average temperature recordings for at least one sensor exceeded 120 on three consecutive executions of the query
Answer: E (LEAVE A REPLY)
This is the correct answer because the query is using a GROUP BY clause on the sensor_id column, which means it will calculate the mean temperature for each sensor separately. The alert will trigger when the mean temperature for any sensor is greater than 120, which means at least one sensor had an average temperature above 120 for three consecutive minutes. The alert will stop when the mean temperature for all sensors drops below 120. Verified References: [Databricks Certified Data Engineer Professional], under "SQL Analytics" section; Databricks Documentation, under "Alerts" section.

NEW QUESTION: 135
Which of the following operations are not supported on a streaming dataset view?
spark.readStream.format("delta").table("sales").createOrReplaceTempView("streaming_view")
A. SELECT sum(unitssold) FROM streaming_view
B. SELECT max(unitssold) FROM streaming_view
C. SELECT id, sum(unitssold) FROM streaming_view GROUP BY id ORDER BY id
D. SELECT id, count(*) FROM streaming_view GROUP BY id
E. SELECT * FROM streadming_view ORDER BY id
Answer: (SHOW ANSWER)
Explanation

The answer isSELECT * FROM streadming_view order by id Please Note: Sorting with Group by will work without any issues see below explanation for each option of the options, Graphical user interface, text, application Description automatically generated



Certain operations are not allowed on streaming data, please see highlighted in bold.

https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#unsupported-operations

* Multiple streaming aggregations (i.e. a chain of aggregations on a streaming DF) are not yet supported on streaming Datasets.
* Limit and take the first N rows are not supported on streaming Datasets.
* Distinct operations on streaming Datasets are not supported.
* Deduplication operation is not supported after aggregation on a streaming Datasets.
* Sorting operations are supported on streaming Datasets only after an aggregation and in Complete Output Mode.

Note: Sorting without aggregation function is not supported.

Here is the sample code to prove this,

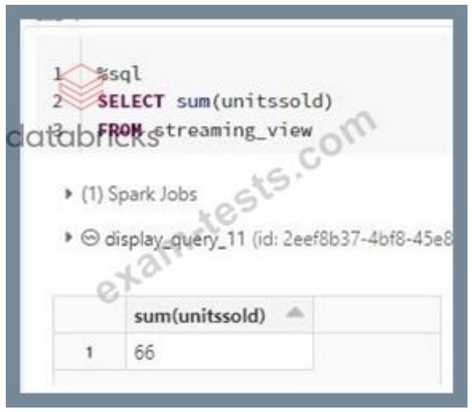
Setup test stream

Graphical user interface, text, application, email Description automatically generated



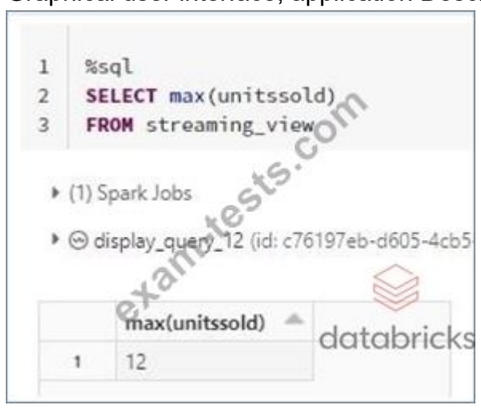
Sum aggregation function has no issues on stream

Graphical user interface, application Description automatically generated



Max aggregation function has no issues on stream

Graphical user interface, application Description automatically generated



Group by with Order by has no issues on stream



Group by has no issues on stream

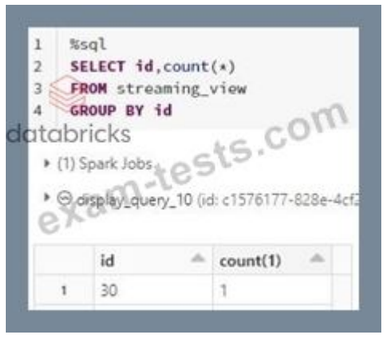


Table Description automatically generated

Order by without group by fails.

Graphical user interface, text, application Description automatically generated



NEW QUESTION: 136

A production cluster has 3 executor nodes and uses the same virtual machine type for the driver and executor.

When evaluating the Ganglia Metrics for this cluster, which indicator would signal a bottleneck caused by code executing on the driver?

