## Question 1

Which of the statements are **incorrect** when choosing between lakehouse and Datawarehouse?

* **Lakehouse can have special indexes and caching which are optimized for Machine learning**
* **Lakehouse cannot serve low query latency with high reliability for BI workloads, only suitable for batch workloads.**
* **Lakehouse can be accessed through various API’s including but not limited to Python/R/SQL**
* **Traditional Data warehouses have storage and compute are coupled.**
* **Lakehouse uses standard data formats like Parquet.**

**Explanation**

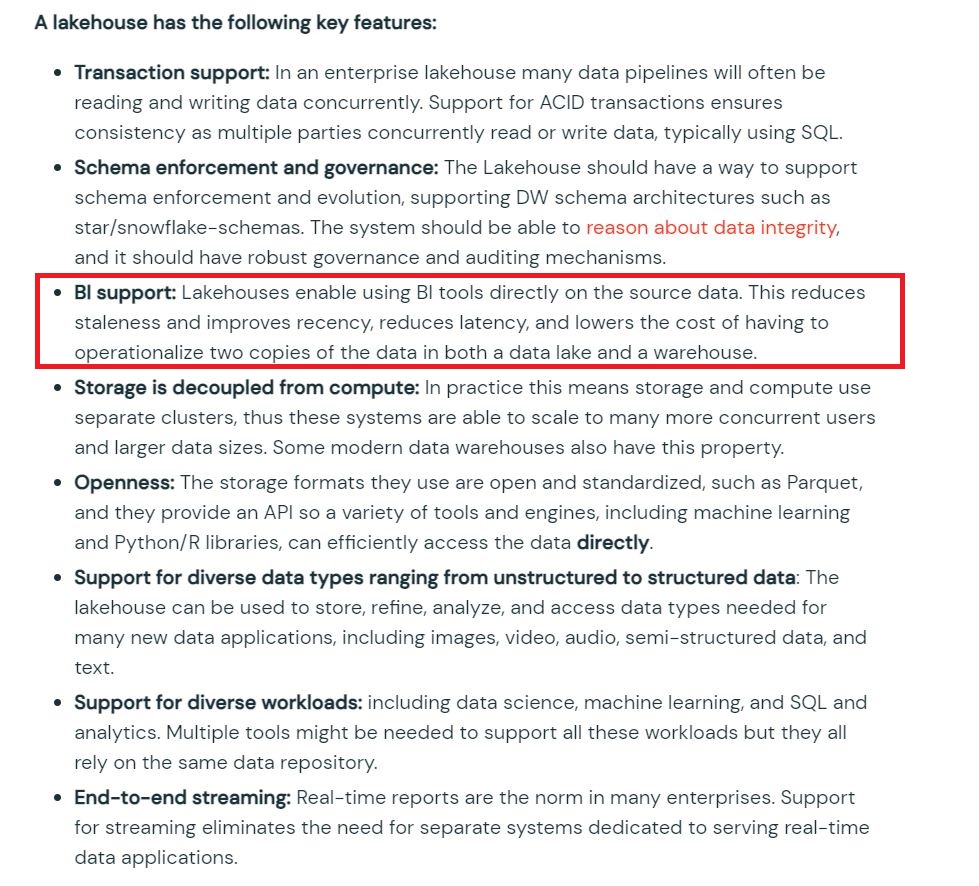
The answer is Lakehouse cannot serve low query latency with high reliability for BI workloads, only suitable for batch workloads.

Lakehouse can replace traditional warehouses by leveraging storage and compute optimizations like caching to serve them with low query latency with high reliability.

Focus on comparisons between Spark Cache vs Delta Cache.

<https://docs.databricks.com/delta/optimizations/delta-cache.html>

[What Is a Lakehouse? - The Databricks Blog](https://databricks.com/blog/2020/01/30/what-is-a-data-lakehouse.html)



## Question 2

Which of the statements are correct about lakehouse?

* **Lakehouse only supports Machine learning workloads and Data warehouses support BI workloads**
* **Lakehouse only supports end-to-end streaming workloads and Data warehouses support Batch workloads**
* **Lakehouse does not support ACID**
* **In Lakehouse Storage and compute are coupled**
* **Lakehouse supports schema enforcement and evolution**

**Explanation**

The answer is  Lakehouse supports schema enforcement and evolution,

Lakehouse using Delta lake can not only enforce a schema on write which is contrary to traditional big data systems that can only enforce a schema on read, it also supports evolving schema over time with the ability to control the evolution.

For example below is the Dataframe writer API and it supports three modes of enforcement and evolution,

***Default:***Only enforcement, no changes are allowed and any schema drift/evolution will result in failure.

***Merge:*** Flexible, supports enforcement and evolution

* New columns are added
* Evolves nested columns
* Supports evolving data types, like Byte to Short to Integer to Bigint

How to enable:

1. DF.write.format("delta").option("mergeSchema", "true").saveAsTable("table\_name")
2. or
3. spark.databricks.delta.schema.autoMerge = True  ## Spark session

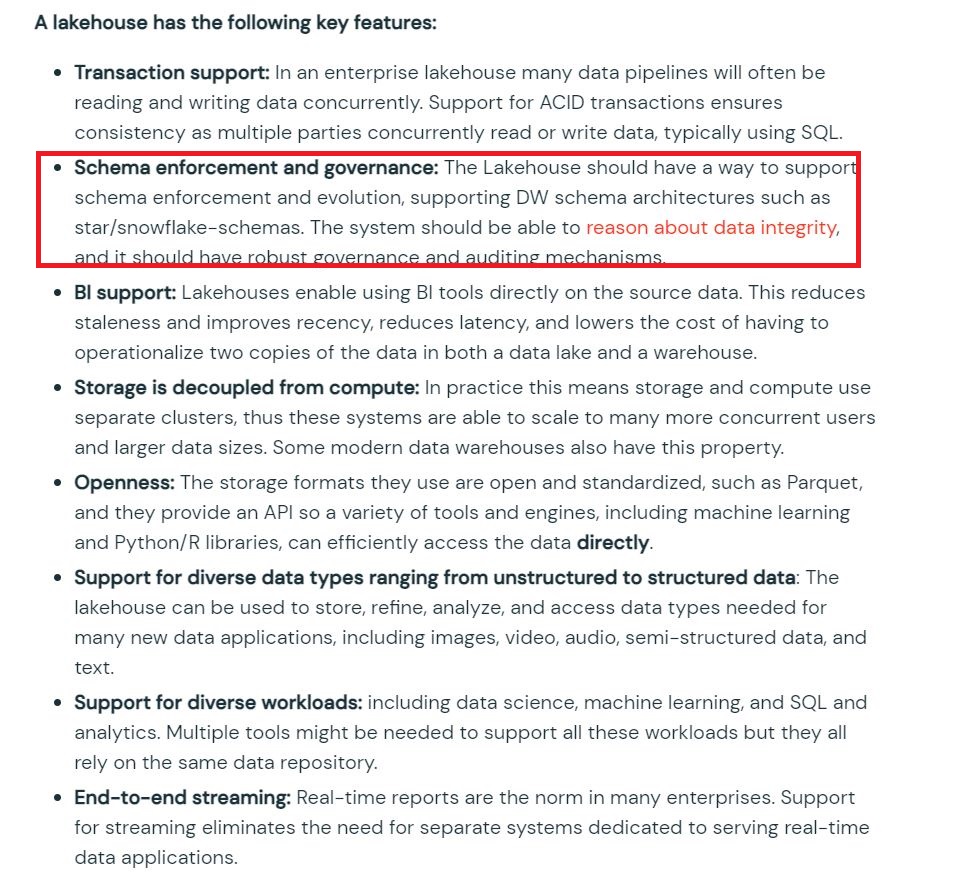
***Overwrite:***No enforcement

* Dropping columns
* Change string to integer
* Rename columns

How to enable:

1. DF.write.format("delta").option("overwriteSchema", "True").saveAsTable("table\_name")

[What Is a Lakehouse? - The Databricks Blog](https://databricks.com/blog/2020/01/30/what-is-a-data-lakehouse.html)



## Question 3

Which of the following are stored in the control pane of Databricks Architecture?

* **Job Clusters**
* **All Purpose Clusters**
* **Databricks Filesystem**
* **Databricks Web Application**
* **Delta tables**

**Explanation**

The answer is Databricks Web Application

[Azure Databricks architecture overview - Azure Databricks | Microsoft Docs](https://docs.microsoft.com/en-us/azure/databricks/getting-started/overview)

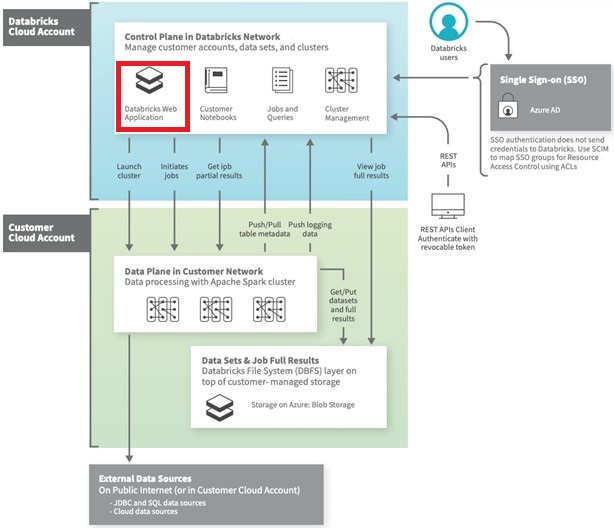
Databricks operates most of its services out of a control plane and a data plane, *please note serverless features like SQL Endpoint and DLT compute use shared compute in Control pane.*

**Control Plane:**Stored in Databricks Cloud Account

* The control plane includes the backend services that Databricks manages in its own Azure account. Notebook commands and many other workspace configurations are stored in the control plane and encrypted at rest.

**Data Plane:** Stored in Customer Cloud Account

* The data plane is managed by your Azure account and is where your data resides. This is also where data is processed. You can use Azure Databricks connectors so that your clusters can connect to external data sources outside of your Azure account to ingest data or for storage.



## Question 4

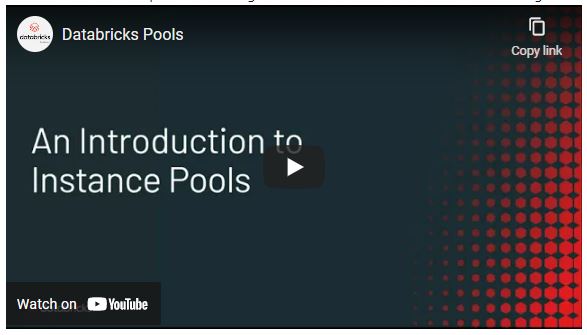
You have written a notebook to generate a summary data set for reporting, Notebook was scheduled using the job cluster, but you realized it takes 8 minutes to start the cluster, what feature can be used to start the cluster in a timely fashion so your job can run immediately?

* **Setup an additional job to run ahead of the actual job so the cluster is running second job starts**
* **Use the Databricks cluster pools feature to reduce the startup time**
* **Use Databricks Premium edition instead of Databricks standard edition**
* **Pin the cluster in the cluster UI page so it is always available to the jobs**
* **Disable auto termination so the cluster is always running**

**Explanation**

Cluster pools allow us to reserve VM's ahead of time, when a new job cluster is created VM are grabbed from the pool. Note: when the VM's are waiting to be used by the cluster only cost incurred is Azure. Databricks run time cost is only billed once VM is allocated to a cluster.

Here is a demo of how to setup a pool and follow some best practices,

[](https://www.youtube.com/watch?v=FVtITxOabxg&ab_channel=DatabricksAcademy)

## Question 5

Which of the following developer operations in the CI/CD can only be implemented through a GIT provider when using Databricks Repos.

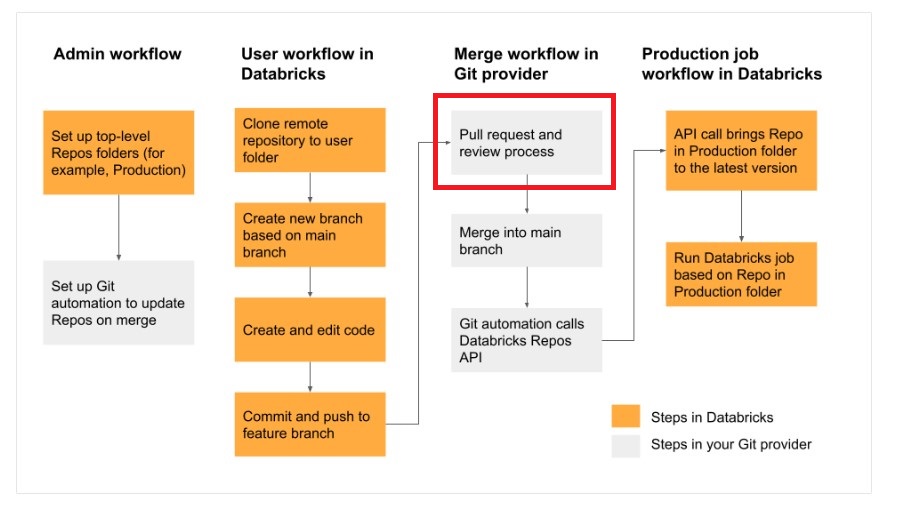
* **Trigger Databricks Repos pull API to update the latest version**
* **Commit and push code**
* **Create and edit code**
* **Create a new branch**
* **Pull request and review process**

**Explanation**

The answer is Pull request and review process, please note: the question is asking for steps that are being implemented in GIT provider not Databricks Repos.

See below diagram to understand the role of Databricks Repos and Git provider plays when building a CI/CD workdlow.

All the steps highlighted in yellow can be done Databricks Repo, all the steps highlighted in Gray are done in a **git**provider like Github or Azure Devops.



## Question 6

You have noticed the Data scientist team is using the notebook versioning feature with git integration, 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.

* **Databricks Repos allows multiple users to make changes**
* **Databricks Repos allows merge and conflict resolution**
* **Databricks Repos has a built-in version control system**
* **Databricks Repos automatically saves changes**
* **Databricks Repos allow you to add comments and select the changes you want to commit.**

**Explanation**

The answer is  Databricks Repos allow you to add comments and select the changes you want to commit.

## Question 7

Data science team members are using a single cluster to perform data analysis, although cluster size was chosen to handle multiple users and auto-scaling was enabled, the team realized queries are still running slow, what would be the suggested fix for this?

* **Setup multiple clusters so each team member has their own cluster**
* **Disable the auto-scaling feature**
* **Use High concurrency mode instead of the standard mode**
* **Increase the size of the driver node**

**Explanation**

The answer is Use High concurrency mode instead of the standard mode,

<https://docs.databricks.com/clusters/cluster-config-best-practices.html#cluster-mode>

High Concurrency clusters are ideal for groups of users who need to share resources or run ad-hoc jobs.  Databricks recommends enabling autoscaling for High Concurrency clusters.

Bottom of Form

Top of Form

Question 8: **Correct**

Which of the following SQL commands are used to append rows to an existing delta table?

* **APPEND INTO DELTA table\_name**
* **APPEND INTO table\_name**
* **COPY DELTA INTO table\_name**
* **INSERT INTO table\_name**
* **UPDATE table\_name**

**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 Datawarehouse.

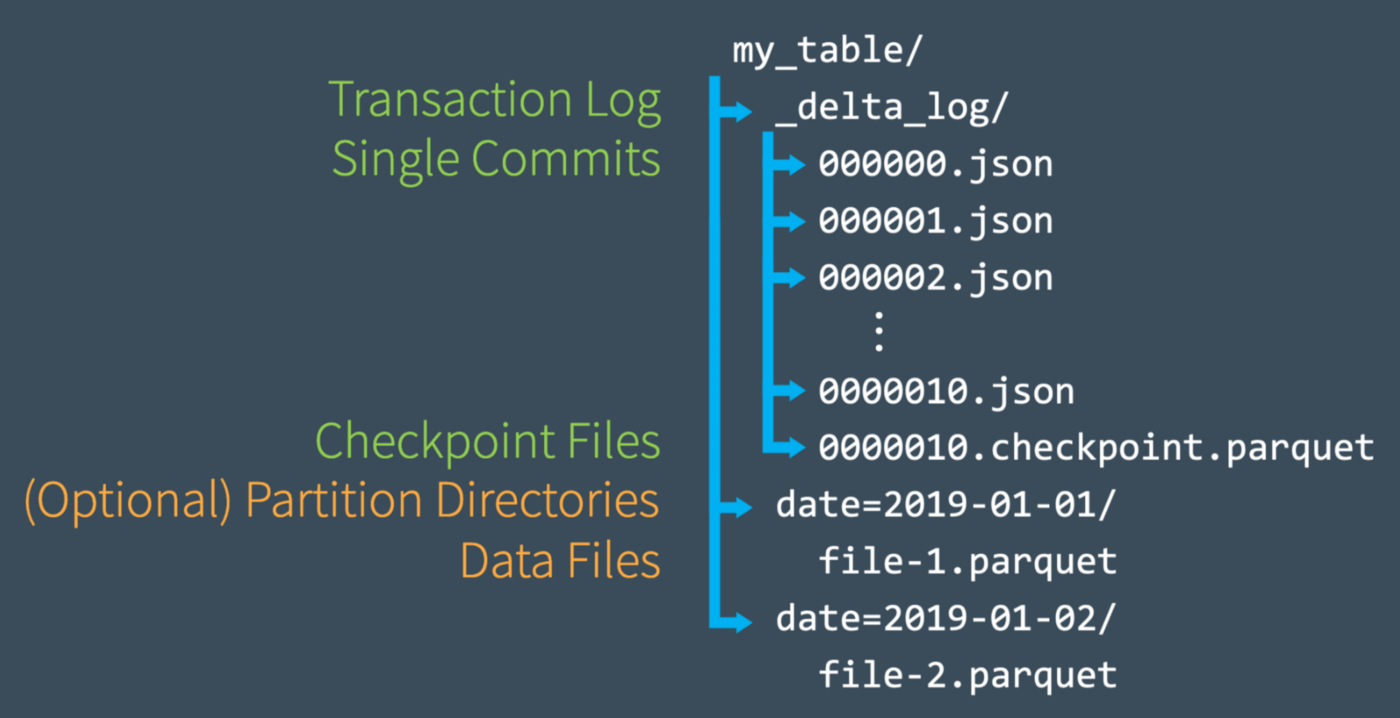
## Question 9

How are Delt tables stored?

* **A Directory where parquet data files are stored, a sub directory \_delta\_log where meta data, and the transaction log is stored as JSON files.**
* **A Directory where parquet data files are stored, all of the meta data is stored in memory**
* **A Directory where parquet data files are stored in Data plane, a sub directory \_delta\_log where meta data, history and log is stored in control pane.**
* **A Directory where parquet data files are stored, all of the metadata is stored in parquet files**
* **Data is stored in Data plane and Metadata and delta log are stored in control pane**

**Explanation**

The answer is A Directory where parquet data files are stored, a sub directory \_delta\_log where meta data, and the transaction log is stored as JSON files.



## Question 10

While investigating a data issue in a Delta table, you wanted to review logs to see when and who updated the table, what is the best way to review this data?