A. The five Minute Load Average remains consistent/flat

B. Bytes Received never exceeds 80 million bytes per second

C. Total Disk Space remains constant

D. Network I/O never spikes

E. Overall cluster CPU utilization is around 25%

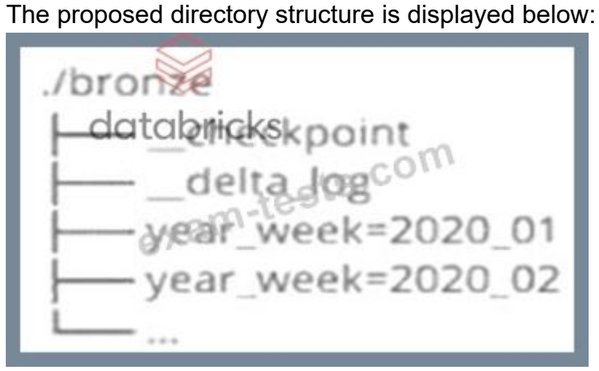
Answer: E (LEAVE A REPLY)

Explanation

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

Valid Databricks-Certified-Professional-Data-Engineer Dumps shared by BraindumpsPass.com for Helping Passing Databricks-Certified-Professional-Data-Engineer Exam! BraindumpsPass.com now offer the **newest Databricks-Certified-Professional-Data-Engineer exam dumps**, the BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer exam **questions have been updated** and **answers have been corrected** get the **newest** BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer dumps with Test Engine here:
https://www.braindumpspass.com/Databricks/Databricks-Certified-Professional-Data-Engineer-practice-exam-dumps.html (**217** Q&As Dumps, **40%OFF** **Special Discount:** **Exam-Tests**)

NEW QUESTION: 137
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 (LEAVE A REPLY)
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: 138
A Delta Lake table representing metadata about content from user has the following schema:

```
user_id LONG, post_text STRING, post_id STRING, longitude FLOAT, latitude FLOAT, post_time TIMESTAMP, date
DATE
```

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: (SHOW ANSWER)
Partitioning a Delta Lake table improves query performance by organizing data into partitions based on the values of a column. In the given schema, thedatecolumn is a good candidate for partitioning for several reasons:
* Time-Based Queries: If queries frequently filter or group by date, partitioning by thedatecolumn can significantly improve performance by limiting the amount of data scanned.
* Granularity: Thedatecolumn 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 likepost_idoruser_idmight lead to uneven partition sizes (data skew), which can negatively impact performance.
Partitioning bypost_timecould also be considered, but typicallydateis preferred due to its more manageable granularity.
References:
* Delta Lake Documentation on Table Partitioning: Optimizing Layout with Partitioning

NEW QUESTION: 139
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 abronzetable created with the propertydelta.enableChangeDataFeed = true. They plan to execute the following code as a daily job:

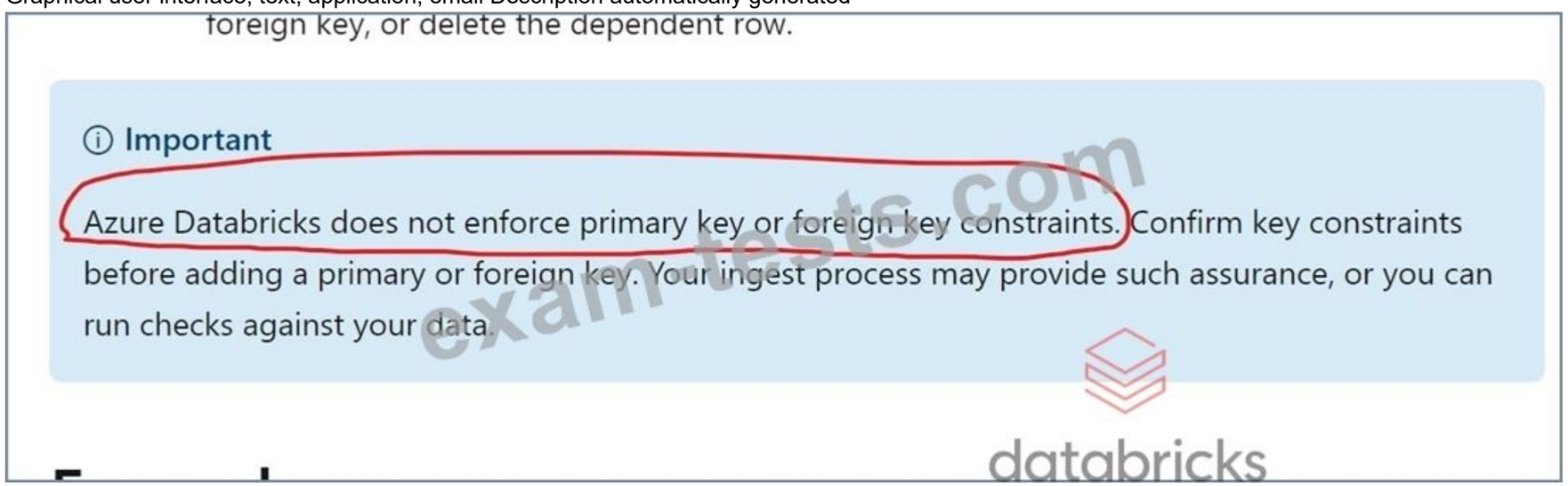
```
from pyspark.sql.functions import col

(spark.read.format("delta")
  .option("readChangeFeed", "true")
  .option("startingVersion", 0)
  .table("bronze")
  .filter(col("_change_type").isin(["update_postimage", "insert"]))
  .write
  .mode("append")
  .table("bronze_history_type1")
)
```