* **Review event logs in the Workspace**
* **Run SQL SHOW HISTORY table\_name**
* **Check Databricks SQL Audit logs**
* **Run SQL command DESCRIBE HISTORY table\_name**
* **Review workspace audit logs**

**Explanation**

The answer is Run SQL command DESCRIBE HISTORY table\_name.

here is the sample data of how DESCRIBE HISTORY table\_name looks

1. +-------+-------------------+------+--------+---------+--------------------+----+--------+---------+-----------+--------------+-------------+--------------------+
2. |version| timestamp|userId|userName|operation| operationParameters| job|notebook|clusterId|readVersion|isolationLevel|isBlindAppend| operationMetrics|
3. +-------+-------------------+------+--------+---------+--------------------+----+--------+---------+-----------+--------------+-------------+--------------------+
4. | 5|2019-07-29 14:07:47| null| null| DELETE|[predicate -> ["(...|null| null| null| 4| Serializable| false|[numTotalRows -> ...|
5. | 4|2019-07-29 14:07:41| null| null| UPDATE|[predicate -> (id...|null| null| null| 3| Serializable| false|[numTotalRows -> ...|
6. | 3|2019-07-29 14:07:29| null| null| DELETE|[predicate -> ["(...|null| null| null| 2| Serializable| false|[numTotalRows -> ...|
7. | 2|2019-07-29 14:06:56| null| null| UPDATE|[predicate -> (id...|null| null| null| 1| Serializable| false|[numTotalRows -> ...|
8. | 1|2019-07-29 14:04:31| null| null| DELETE|[predicate -> ["(...|null| null| null| 0| Serializable| false|[numTotalRows -> ...|
9. | 0|2019-07-29 14:01:40| null| null| WRITE|[mode -> ErrorIfE...|null| null| null| null| Serializable| true|[numFiles -> 2, n...|
10. +-------+-------------------+------+--------+---------+--------------------+----+--------+---------+-----------+--------------+-------------+--------------------+

## Question 11

While investigating a performance issue, you realized that you have too many small files for a given table, which command are you going to run to fix this issue

* **COMPACT table\_name**
* **VACUUM table\_name**
* **MERGE table\_name**
* **SHRINK table\_name**
* **OPTIMIZE table\_name**

**Explanation**

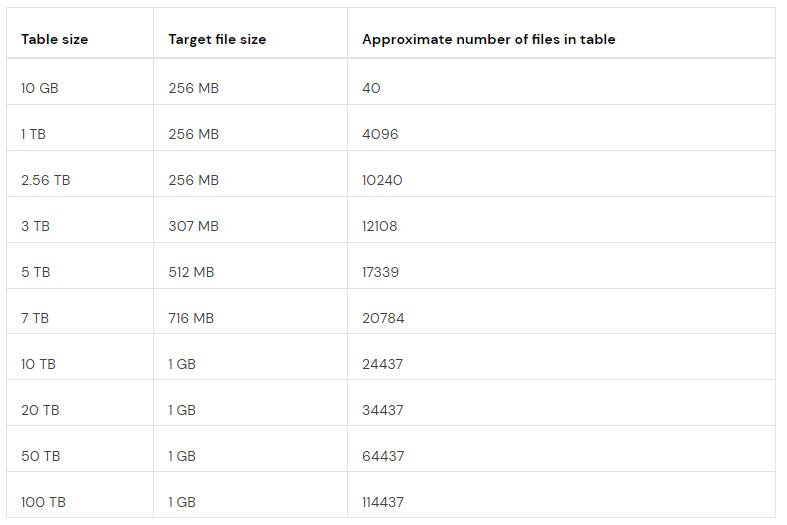
The answer is OPTIMIZE table\_name,

Optimize compacts small parquet files into a bigger file, by default the size of the files are determined based on the table size at the time of OPTIMIZE, the file size can also be set manually or adjusted based on the workload.

<https://docs.databricks.com/delta/optimizations/file-mgmt.html>

Tune file size based on Table size

To minimize the need for manual tuning, Databricks automatically tunes the file size of Delta tables based on the size of the table. Databricks will use smaller file sizes for smaller tables and larger file sizes for larger tables so that the number of files in the table does not grow too large.



## Question 12

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

* **CREATE DATABASE sales FORMAT DELTA LOCATION 'dbfs:/mnt/delta/databases/sales.db/'’**
* **CREATE DATABASE sales USING LOCATION 'dbfs:/mnt/delta/databases/sales.db/'**
* **CREATE DATABASE sales LOCATION 'dbfs:/mnt/delta/databases/sales.db/'**
* **The sales database can only be created in Delta lake**
* **CREATE DELTA DATABASE sales LOCATION 'dbfs:/mnt/delta/databases/sales.db/'**

**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

## Question 13

What is the type of table created when you issue SQL DDL command CREATE TABLE sales (id int, units int)

* **Query fails due to missing location**
* **Query fails due to missing format**
* **Managed Delta table**
* **External Table**
* **Managed Parquet table**

**Explanation**

Answer is Managed Delta table

Anytime a table is created without the Location keyword it is considered a managed table, by default all managed tables DELTA tables

Syntax

CREATE TABLE table\_name ( column column\_data\_type…)

## Question 14

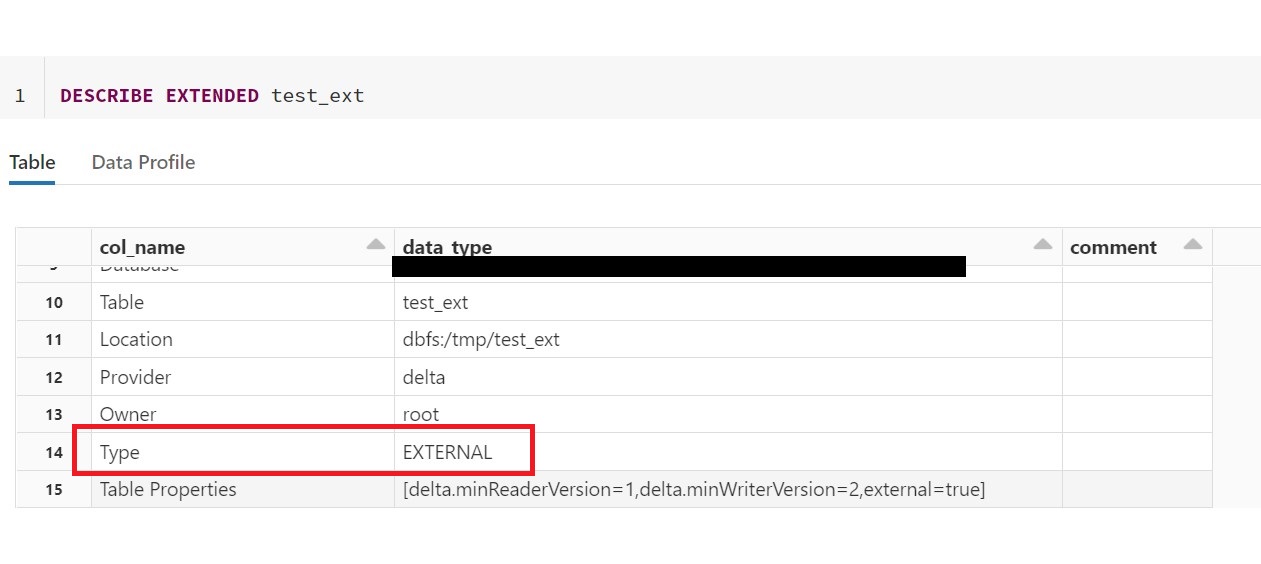
How to determine if a table is a managed table vs external table?

* **Run IS\_MANAGED(‘table\_name’) function**
* **All external tables are stored in data lake, managed tables are stored in DELTA lake**
* **All managed tables are stored in unity catalog**
* **Run SQL command DESCRIBE EXTENDED table\_name and check type**
* **Run SQL command SHOW TABLES to see the type of the table**

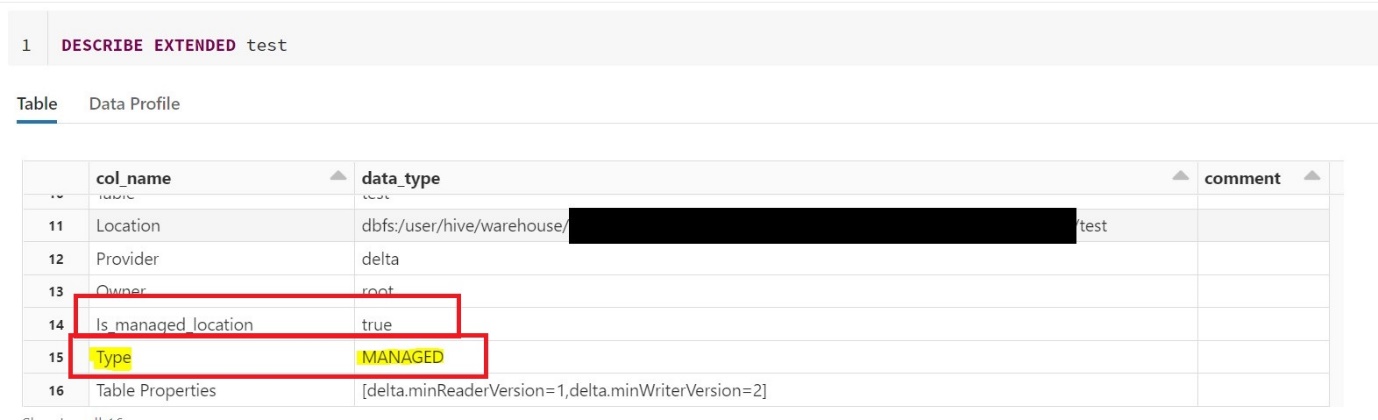
**Explanation**

The answer is Run SQL command DESCRIBE EXTENDED table\_name and check type

Example of External table



Example of managed table



## Question 15

Which of the below SQL commands creates a session scoped temporary view?

CREATE OR REPLACE TEMPORARY VIEW view\_name

AS SELECT \* FROM table\_name

CREATE OR REPLACE LOCAL TEMPORARY VIEW view\_name

AS SELECT \* FROM table\_name

CREATE OR REPLACE GLOBAL TEMPORARY VIEW view\_name

AS SELECT \* FROM table\_name

CREATE OR REPLACE VIEW view\_name

AS SELECT \* FROM table\_name

CREATE OR REPLACE LOCAL VIEW view\_name

AS SELECT \* FROM table\_name

**Explanation**

The answer is

1. CREATE OR REPLACE TEMPORARY VIEW view\_name
2. AS SELECT \* FROM table\_name

The default temporary view is session scoped, as soon as the session ends or if a notebook is detached session scoped temporary view is dropped.

Bottom of Form

## Question 16

Drop the customers database and associated tables and data, all of the tables inside the database are managed tables. Which of the following SQL commands will help you accomplish this?

**DROP DATABASE customers FORCE**

**DROP DATABASE customers CASCADE**

**DROP DATABASE customers  INCLUDE**

**All the tables must be dropped first before dropping database**

**DROP DELTA DATABSE customers**

**Explanation**

The answer is DROP DATABASE customers CASCADE

Drop database with cascade option drops all the tables, since all of the tables inside the database are managed tables we do not need to perform any additional steps to clean the data in the storage.

Bottom of Form

Top of Form

## Question 17

Define an external SQL table by connecting to a local instance of an SQLite database using JDBC

CREATE TABLE users\_jdbc

USING SQLITE

OPTIONS (

url = "jdbc:/sqmple\_db",

dbtable = "users"

)

CREATE TABLE users\_jdbc

USING SQL

URL = {server:"jdbc:/sqmple\_db",dbtable: “users”}

CREATE TABLE users\_jdbc

USING SQL

OPTIONS (

url = "jdbc:sqlite:/sqmple\_db",

dbtable = "users"

)

CREATE TABLE users\_jdbc

USING org.apache.spark.sql.jdbc.sqlite

OPTIONS (

url = "jdbc:/sqmple\_db",

dbtable = "users"

)

CREATE TABLE users\_jdbc

USING org.apache.spark.sql.jdbc

OPTIONS (

url = "jdbc:sqlite:/sqmple\_db",

dbtable = "users"

)

**Explanation**

The answer is,

CREATE TABLE users\_jdbc

USING org.apache.spark.sql.jdbc

OPTIONS (

url = "jdbc:sqlite:/sqmple\_db",

dbtable = "users"

)

Databricks runtime currently supports connecting to a few flavors of SQL Database including SQL Server, My SQL, SQL Lite and Snowflake using JDBC.

CREATE TABLE <jdbcTable>

USING org.apache.spark.sql.jdbc or JDBC

OPTIONS (

url = "jdbc:<databaseServerType>://<jdbcHostname>:<jdbcPort>",

dbtable " = <jdbcDatabase>.atable",

user = "<jdbcUsername>",

password = "<jdbcPassword>"

)

For more detailed documentation

[SQL databases using JDBC - Azure Databricks | Microsoft Docs](https://docs.microsoft.com/en-us/azure/databricks/data/data-sources/sql-databases)

## Question 18

When defining external tables using formats CSV, JSON, TEXT, BINARY any query on the external tables caches the data and location for performance reasons, so within a given spark session any new files that may have arrived will not be available after the initial query. How can we address this limitation?

* **UNCACHE TABLE table\_name**
* **CACHE TABLE table\_name**
* **REFRESH TABLE table\_name**
* **BROADCAST TABLE table\_name**
* **CLEAR CACH table\_name**

**Explanation**

The answer is REFRESH TABLE table\_name

REFRESH TABLE table\_name  will force Spark to refresh the availability of external files and any changes.

When spark queries an external table it caches the files associated with it, so that way if the table is queried again it can use the cached files so it does not have to retrieve them again from cloud object storage, but the drawback here is that if new files are available Spark does not know until the Refresh command is ran.

## Question 19

Which of the following table constraints that can be enforced on Delta lake tables are supported?

* **Primary key, foreign key, Not Null, Check Constraints**
* **Primary key, Not Null, Check Constraints**
* **Default, Not Null, Check Constraints**
* **Not Null, Check Constraints**
* **Unique, Not Null, Check Constraints**

**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

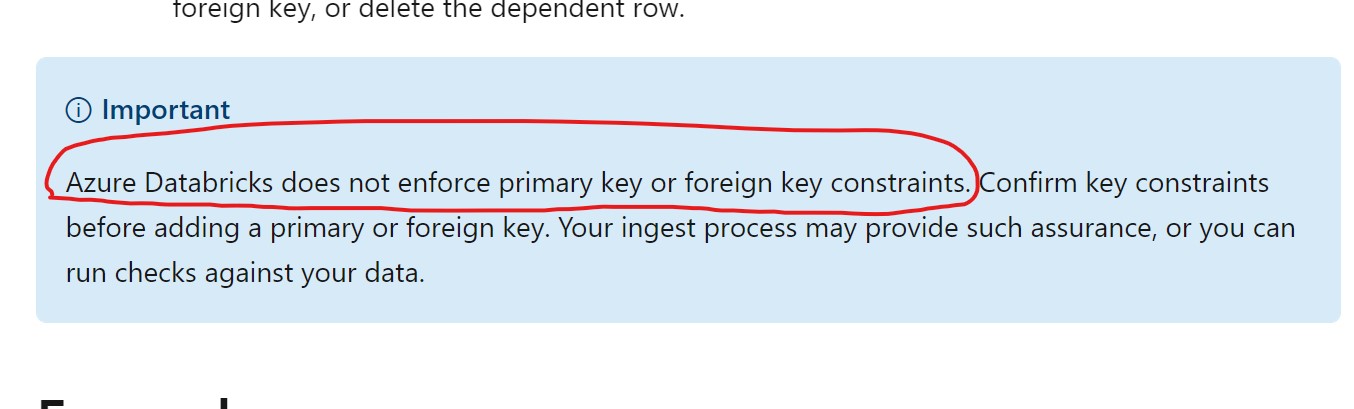
) USING DELTA;

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](https://learn.microsoft.com/en-us/azure/databricks/spark/latest/spark-sql/language-manual/sql-ref-syntax-ddl-alter-table-add-constraint)



## Question 20

The data engineering team is looking to add a new column to the table, but the QA team would like to test the change before implementing in production, which of the below options allow you to quickly copy the table from Prod to the QA environment, modify and run the tests?

* **DEEP CLONE**
* **SHADOW CLONE**
* **ZERO COPY CLONE**
* **SHALLOW CLONE**
* **METADATA CLONE**

**Explanation**

The answer is SHALLOW CLONE

**SHALLOW CLONE**If you wish to create a copy of a table quickly to test out applying changes without the risk of modifying the current table, **SHALLOW CLONE** can be a good option. Shallow clones just copy the Delta transaction logs, meaning that the data doesn't move so it can be very quick.

1. CREATE OR REPLACE TABLE {new\_table\_name} SHALLOW CLONE {source\_table\_name}|[LOCATION path]

**DEEP CLONE** fully copies data and metadata from a source table to a target. This copy occurs incrementally, so executing this command again can sync changes from the source to the target location. It copies all of the data and transaction logs this can take a long time based on the size of the table.

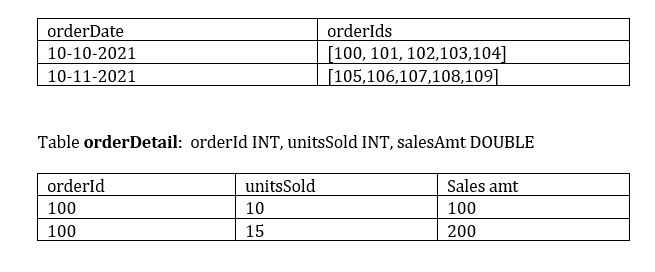
1. CREATE OR REPLACE TABLE {new\_table\_name} DEEP CLONE {source\_table\_name}|[LOCATION path]

## Question 21

Sales team is looking to get a report on a measure number of units sold by date, below is the schema. Fill in the blank with the appropriate array function.

Table **orders**: orderDate DATE, orderIds ARRAY<INT>

Table **orderDetail**: orderId INT, unitsSold INT, salesAmt DOUBLE

****

SELECT orderDate, SUM(unitsSold)

FROM orderDetail od

JOIN (select orderDate, \_\_\_\_\_\_\_\_\_\_\_(orderIds) as orderId FROM orders) o

ON o.orderId = od.orderId

GROUP BY orderDate

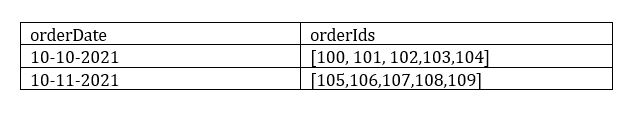
* **FLATTEN**
* **EXTEND**
* **EXPLODE**
* **EXTRACT**
* **ARRAY\_FLATTEN**

**Explanation**

The answer is EXPLODE,

explode is table-valued function, takes an array or map and returns a row for each element in the array.

For below table



select explode(orderIds) orderId, orderdate from orders

This above would return a result something like below, creates a row for each element in the array

OrderDate            orderId

10-10-2021            100

10-10-2021            101

10-10-2021            102

## Question 22

You are asked to write a python function that can read data from a delta table and return the DataFrame, which of the following is correct?

* **Python function cannot return a DataFrame**
* **Write SQL UDF to return a DataFrame**
* **Write SQL UDF that can return tabular data**
* **Python function will result in out of memory error due to data volume**
* **Python function can return a DataFrame**

**Explanation**

The answer is Python function can return a DataFrame

The function would something like this,

1. get\_source\_dataframe(tablename):
2. df = spark.read.table(tablename)
3. return df

df = get\_source\_dataframe('test\_table')

since there is no action spark returns a Dataframe and assigns to df python variable

## Question 23

What is the output of the below function when executed with input parameters 1, 3  :

def check\_input(x,y):

if x < y:

x= x+1

if x<y:

x= x+1

if x <y:

x = x+1

return x

check\_input(1,3)

* **1**
* **2**
* **3**
* **4**
* **5**

**Explanation**

The answer is 3

## Question 24

Which of the following SQL statements can replace a python variable, when the notebook is set in SQL mode

1. table\_name = "sales"
2. schema\_name = "bronze"

* **spark.sql(f"SELECT \* FROM f{schema\_name.table\_name}")**
* **spark.sql(f"SELECT \* FROM {schem\_name.table\_name}")**
* **spark.sql(f"SELECT \* FROM ${schema\_name}.${table\_name}")**
* **spark.sql(f"SELECT \* FROM {schema\_name}.{table\_name}")**
* **spark.sql("SELECT \* FROM schema\_name.table\_name")**

**Explanation**

The answer is spark.sql(f"SELECT \* FROM {schema\_name}.{table\_name}")

Bottom of Form

## Question 25

When writing streaming data, Spark’s structured stream supports the below write modes

* **Append, Delta, Complete**
* **Delta, Complete, Continuous**
* **Append, Complete, Update**
* **Complete, Incremental, Update**
* **Append, overwrite, Continuous**

**Explanation**

The answer is Append, Complete, Update

* **Append mode (default)** - This is the default mode, where only the new rows added to the Result Table since the last trigger will be outputted to the sink. This is supported for only those queries where rows added to the Result Table is never going to change. Hence, this mode guarantees that each row will be output only once (assuming fault-tolerant sink). For example, queries with only select, where, map, flatMap, filter, join, etc. will support Append mode.
* **Complete mode** - The whole Result Table will be outputted to the sink after every trigger. This is supported for aggregation queries.
* **Update mode** - (*Available since Spark 2.1.1*) Only the rows in the Result Table that were updated since the last trigger will be outputted to the sink. More information to be added in future releases.

## Question 26

When using the complete mode to write stream data, how does it impact the target table?