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: C (LEAVE A REPLY)
Explanation
This is the correct answer because it describes the execution and results of running the above query multiple times. The query uses the readChanges function to read all change events from a bronze table that has enabled change data feed. The readChanges function takes two arguments: version and options. The version argument specifies which version of the table to read changes from, and can be either a specific version number or -1 to indicate all versions. The options argument specifies additional options for reading changes, such as whether to include deletes or not. In this case, the query reads all changes from all versions of the bronze table and filters out delete events by setting includeDeletes to false. Then, it uses write.format("delta").mode("overwrite") to overwrite a target table using the entire history of inserted or updated records, giving the desired result of a Type 1 table representing all values that have ever been valid for all rows in the bronze table. Verified References: [Databricks Certified Data Engineer Professional], under

"Delta Lake" section; Databricks Documentation, under "Read changes in batch queries" section.

NEW QUESTION: 140
Which of the following table constraints that can be enforced on Delta lake tables are supported?
A. Primary key, foreign key, Not Null, Check Constraints
B. Primary key, Not Null, Check Constraints
C. Default, Not Null, Check Constraints
D. Not Null, Check Constraints
E. Unique, Not Null, Check Constraints
Answer: (SHOW ANSWER)
Explanation
The answer is Not Null, Check Constraints
https://docs.microsoft.com/en-us/azure/databricks/delta/delta-constraints
* CREATE TABLE events(id LONG,
* date STRING,
* location STRING,
* description STRING
*) USING DELTA;
ALTER TABLE events CHANGE COLUMN id SET NOT NULL;
ALTER TABLE events ADD CONSTRAINT dateWithinRange CHECK (date > '1900-01-01'); Note: Databricks as of DBR 11.1 added support for Primary Key and Foreign Key when Unity Catalog is enabled but this is for information purposes only these are not actually enforced. You may ask then why are we defining these if they are not enforced, so especially these information constraints are very helpful if you have a BI tool that can benefit from knowing the relationship between the tables, so it will be easy when creating reports/dashboards or understanding the data model when using any Data modeling tool.
Primary and Foreign Key
Graphical user interface, text, application, email Description automatically generated



NEW QUESTION: 141

A data engineering team is configuring access controls in Databricks Unity Catalog. They grant the SELECT privilege on the sales catalog to the analyst_group, expecting that members of this group will automatically have SELECT access to all current and future schemas, tables, and views within the catalog.
What describes the privilege inheritance behavior in Unity Catalog?
A. Granting SELECT on a catalog automatically applies SELECT to all current and future schemas, tables, and views within that catalog.
B. Privileges in Unity Catalog do not cascade; SELECT must be explicitly granted on each schema and table, even if granted at the catalog level.
C. Granting SELECT at the catalog level applies to existing schemas and tables but not to those created in the future.
D. Privileges granted at the schema level override any catalog-level privileges and prevent access unless explicitly revoked.
Answer: B (LEAVE A REPLY)
Comprehensive and Detailed Explanation From Exact Extract of Databricks Data Engineer Documents:
In Unity Catalog, privileges are non-cascading-meaning that granting a privilege (like SELECT) on a catalog does not automatically grant the same privilege on contained objects (schemas, tables, or views). Each object type has its own independent access control hierarchy.
According to the Databricks access control documentation: "Privileges do not automatically cascade from catalog to schema or table levels." Administrators must explicitly grant privileges on each level if users need access across objects. This design ensures tighter governance and least-privilege enforcement. Therefore, option B correctly describes Unity Catalog's privilege model, while A and D incorrectly imply automatic inheritance.

NEW QUESTION: 142
Review the following error traceback:

```
---------------------------------------------------------------------------
AnalysisException                         Traceback (most recent call last)
<command-3293767849433948> in <module>
----> 1 display(df.select(3*"heartrate"))

/databricks/spark/python/pyspark/sql/dataframe.py in select(self, *cols)
   1690         [Row(name='Alice', age=12), Row(name='Bob', age=15)]
   1691         """
-> 1692         jdf = self._jdf.select(self._jcols(*cols))
   1693         return DataFrame(jdf, self.sql_ctx)
   1694

/databricks/spark/python/lib/py4j-0.10.9-src.zip/py4j/java_gateway.py in __call__(self, *args)
   1302
   1303         answer = self.gateway_client.send_command(command)
-> 1304         return_value = get_return_value(
   1305             answer, self.gateway_client, self.target_id, self.name)
   1306

/databricks/spark/python/pyspark/sql/utils.py in deco(*a, **kw)
    121                 # Hide where the exception came from that shows a non-Pythonic
    122                 # JVM exception message.
--> 123                 raise converted from None
    124             else:
    125                 raise

AnalysisException: cannot resolve '`heartrateheartrateheartrate`' given input columns:
[spark_catalog.database.table.device_id, spark_catalog.database.table.heartrate,
spark_catalog.database.table.mrn, spark_catalog.database.table.time];
'Project ['heartrateheartrateheartrate]
+- SubqueryAlias spark_catalog.database.table
   +- Relation[device_id#75L,heartrate#76,mrn#77L,time#78] parquet
```

exam-tests.com



databricks

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 (LEAVE A REPLY)
Explanation
The error is a Py4JJavaError, which means that an exception was thrown in Java code called by Python code using Py4J. Py4J is a library that enables Python programs to dynamically access Java objects in a Java Virtual Machine (JVM). PySpark uses Py4J to communicate with Spark's JVM-based engine. The error message shows that the exception was thrown by org.apache.spark.sql.AnalysisException, which means that an error occurred during the analysis phase of Spark SQL query processing. The error message also shows that the cause of the exception was "cannot resolve 'heartrateheartrateheartrate' given input columns". This means that Spark could not find a column named heartrateheartrateheartrate in the input DataFrame or Dataset. The reason for this error is that there is a syntax error in the code that caused this exception. The code is:
df.withColumn("heartrate", heartrate * 3)
The code tries to create a new column called heartrate by multiplying an existing column called heartrate by 3.
However, the code does not correctly identify the heartrate column as a column object, but rather as a plain Python variable. This causes PySpark to concatenate the variable name with itself three times, resulting in heartrateheartrateheartrate, which is not a valid column name. To fix this error, the code should use one of the following ways to identify the heartrate column as a column object:
df.withColumn("heartrate", df["heartrate"] * 3) df.withColumn("heartrate", df.heartrate * 3) df.withColumn("heartrate", col("heartrate") * 3) Verified References: [Databricks Certified Data Engineer Professional], under "Spark Core" section; Py4J Documentation, under "What is Py4J?"; Databricks Documentation, under "Query plans - Analysis phase"; Databricks Documentation, under "Accessing columns".

NEW QUESTION: 143
Which of the following two options are supported in identifying the arrival of new files, and incre-mental data from Cloud object storage using Auto Loader?
A. Directory listing, File notification
B. Checking pointing, watermarking
C. Writing ahead logging, read head logging
D. File hashing, Dynamic file lookup
E. Checkpointing and Write ahead logging
Answer: (SHOW ANSWER)
Explanation
The answer is A, Directory listing, File notifications
Directory listing: Auto Loader identifies new files by listing the input directory.
File notification: Auto Loader can automatically set up a notification service and queue service that subscribe to file events from the input directory.
Choosing between file notification and directory listing modes | Databricks on AWS

NEW QUESTION: 144
A Delta table of weather records is partitioned by date and has the below schema:
date DATE, device_id INT, temp FLOAT, latitude FLOAT, longitude FLOAT
To find all the records from within the Arctic Circle, you execute a query with the below filter:
latitude > 66.3
Which statement describes how the Delta engine identifies which files to load?
A. All records are cached to an operational database and then the filter is applied
B. The Parquet file footers are scanned for min and max statistics for the latitude column
C. All records are cached to attached storage and then the filter is applied
D. The Delta log is scanned for min and max statistics for the latitude column
E. The Hive metastore is scanned for min and max statistics for the latitude column
Answer: D (LEAVE A REPLY)
Explanation

This is the correct answer because Delta Lake uses a transaction log to store metadata about each table, including min and max statistics for each column in each data file. The Delta engine can use this information to quickly identify which files to load based on a filter condition, without scanning the entire table or the file footers. This is called data skipping and it can improve query performance significantly.
Verified References:
[Databricks Certified Data Engineer Professional], under "Delta Lake" section; [Databricks Documentation], under "Optimizations - Data Skipping" section.