* **Entire stream waits for complete data to write**
* **Stream must complete to write the data**
* **Target table cannot be updated while stream is pending**
* **Target table is overwritten for each batch**
* **Delta commits transaction once the stream is stopped**

**Explanation**

The answer is Target table is overwritten for each batch

**Complete mode** - The whole Result Table will be outputted to the sink after every trigger. This is supported for aggregation queries

## Question 27

At the end of the inventory process a file gets uploaded to the cloud object storage, you are asked to build a process to ingest data which of the following method can be used to ingest the data incrementally, the schema of the file is expected to change overtime ingestion process should be able to handle these changes automatically. Below is the auto loader command to load the data, fill in the blanks for successful execution of the below code.

1. spark.readStream
2. .format("cloudfiles")
3. .option("cloudfiles.format",”csv)
4. .option("\_\_\_\_\_\_\_", ‘dbfs:/location/checkpoint/’)
5. .load(data\_source)
6. .writeStream
7. .option("\_\_\_\_\_\_\_",’ dbfs:/location/checkpoint/’)
8. .option("mergeSchema", "true")
9. .table(table\_name))

* **checkpointlocation, schemalocation**
* **checkpointlocation, cloudfiles.schemalocation**
* **schemalocation, checkpointlocation**
* **cloudfiles.schemalocation, checkpointlocation**
* **cloudfiles.schemalocation, cloudfiles.checkpointlocation**

**Explanation**

The answer is cloudfiles.schemalocation, checkpointlocation

When reading the data cloudfiles.schemalocation is used to store the inferred schema of the incoming data.

When writing a stream to recover from failures checkpointlocation is used to store the offset of the byte that was most recently processed.

## Question 28

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?

* **Provide the schema of the source table in the cloudfiles.schemalocation**
* **Provide the schema of the target table in the cloudfiles.schemalocation**
* **Provide schema hints**
* **Update the checkpoint location**
* **Correct the incoming data by explicitly casting the data types**

**Explanation**

The answer is, Provide schema hints.

spark.readStream \

.format("cloudFiles") \

.option("cloudFiles.format", "csv") \

.option("header", "true") \

.option("cloudFiles.schemaLocation", schema\_location) \

.option("cloudFiles.schemaHints", "id int, description string")

.load(raw\_data\_location)

.writeStream \

.option("checkpointLocation", checkpoint\_location) \

.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.

## Question 29

You have configured AUTO LOADER to process incoming IOT data from cloud object storage every 15 mins, recently a change was made to the notebook code to update the processing logic but the team later realized that the notebook was failing for the last 24 hours, what steps team needs to take to reprocess the data that was not loaded after the notebook was corrected?

* **Move the files that were not processed to another location and manually copy the files into the ingestion path to reprocess them**
* **Enable back\_fill = TRUE to reprocess the data**
* **Delete the checkpoint folder and run the autoloader again**
* **Autoloader automatically re-processes data that was not loaded**
* **Manually re-load the data**

**Explanation**

The answer is,

Autoloader automatically re-processes data that was not loaded using the checkpoint.

## Question 30

Which of the following Structured Streaming queries is performing a hop from a bronze table to a Silver table?

(spark.table("sales").groupBy("store")

.agg(sum("sales")).writeStream

.option("checkpointLocation",checkpointPath)

.outputMode("complete")

.table("aggregatedSales"))

(spark.table("sales").agg(sum("sales"),sum("units"))

.writeStream

.option("checkpointLocation",checkpointPath)

.outputMode("complete")

.table("aggregatedSales"))

(spark.table("sales")

.withColumn("avgPrice", col("sales") / col("units"))

.writeStream

.option("checkpointLocation", checkpointPath)

.outputMode("append")

.table("cleanedSales"))

(spark.readStream.load(rawSalesLocation)

.writeStream

.option("checkpointLocation", checkpointPath)

.outputMode("append")

.table("uncleanedSales") )

(spark.read.load(rawSalesLocation)

.writeStream

.option("checkpointLocation", checkpointPath)

.outputMode("append")

.table("uncleanedSales") )

**Explanation**

The question is asking to identify a structured streaming command that is moving data from bronze to silver.

The answer is

(spark.table("sales")

.withColumn("avgPrice", col("sales") / col("units"))

.writeStream

.option("checkpointLocation", checkpointPath)

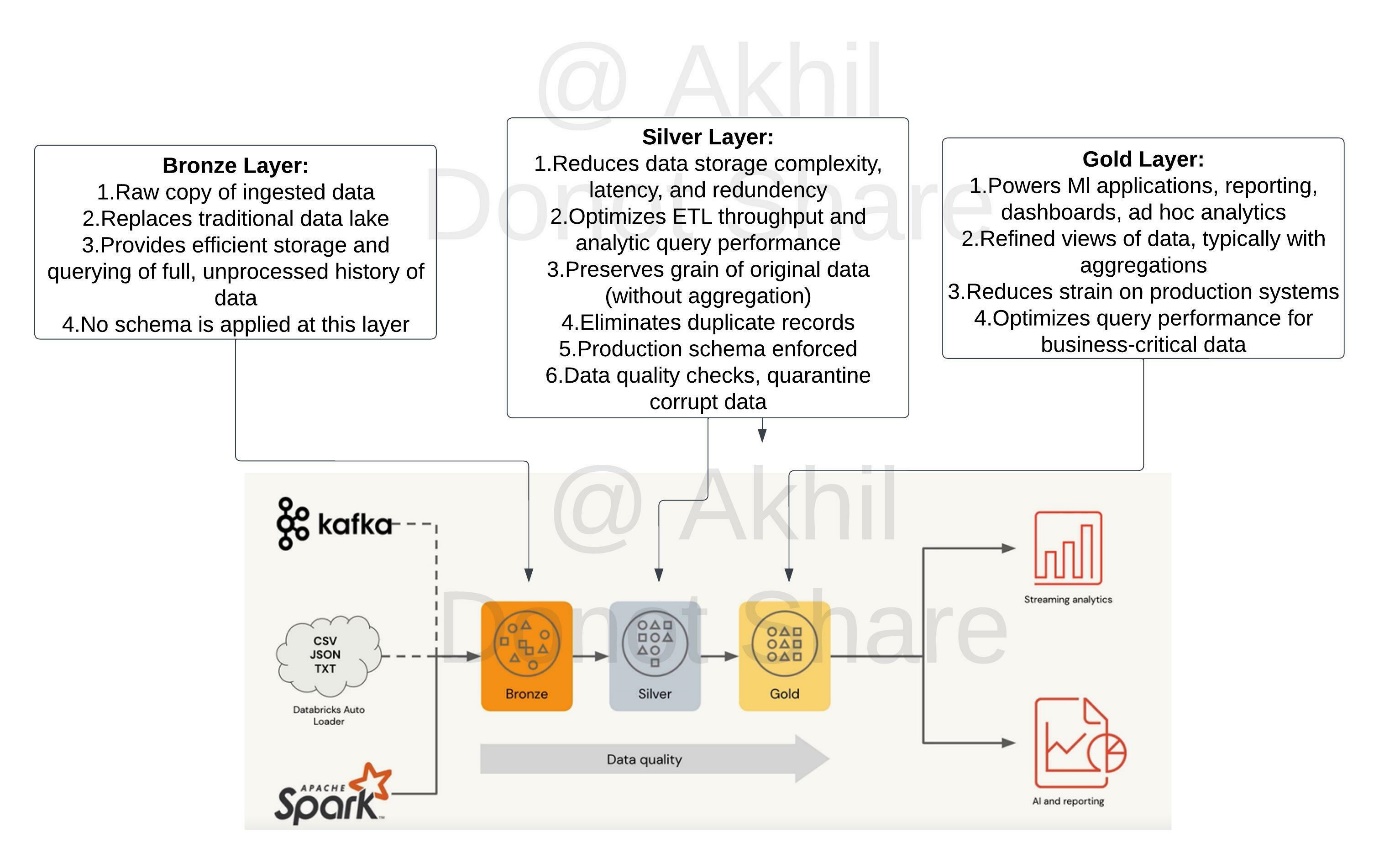
.outputMode("append")

.table("cleanedSales"))

We are preserving the grain of incoming data and enriching the data by adding avg price, the other options listed use aggregations which are mostly performed on top of the silver to move data to Gold.

***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.*



## Question 31

Which of the following Structured Streaming queries successfully performs a hop from a Silver to Gold table?

(spark.table("sales")

.groupBy("store")

.agg(sum("sales"))

.writeStream

.option("checkpointLocation", checkpointPath)

.outputMode("complete")

.table("aggregatedSales") )

(spark.table("sales")

.writeStream

.option("checkpointLocation", checkpointPath)

.outputMode("complete")

.table("sales") )

(spark.table("sales")

.withColumn("avgPrice", col("sales") / col("units"))

.writeStream

.option("checkpointLocation", checkpointPath)

.outputMode("append")

.table("cleanedSales") )

(spark.readStream.load(rawSalesLocation)

.writeStream

.option("checkpointLocation", checkpointPath)

.outputMode("append")

.table("uncleanedSales") )

(spark.read.load(rawSalesLocation)

.writeStream

.option("checkpointLocation", checkpointPath)

.outputMode("append")

.table("uncleanedSales") )

**Explanation**

The answer is

1. (spark.table("sales")
2. .groupBy("store")
3. .agg(sum("sales"))
4. .writeStream
5. .option("checkpointLocation", checkpointPath)
6. .outputMode("complete")
7. .table("aggregatedSales") )

The gold layer is normally used to store aggregated data

Review the below link for more info,

[Medallion Architecture – Databricks](https://databricks.com/glossary/medallion-architecture)

Gold Layer:

1. Powers Ml applications, reporting, dashboards, ad hoc analytics

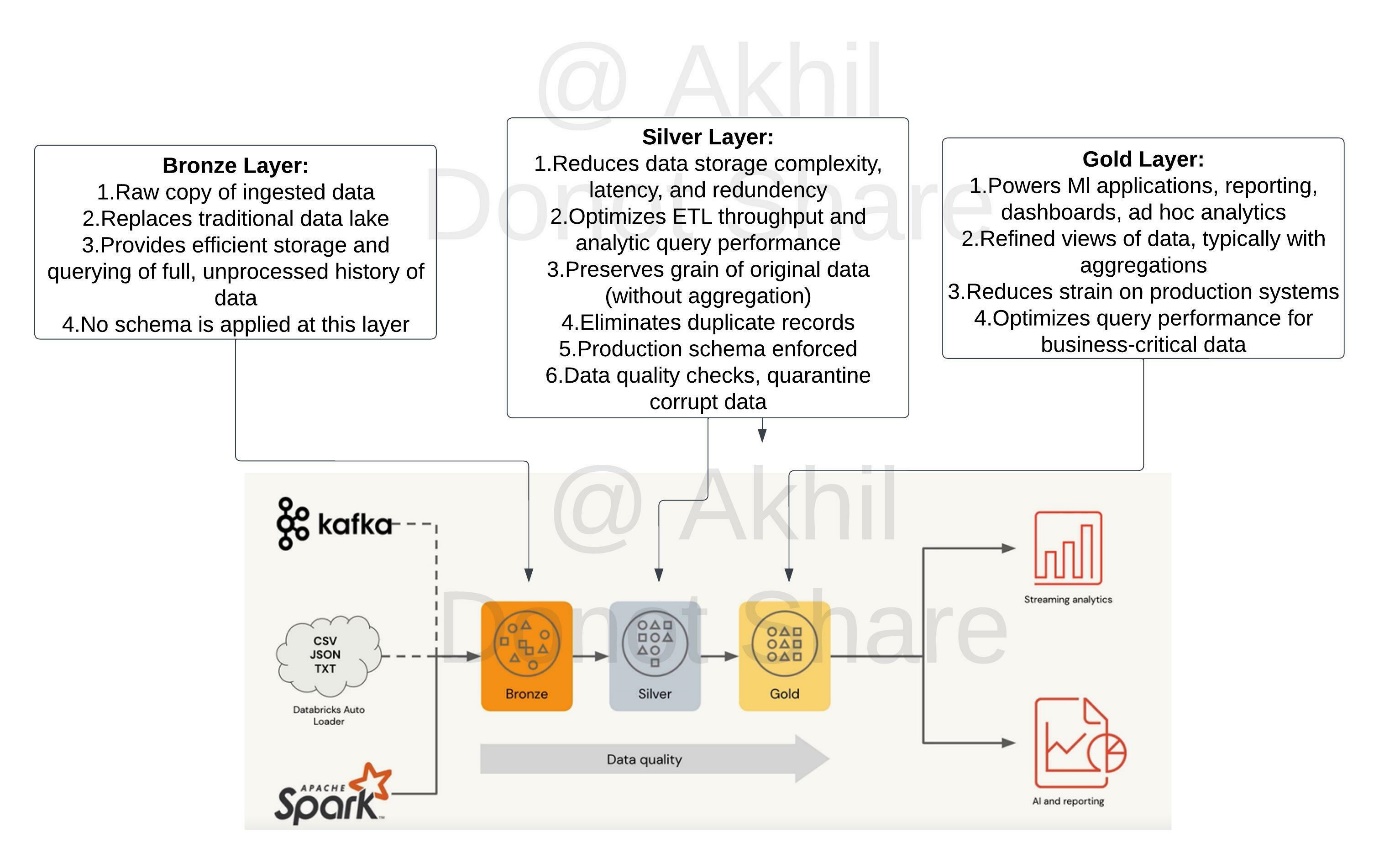
2. **Refined views of data, typically with aggregations**

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.*



## Question 32

Which of the following Auto loader structured streaming commands successfully performs a hop from the landing area into Bronze?

spark\

.readStream\

.format("csv")\

.option("cloudFiles.schemaLocation", checkpoint\_directory)\

.load("landing")\

.writeStream.option("checkpointLocation", checkpoint\_directory)\

.table(raw)

spark\

.readStream\

.format("cloudFiles")\

.option("cloudFiles.format","csv")\

.option("cloudFiles.schemaLocation", checkpoint\_directory)\

.load("landing")\

.writeStream.option("checkpointLocation", checkpoint\_directory)\

.table(raw)

spark\

.read\

.format("cloudFiles")\

.option("cloudFiles.format",”csv”)\

.option("cloudFiles.schemaLocation", checkpoint\_directory)\

.load("landing")\

.writeStream.option("checkpointLocation", checkpoint\_directory)\

.table(raw)

spark\

.readStream\

.load(rawSalesLocation)\

.writeStream \

.option("checkpointLocation", checkpointPath).outputMode("append")\

.table("uncleanedSales")

spark\

.read\

.load(rawSalesLocation) \

.writeStream\

.option("checkpointLocation", checkpointPath) \

.outputMode("append")\

.table("uncleanedSales")

**Explanation**

The answer is

spark\

.readStream\

.format("cloudFiles") \# use Auto loader

.option("cloudFiles.format","csv") \ # csv format files

.option("cloudFiles.schemaLocation", checkpoint\_directory)\

.load('landing')\

.writeStream.option("checkpointLocation", checkpoint\_directory)\

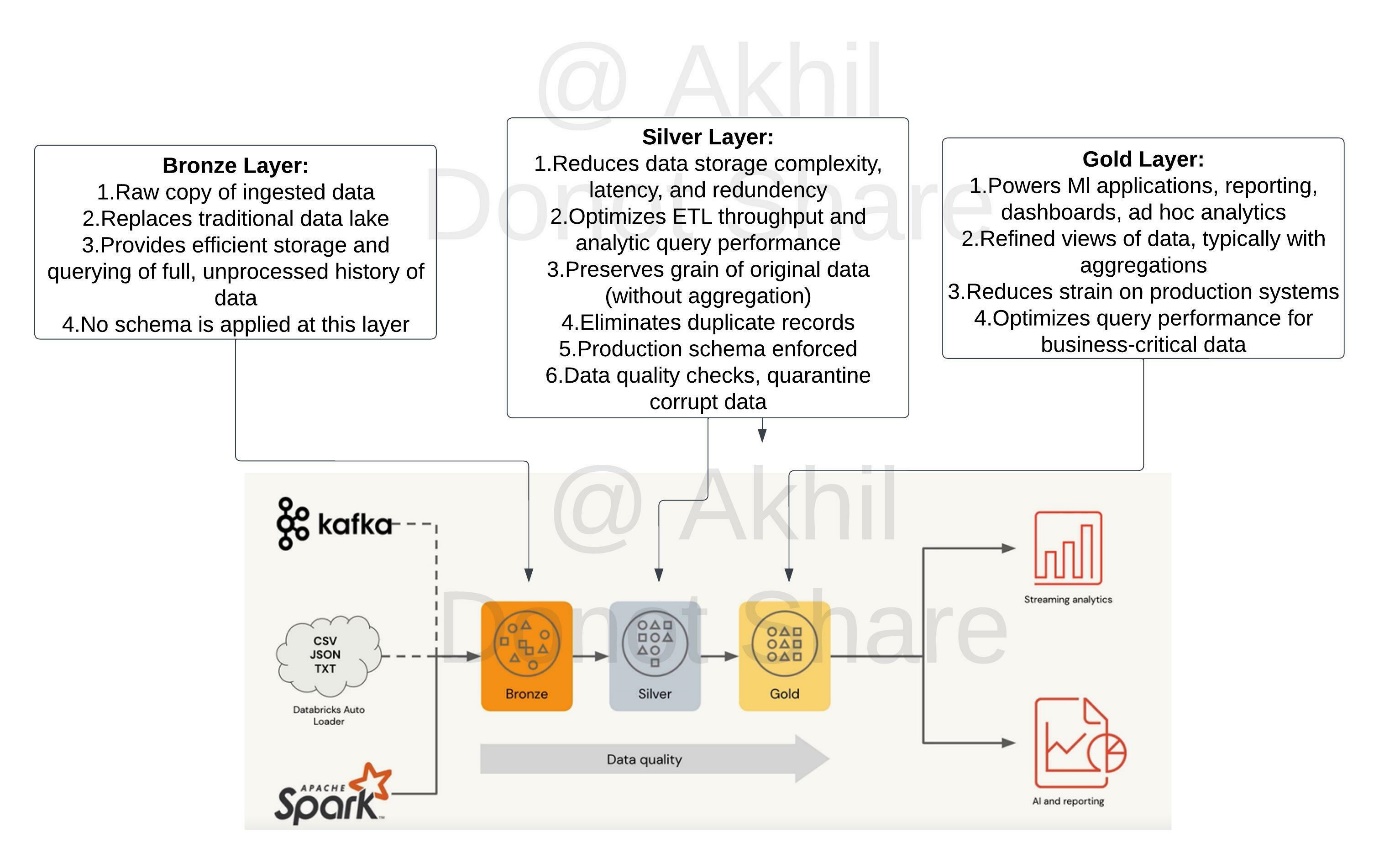
.table(raw)

Note: if you chose the below option which is incorrect because it does not have readStream

1. spark.read.format("cloudFiles")
2. .option("cloudFiles.format",”csv”)
3. ...
4. ..
5. ..

***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.*



## Question 33

A DELTA LIVE TABLE pipelines can be scheduled to run in two different modes, what are these two different modes?

* **Triggered, Incremental**
* **Once, Continuous**
* **Triggered, Continuous**
* **Once, Incremental**
* **Continuous, Incremental**

**Explanation**

The answer is Triggered, Continuous

<https://docs.microsoft.com/en-us/azure/databricks/data-engineering/delta-live-tables/delta-live-tables-concepts#--continuous-and-triggered-pipelines>

* Triggered pipelines update each table with whatever data is currently available and then stop the cluster running the pipeline. Delta Live Tables automatically analyzes the dependencies between your tables and starts by computing those that read from external sources. Tables within the pipeline are updated after their dependent data sources have been updated.
* Continuous pipelines update tables continuously as input data changes. Once an update is started, it continues to run until manually stopped. Continuous pipelines require an always-running cluster but ensure that downstream consumers have the most up-to-date data.

## Question 34

Your team member is trying to set up a delta pipeline and build a second gold table to the same pipeline with aggregated metrics based on an existing Delta Live table called sales\_orders\_cleaned but he is facing a problem in starting the pipeline, the pipeline is failing to state it cannot find the table sales\_orders\_cleaned, you are asked to identify and fix the problem.