NEW QUESTION: 145
Which of the following commands can be used to query a delta table?
A. 1.%python
2.spark.sql("select * from table_name")
B. 1.%sql
2.Select * from table_name
C. Both A & B
(Correct)
D. 1.%python
2.execute.sql("select * from table")
E. 1.%python
2.delta.sql("select * from table")
Answer: C (LEAVE A REPLY)
Explanation
The answer is both options A and B
Options C and D are incorrect because there is no command in Spark called execute.sql or delta.sql

NEW QUESTION: 146
In order to facilitate near real-time workloads, a data engineer is creating a helper function to leverage the schema detection and evolution functionality of Databricks Auto Loader. The desired function will automatically detect the schema of the source directly, incrementally process JSON files as they arrive in a source directory, and automatically evolve the schema of the table when new fields are detected.
The function is displayed below with a blank:

```
def auto_load_json(source_path: str,
                   checkpoint_path: str,
                   target_table_path: str):
    (spark.readStream
        .format("cloudFiles")
        .option("cloudFiles.format", "json")
        .option("cloudFiles.schemaLocation", checkpoint_path)
        .load(source_path)
        ______
    )
```

databricks
exam-tests.com

Which response correctly fills in the blank to meet the specified requirements?

```
  .writeStream
A.).option("mergeSchema", True)
  .start(target_table_path)
  .writeStream
  .option("checkpointLocation", checkpoint_path)
B. .option("mergeSchema", True)
  .trigger(once=True)
  .start(target_table_path)
  .write
  .option("checkpointLocation", checkpoint_path)
C. .option("mergeSchema", True)
  .outputMode("append")
  .save(target_table_path)
  .write
  .option("mergeSchema", True)
D.
  .mode("append")
  .save(target_table_path)
  .writeStream
  .option("checkpointLocation", checkpoint_path)
E.
  .option("mergeSchema", True)
  .start(target_table_path)
```

databricks exam-tests.com

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

Answer: B (LEAVE A REPLY)

Option B correctly fills in the blank to meet the specified requirements. Option B uses the "cloudFiles.
schemaLocation" option, which is required for the schema detection and evolution functionality of Databricks Auto Loader. Additionally, option B uses the "mergeSchema" option, which is required for the schema evolution functionality of Databricks Auto Loader. Finally, option B uses the "writeStream" method, which is required for the incremental processing of JSON files as they arrive in a source directory.
The other options are incorrect because they either omit the required options, use the wrong method, or use the wrong format. References:
* Configure schema inference and evolution in Auto Loader : https://docs.databricks.com/en/ingestion
/auto-loader/schema.html

* Write streaming data : https://docs.databricks.com/spark/latest/structured-streaming/writing-streaming- data.html

NEW QUESTION: 147
Which of the following describes a scenario in which a data engineer will want to use a Job cluster instead of
an all-purpose cluster?
A. A data engineer needs to manually investigate a production error
B. A Databricks SQL query needs to be scheduled for upward reporting
C. An ad-hoc analytics report needs to be developed while minimizing compute costs
D. An automated workflow needs to be run every 30 minutes
E. A data team needs to collaborate on the development of a machine learning model
Answer: (SHOW ANSWER)

NEW QUESTION: 148
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 (LEAVE A REPLY)
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: 149
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 (LEAVE A REPLY)
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: 150
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: B (LEAVE A REPLY)
Explanation
This is the best configuration for scheduling Structured Streaming jobs for production, as it automatically recovers from query failures and keeps costs low. A new job cluster is created for each run of the job and terminated when the job completes, which saves costs and avoids resource contention. Retries are not needed for Structured Streaming jobs, as they can automatically recover from failures using checkpointing and write-ahead logs. Maximum concurrent runs should be set to 1 to avoid duplicate output or data loss. Verified References: Databricks Certified Data Engineer Professional, under "Monitoring & Logging" section; Databricks Documentation, under "Schedule streaming jobs" section.

NEW QUESTION: 151
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 namedusers.

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

Assuming thatuser_idis a unique identifying key and thatdelete_requestscontains 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 (LEAVE A REPLY)
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.

Valid Databricks-Certified-Professional-Data-Engineer Dumps shared by BraindumpsPass.com for Helping Passing Databricks-Certified-Professional-Data-Engineer Exam! BraindumpsPass.com now offer the **newest Databricks-Certified-Professional-Data-Engineer exam dumps**, the BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer exam **questions have been updated** and **answers have been corrected** get the **newest** BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer dumps with Test Engine here:
https://www.braindumpspass.com/Databricks/Databricks-Certified-Professional-Data-Engineer-practice-exam-dumps.html (**217** Q&As Dumps, **40%OFF** **Special Discount:** **Exam-Tests**)

NEW QUESTION: 152
The view updates represents an incremental batch of all newly ingested data to be inserted or updated in the customers table.
The following logic is used to process these records.
MERGE INTO customers
USING (
SELECT updates.customer_id as merge_ey, updates .*
FROM updates
UNION ALL
SELECT NULL as merge_key, updates .*
FROM updates JOIN customers
ON updates.customer_id = customers.customer_id
WHERE customers.current = true AND updates.address <> customers.address) staged_updates ON customers.customer_id = mergekey WHEN MATCHED AND customers. current = true AND customers.address <> staged_updates.
address THEN
UPDATE SET current = false, end_date = staged_updates.effective_date
WHEN NOT MATCHED THEN
INSERT (customer_id, address, current, effective_date, end_date)
VALUES (staged_updates.customer_id, staged_updates.address, true, staged_updates.effective_date, null) Which statement describes this implementation?
* The customers table is implemented as a Type 2 table; old values are overwritten and new customers are appended.
A. The customers table is implemented as a Type 1 table; old values are overwritten by new values and no history is maintained.

B. The customers table is implemented as a Type 2 table; old values are maintained but marked as no longer current and new values are inserted.
C. The customers table is implemented as a Type 0 table; all writes are append only with no changes to existing values.
Answer: (SHOW ANSWER)
The providedMERGEstatement is a classic implementation of a Type 2 SCD in a data warehousing context.
In this approach, historical data is preserved by keeping old records (marking them as not current) and adding new records for changes. Specifically, when a match is found and there's a change in the address, the existing record in thecustomerstable is updated to mark it as no longer current (current = false), and an end date is assigned (end_date = staged_updates.effective_date). A new record for the customer is then inserted with the updated information, marked as current. This method ensures that the full history of changes to customer information is maintained in the table, allowing for time-based analysis of customer data.References:
Databricks documentation on implementing SCDs using Delta Lake and theMERGEstatement (https://docs.
databricks.com/delta/delta-update.html#upsert-into-a-table-using-merge).

NEW QUESTION: 153
Which of the following is true of Delta Lake and the Lakehouse?
A. Because Parquet compresses data row by row. strings will only be compressed when a character is repeated multiple times.
B. Delta Lake automatically collects statistics on the first 32 columns of each table which are leveraged in data skipping based on query filters.
C. Views in the Lakehouse maintain a valid cache of the most recent versions of source tables at all times.
D. Primary and foreign key constraints can be leveraged to ensure duplicate values are never entered into a dimension table.
E. Z-order can only be applied to numeric values stored in Delta Lake tables
Answer: B (LEAVE A REPLY)
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: 154
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:

```
df._____
    .groupBy(
        window("event_time", "5 minutes").alias("time"),
        "device_id"
    )
    .agg(
        avg("temp").alias("avg_temp"),
        avg("humidity").alias("avg_humidity")
    )
    .writeStream
    .format("delta")
    .saveAsTable("sensor_avg")
```

exam-tests.com
databricks

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 (LEAVE A REPLY)

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 otheroptions 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: 155

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")
  )
```

exam-tests.com
databricks

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

B. preds.write.format("delta").save("/preds/churn_preds")

C. preds.write.mode("append").saveAsTable("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")
)
```

Answer: C (LEAVE A REPLY)

NEW QUESTION: 156

The view updates represents an incremental batch of all newly ingested data to be inserted or updated in the customers table.
The following logic is used to process these records.

```
MERGE INTO customers
USING (
  SELECT updates.customer_id as merge_ey, updates.*
  FROM updates

  UNION ALL

  SELECT NULL as merge_key, updates.*
  FROM updates JOIN customers
  ON updates.customer_id = customers.customer_id
  WHERE customers.current = true AND updates.address <> customers.address
) staged_updates
ON customers.customer_id = mergeKey
WHEN MATCHED AND customers.current = true AND customers.address <> staged_updates.address THEN
  UPDATE SET current = false, end_date = staged_updates.effective_date
WHEN NOT MATCHED THEN
  INSERT(customer_id, address, current, effective_date, end_date)
  VALUES(staged_updates.customer_id, staged_updates.address, true, staged_updates.effective_date,
null)
```

exam-tests.com
databricks

Which statement describes this implementation?

A. The customers table is implemented as a Type 3 table; old values are maintained as a new column alongside the current value.

B. The customers table is implemented as a Type 2 table; old values are maintained but marked as no longer current and new values are inserted.

C. The customers table is implemented as a Type 0 table; all writes are append only with no changes to existing values.

D. The customers table is implemented as a Type 1 table; old values are overwritten by new values and no history is maintained.