CREATE LIVE TABLE sales\_order\_in\_chicago

AS

SELECT order\_date, city, sum(price) as sales,

FROM sales\_orders\_cleaned

WHERE city = 'Chicago')

GROUP BY order\_date, city

* **Use STREAMING LIVE instead of LIVE table**
* **Delta live table can be used in a group by clause**
* **Delta live tables pipeline can only have one table**
* **Sales\_orders\_cleaned table is missing schema name LIVE**
* **The pipeline needs to be deployed so the first table is created before we add a second table**

**Explanation**

The answer is, Sales\_orders\_cleaned table is missing schema name LIVE

Every Delta live table should have schema LIVE

Here is the correct syntax,

1. CREATE LIVE TABLE sales\_order\_in\_chicago
2. AS
3. SELECT order\_date, city, sum(price) as sales,
4. FROM LIVE.sales\_orders\_cleaned
5. WHERE city = 'Chicago')
6. GROUP BY order\_date, city

## Question 35

Which of the following type of tasks cannot setup through a job?

* + **Notebook**
  + **DELTA LIVE PIPELINE**
  + **Spark Submit**
  + **Python**
  + **Databricks SQL Dashboard refresh**

## Question 36

Which of the following approaches can the data engineer use to obtain a version-controllable configuration of the Job’s schedule and configuration?

* + **They can link the Job to notebooks that are a part of a Databricks Repo.**
  + **They can submit the Job once on a Job cluster.**
  + **They can download the JSON equivalent of the job from the Job’s page.**
  + **They can submit the Job once on an all-purpose cluster.**
  + **They can download the XML description of the Job from the Job’s page**

## Question 37

What steps need to be taken to set up a DELTA LIVE PIPELINE as a job using the workspace UI?

* **DELTA LIVE TABLES do not support job cluster**
* **Select Workflows UI and Delta live tables tab, under task type select Delta live tables pipeline and select the notebook**
* **Select Workflows UI and Delta live tables tab, under task type select Delta live tables pipeline and select the pipeline JSON file**
* **Use Pipeline creation UI, select a new pipeline and job cluster**

**Explanation**

The answer is,

Select Workflows UI and Delta live tables tab, under task type select Delta live tables pipeline and select the notebook.

**Create a pipeline**

To create a new pipeline using the Delta Live Tables notebook:

1. Click  **Workflows** in the sidebar, click the **Delta Live Tables** tab, and click **Create Pipeline**.
2. Give the pipeline a name and click  to select a notebook.
3. Optionally enter a storage location for output data from the pipeline. The system uses a default location if you leave **Storage Location** empty.
4. Select **Triggered** for **Pipeline Mode**.
5. Click **Create**.

The system displays the **Pipeline Details** page after you click **Create**. You can also access your pipeline by clicking the pipeline name in the **Delta Live Tables** tab.

## Question 38

Data engineering team has provided 10 queries and asked Data Analyst team to build a dashboard and refresh the data every day at 8 AM, identify the best approach to set up data refresh for this dashaboard?

* + **Each query requires a separate task and setup 10 tasks under a single job to run at 8 AM to refresh the dashboard**
  + **The entire dashboard with 10 queries can be refreshed at once, single schedule needs to be set up to refresh at 8 AM.**
  + **Setup JOB with linear dependency to all load all 10 queries into a table so the dashboard can be refreshed at once.**
  + **A dashboard can only refresh one query at a time, 10 schedules to set up the refresh.**
  + **Use Incremental refresh to run at 8 AM every day.**

**Explanation**

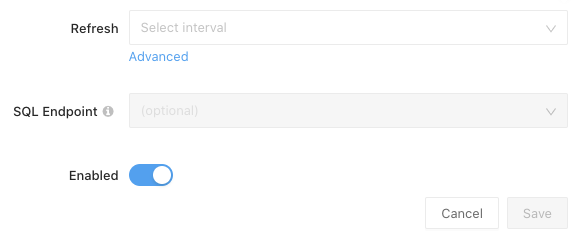
The answer is,

The entire dashboard with 10 queries can be refreshed at once, single schedule needs to be set up to refresh at 8 AM.

**Automatically refresh a dashboard**

A dashboard’s owner and users with the **Can Edit** permission can configure a dashboard to automatically refresh on a schedule. To automatically refresh a dashboard:

1. Click the **Schedule** button at the top right of the dashboard. The scheduling dialog appears.



1. In the **Refresh every** drop-down, select a period.
2. In the **SQL Warehouse** drop-down, optionally select a SQL warehouse to use for all the queries. If you don’t select a warehouse, the queries execute on the last used SQL warehouse.
3. Next to **Subscribers**, optionally enter a list of email addresses to notify when the dashboard is automatically updated.

Each email address you enter must be associated with a Azure Databricks account or configured as an [alert destination](https://docs.microsoft.com/en-us/azure/databricks/sql/admin/alert-destinations#create-an-alert-destination).

1. Click **Save**. The **Schedule** button label changes to **Scheduled**.

## Question 39

The data engineering team is using a SQL query to review data completeness every day to monitor the ETL job, and query output is being used in multiple dashboards which of the following approaches can be used to set up a schedule and automate this process?

* **They can schedule the query to run every day from the Jobs UI.**
* **They can schedule the query to refresh every day from the query’s page in Databricks SQL**
* **They can schedule the query to run every 12 hours from the Jobs UI.**
* **They can schedule the query to refresh every day from the SQL endpoint’s page in Databricks SQL.**
* **They can schedule the query to refresh every 12 hours from the SQL endpoint’s page in Databricks SQL**

**Explanation**

The answer is They can schedule the query to refresh every 12 hours from the SQL endpoint’s page in Databricks SQL,

The query pane view in Databricks SQL workspace provides the ability to add or edit and schedule individual queries to run.

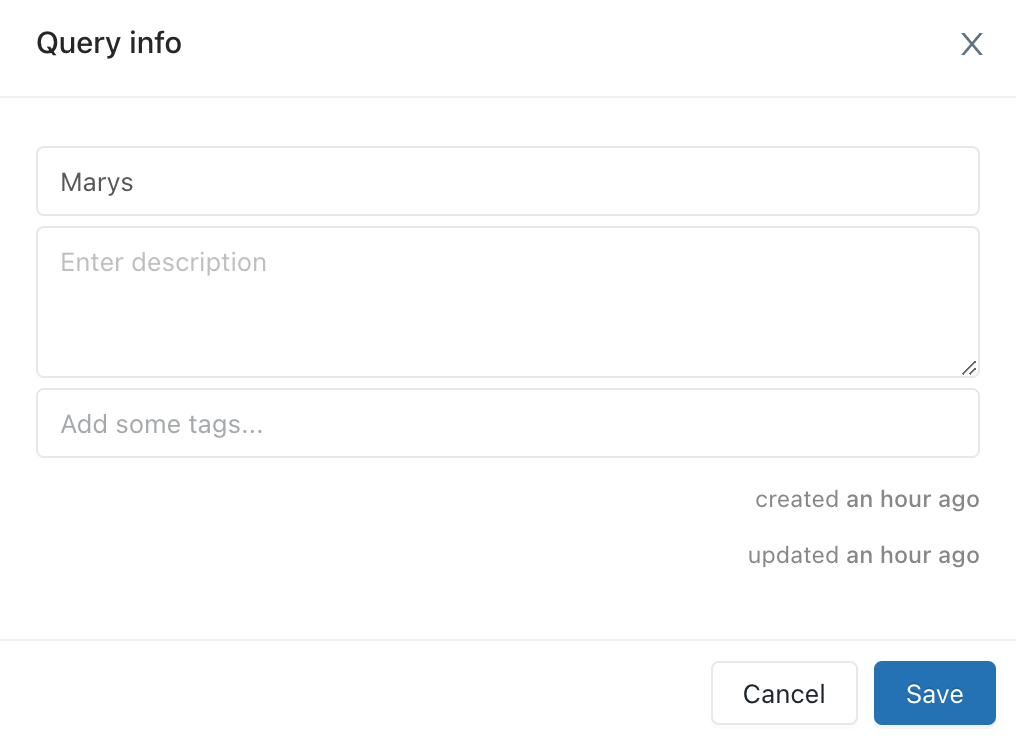
You can use scheduled query executions to keep your dashboards updated or to enable routine alerts. By default, your queries do not have a schedule.

Note

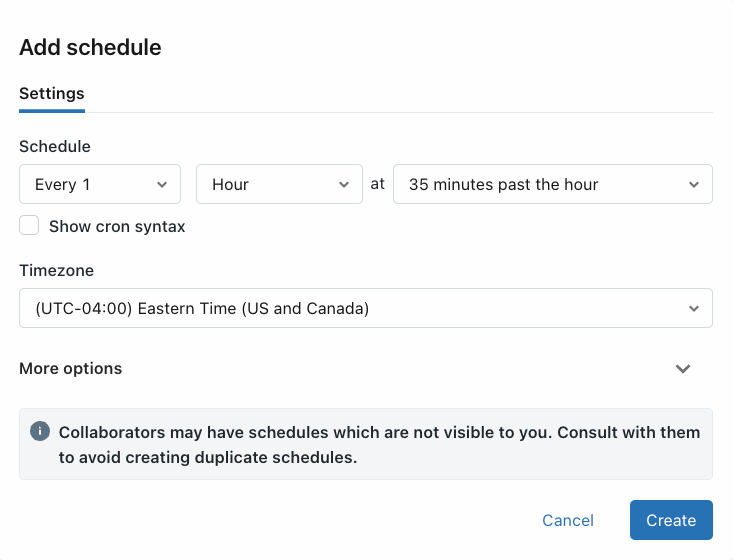
If your query is used by an alert, the alert runs on its own refresh schedule and does not use the query schedule.

To set the schedule:

1. Click the query info tab.



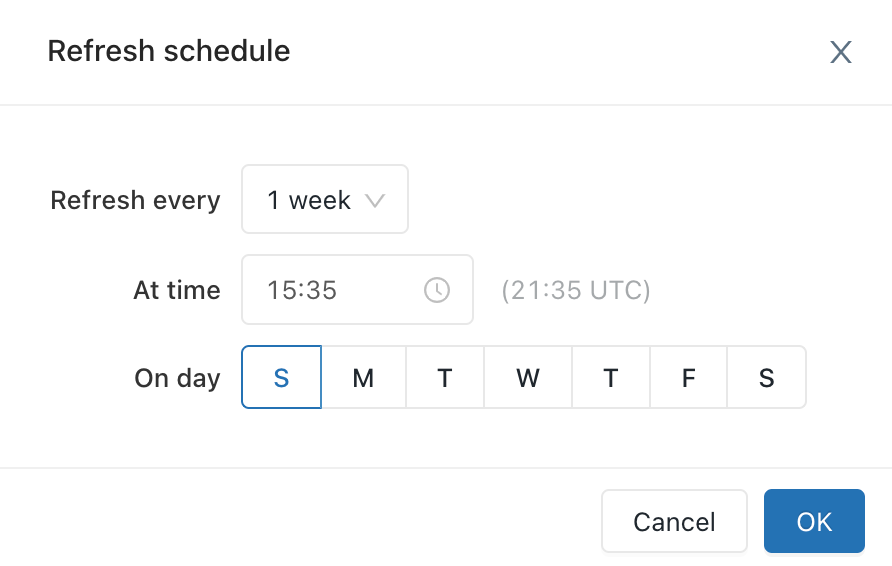
1. Click the link to the right of **Refresh Schedule** to open a picker with schedule intervals.



1. Set the schedule.

The picker scrolls and allows you to choose:

* + An interval: 1-30 minutes, 1-12 hours, 1 or 30 days, 1 or 2 weeks
  + A time. The time selector displays in the picker only when the interval is greater than 1 day and the day selection is greater than 1 week. When you schedule a specific time, Databricks SQL takes input in your computer’s timezone and converts it to UTC. If you want a query to run at a certain time in UTC, you must adjust the picker by your local offset. For example, if you want a query to execute at 00:00 UTC each day, but your current timezone is PDT (UTC-7), you should select 17:00 in the picker:



## Question 40

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?

* + **They can set up an Alert for the accompanying dashboard to notify them if the returned value is greater than 60.**
  + **They can set up an Alert for the query to notify when the ELT job fails.**
  + **They can set up an Alert for the accompanying dashboard to notify when it has not refreshed in 60 minutes.**
  + **They can set up an Alert for the query to notify them if the returned value is greater than 60.**
  + **This type of alert is not possible in Databricks**

**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.

## Question 41

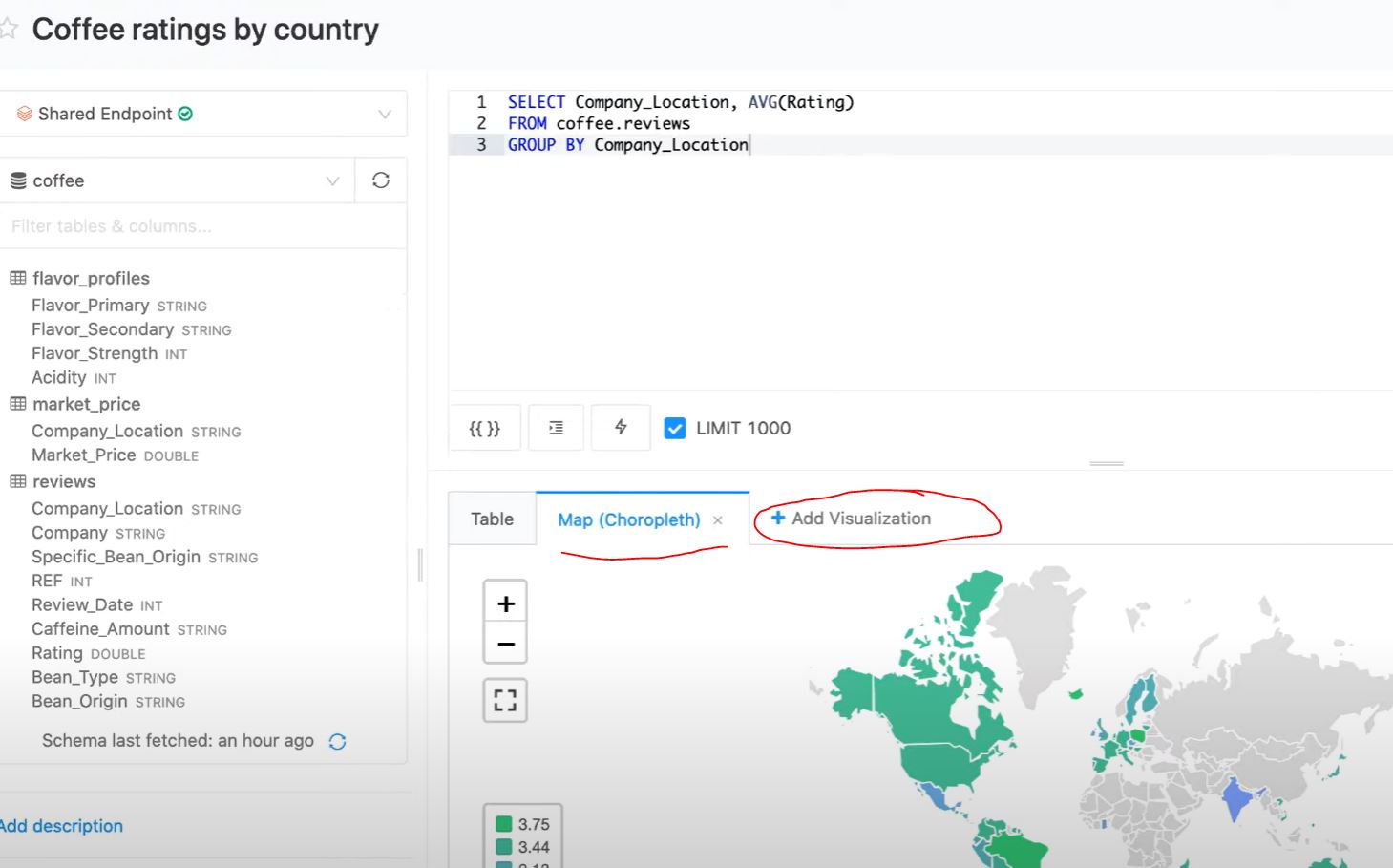
Which of the following is true, when building a Databricks SQL dashboard?

* **A dashboard can only use results from one query**
* **Only one visualization can be developed with one query result**
* **A dashboard can only connect to one schema/Database**
* **More than one visualization can be developed using a single query result**
* **A dashboard can only have one refresh schedule**

**Explanation**

the answer is, More than one visualization can be developed using a single query result.

In the query editor pane + Add visualization tab can be used for many visualizations for a single query result.



Bottom of Form

Top of Form

Question 42

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?

* **GRANT UPDATE ON TABLE table\_name TO john.smith@marketing.com**
* **GRANT USAGE ON TABLE table\_name TO john.smith@marketing.com**
* **GRANT MODIFY ON TABLE table\_name TO john.smith@marketing.com**
* **GRANT UPDATE TO TABLE table\_name ON john.smith@marketing.com**
* **GRANT MODIFY TO TABLE table\_name ON john.smith@marketing.com**

**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>

## Question 43

A new user who currently does not have access to the catalog or schema is requesting access to the customer table in sales schema, but the customer table contains sensitive information, so you have decided to create view on the table excluding columns that are sensitive and granted access to the view GRANT SELECT ON view\_name to user@company.com but when the user tries to query the view, gets the error view does not exist. What is the issue preventing user to access the view and how to fix it?

* + **User requires SELECT on the underlying table**
  + **User requires to be put in a special group that has access to PII data**
  + **User has to be the owner of the view**
  + **User requires USAGE privilege on Sales schema**
  + **User needs ADMIN privilege on the view**

**Explanation**

The answer is User requires USAGE privilege on Sales schema,

[Data object privileges - Azure Databricks | Microsoft Docs](https://docs.microsoft.com/en-us/azure/databricks/security/access-control/table-acls/object-privileges#privileges)

GRANT USAGE ON SCHEMA sales TO user@company.com;

* USAGE: does not give any abilities, but is an additional requirement to perform any action on a schema object.

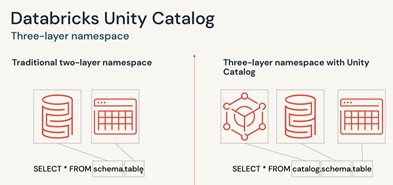
## Question 44

How do you access or use tables in the unity catalog?

* + **schema\_name.table\_name**
  + **schema\_name.catalog\_name.table\_name**
  + **catalog\_name.table\_name**
  + **catalog\_name.database\_name.schema\_name.table\_name**
  + **catalog\_name.schema\_name.table\_name**

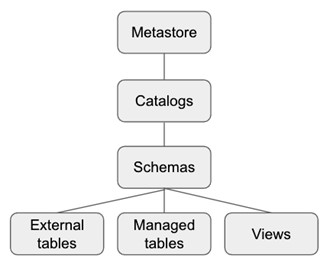
**Explanation**

The answer is**catalog\_name.schema\_name.table\_name**



**note: Database and Schema are analogous they are interchangeably used in the Unity catalog.**

FYI, A catalog is registered under a metastore, by default every workspace has a default metastore called hive\_metastore, with a unity catalog you have the ability to create meatstores and share that across multiple workspaces.



Bottom of Form

## Question 45

How do you upgrade an existing workspace managed table to a unity catalog table?

* + **ALTER TABLE table\_name SET UNITY\_CATALOG = TRUE**
  + **Create table catalog\_name.schema\_name.table\_name**

**as select \* from hive\_metastore.old\_schema.old\_table**

* + **Create table table\_name as select \* from hive\_metastore.old\_schema.old\_table**
  + **Create table table\_name format = UNITY as select \* from old\_table\_name**
  + **Create or replace table\_name format = UNITY using deep clone old\_table\_name**

**Explanation**

The answer is Create table catalog\_name.schema\_name.table\_name as select \* from hive\_metastore.old\_schema.old\_table

Basically, we are moving the data from an internal hive metastore to a metastore and catalog that is registered in the Unity catalog.

note: if it is a managed table the data is copied to a different storage account, for a large tables this can take a lot of time.  For an external table the process is different.

Managed table:[Upgrade a managed to Unity Catalog](https://docs.microsoft.com/en-us/azure/databricks/data-governance/unity-catalog/migrate#--upgrade-a-table-to-unity-catalog)

External table:  [Upgrade an external table to Unity Catalog](https://docs.microsoft.com/en-us/azure/databricks/data-governance/unity-catalog/migrate#--upgrade-an-external-table-to-unity-catalog)

Bottom of Form