E. The customers table is implemented as a Type 2 table; old values are overwritten and new customers are appended.

Answer: B (LEAVE A REPLY)

The logic uses the MERGE INTO command to merge new records from the view updates into the table customers. The MERGE INTO command takes two arguments: a target table and a source table or view. The command also specifies a condition to match records between the target and the source, and a set of actions to perform when there is a match or not. In this case, the condition is to match records by customer_id, which is the primary key of the customers table. The actions are to update the existing record in the target with the new values from the source, and set the current_flag to false to indicate that the record is no longer current; and to insert a new record in the target with the new values from the source, and set the current_flag to true to indicate that the record is current. This means that old values are maintained but marked as no longer current and new values are inserted, which is the definition of a Type 2 table. Verified Reference: [Databricks Certified Data Engineer Professional], under "Delta Lake" section; Databricks Documentation, under "Merge Into (Delta Lake on Databricks)" section.

NEW QUESTION: 157

What is the probability that the total of two dice will be greater than 8, given that the first die is a 6?

A. 2/6

B. 1/6

C. 2/3

D. 1/3
Answer: (SHOW ANSWER)

NEW QUESTION: 158
Identify one of the below statements that can query a delta table in PySpark Dataframe API
A. Spark.read.table.delta("table_name")
B. Spark.read.format("delta").TableAs("table_name")
C. Spark.read.table("table_name")
D. Spark.read.mode("delta").table("table_name")
E. Spark.read.format("delta").LoadTableAs("table_name")
Answer: (SHOW ANSWER)

NEW QUESTION: 159
The business reporting team requires that data for their dashboards be updated every hour. The total processing time for the pipeline that extracts, transforms, and loads the data for their pipeline runs in 10 minutes. Assuming normal operating conditions, which configuration will meet their service-level agreement requirements with the lowest cost?
A. Schedule a job to execute the pipeline once an hour on a dedicated interactive cluster.
B. Schedule a job to execute the pipeline once an hour on a new job cluster.
C. Schedule a Structured Streaming job with a trigger interval of 60 minutes.
D. Configure a job that executes every time new data lands in a given directory.
Answer: B (LEAVE A REPLY)
Comprehensive and Detailed Explanation From Exact Extract:
Exact extract: "Job clusters are created for a job run and terminate when the job completes." Exact extract: "All-purpose (interactive) clusters are intended for interactive development and collaboration." An hourly batch that completes in ~10 minutes is best run on ephemeral job clusters so compute exists only during execution and auto-terminates afterward, minimizing cost while meeting the hourly SLA. Interactive clusters persist (incurring idle cost). Structured Streaming for an hourly refresh adds continuous compute cost with no benefit here. Event-driven triggers are unnecessary for a simple hourly SLA.
Reference:

NEW QUESTION: 160
You were asked to write python code to stop all running streams, which of the following command can be used to get a list of all active streams currently running so we can stop them, fill in the blank.
1.for s in _____:
2. s.stop()
A. Spark.getActiveStreams()
B. activeStreams()
C. getActiveStreams()
D. spark.streams.getActive
E. spark.streams.active
Answer: E (LEAVE A REPLY)

NEW QUESTION: 161
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).



databricks
exam-tests.com

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 (LEAVE A REPLY)

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: 162

The marketing team is launching a new campaign to monitor the performance of the new campaign for the first two weeks, they would like to set up a dashboard with a refresh schedule to run every 5 minutes, which of the below steps can be taken to reduce of the cost of this refresh over time?

A. Reduce the size of the SQL Cluster size

B. Reduce the max size of auto scaling from 10 to 5

C. Setup the dashboard refresh schedule to end in two weeks

D. Change the spot instance policy from reliability optimized to cost optimized

E. Always use X-small cluster

Answer: C (LEAVE A REPLY)

Explanation

The answer is Setup the dashboard refresh schedule to end in two weeks

NEW QUESTION: 163

A healthcare analytics team is implementing a dimensional model in Delta Lake for patient care analysis.

They have a date dimension table and are evaluating design options to ensure it supports a wide range of time- based analyses.

Which design approach for the date dimension will support efficient time-based querying and aggregation?

A. Store the date as a string in the format YYYY-MM-DD for readability.
B. Create separate dimension tables for different calendar systems (fiscal, academic, etc.).
C. Store only the date value and calculate all time attributes dynamically in queries.
D. Pre-calculate attributes like fiscal_period, quarter, month_name, day_of_week, and holiday.
Answer: (SHOW ANSWER)
In dimensional modeling, Databricks recommends denormalized, attribute-rich dimension tables for performance and usability. A date dimension should include all commonly used derived time attributes such as fiscal period, quarter, month, weekday, and holiday flags. Precomputing these attributes ensures consistent business logic, eliminates repeated calculations during query time, and enables efficient filtering and aggregation. The documentation for Delta Lake and Lakehouse design explicitly advises precomputing these attributes for analytical workloads that depend heavily on time-based slicing. Options A and C degrade performance and consistency, while maintaining multiple calendar-specific dimension tables (B) complicates the model unnecessarily.

NEW QUESTION: 164
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 compaction C 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 (LEAVE A REPLY)
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 Reference: [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: 165
Which Python variable contains a list of directories to be searched when trying to locate required modules?
A. os-path
B. importlib.resource path
C. pylib.source
D. pypi.path
E. ,sys.path
Answer: (SHOW ANSWER)

NEW QUESTION: 166
A data analyst has provided a data engineering team with the following Spark SQL query:
1.SELECT district,
2.avg(sales)
3.FROM store_sales_20220101
4.GROUP BY district;

The data analyst would like the data engineering team to run this query every day. The date at the end of the table name (20220101) should automatically be replaced with the current date each time the query is run.
Which of the following approaches could be used by the data engineering team to efficiently auto-mate this process?
A. They could manually replace the date within the table name with the current day's date
B. They could wrap the query using PySpark and use Python's string variable system to automatically update the table name
C. They could replace the string-formatted date in the table with a timestamp-formatted date
D. They could request that the data analyst rewrites the query to be run less frequently
E. They could pass the table into PySpark and develop a robustly tested module on the existing query
Answer: B (LEAVE A REPLY)

Valid Databricks-Certified-Professional-Data-Engineer Dumps shared by BraindumpsPass.com for Helping Passing Databricks-Certified-Professional-Data-Engineer Exam! BraindumpsPass.com now offer the **newest Databricks-Certified-Professional-Data-Engineer exam dumps**, the BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer exam **questions have been updated** and **answers have been corrected** get the **newest** BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer dumps with Test Engine here:
https://www.braindumpspass.com/Databricks/Databricks-Certified-Professional-Data-Engineer-practice-exam-dumps.html (**217** Q&As Dumps, **40%OFF** **Special Discount:** **Exam-Tests**)

NEW QUESTION: 167
The view updates represents an incremental batch of all newly ingested data to be inserted or updated in the customers table.
The following logic is used to process these records.
MERGE INTO customers
USING (
SELECT updates.customer_id as merge_ey, updates .*
FROM updates
UNION ALL
SELECT NULL as merge_key, updates .*
FROM updates JOIN customers
ON updates.customer_id = customers.customer_id
WHERE customers.current = true AND updates.address < > customers.address) staged_updates ON customers.customer_id = mergekey WHEN MATCHED AND customers. current = true AND customers.address < > staged_updates.
address THEN
UPDATE SET current = false, end_date = staged_updates.effective_date
WHEN NOT MATCHED THEN
INSERT (customer_id, address, current, effective_date, end_date)
VALUES (staged_updates.customer_id, staged_updates.address, true, staged_updates.effective_date, null) Which statement describes this implementation?
A. The customers table is implemented as a Type 2 table; old values are overwritten and new customers are appended.
B. The customers table is implemented as a Type 1 table; old values are overwritten by new values and no history is maintained.
C. The customers table is implemented as a Type 2 table; old values are maintained but marked as no longer current and new values are inserted.
D. The customers table is implemented as a Type 0 table; all writes are append only with no changes to existing values.

Answer: (SHOW ANSWER)

The provided MERGE statement is a classic implementation of a Type 2 SCD in a data warehousing context.

In this approach, historical data is preserved by keeping old records (marking them as not current) and adding new records for changes. Specifically, when a match is found and there ' s a change in the address, the existing record in the customers table is updated to mark it as no longer current (current = false), and an end date is assigned (end_date = staged_updates.effective_date). A new record for the customer is then inserted with the updated information, marked as current. This method ensures that the full history of changes to customer information is maintained in the table, allowing for time-based analysis of customer data.

Databricks documentation on implementing SCDs using Delta Lake and the MERGE statement (https://docs.databricks.com/delta/delta-update.html#upsert-into-a-table-using-merge).

Valid Databricks-Certified-Professional-Data-Engineer Dumps shared by BraindumpsPass.com for Helping Passing Databricks-Certified-Professional-Data-Engineer Exam! BraindumpsPass.com now offer the **newest Databricks-Certified-Professional-Data-Engineer exam dumps**, the BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer exam **questions have been updated** and **answers have been corrected** get the **newest** BraindumpsPass.com Databricks-Certified-Professional-Data-Engineer dumps with Test Engine here:

https://www.braindumpspass.com/Databricks/Databricks-Certified-Professional-Data-Engineer-practice-exam-dumps.html (**217** Q&As Dumps, **40%OFF** **Special Discount:** **Exam-Tests**)