**DATABRICKS ASSOCIATE EXAM**

Contents

[Question 1 6](#_Toc154291897)

[Question 2 7](#_Toc154291898)

[Question 3 8](#_Toc154291899)

[Question 4 9](#_Toc154291900)

[Question 5 10](#_Toc154291901)

[Question 6 11](#_Toc154291902)

[Question 7 12](#_Toc154291903)

[Question 8: 14](#_Toc154291904)

[Question 9 15](#_Toc154291905)

[Question 10: 16](#_Toc154291906)

[Question 11 17](#_Toc154291907)

[Question 12 18](#_Toc154291908)

[Question 13: 19](#_Toc154291909)

[Question 14: 20](#_Toc154291910)

[Question 15: 21](#_Toc154291911)

[Question 16: 22](#_Toc154291912)

[Question 17: 23](#_Toc154291913)

[Question 18: 24](#_Toc154291914)

[Question 20 25](#_Toc154291915)

[Question 20: 26](#_Toc154291916)

[Question 21: 27](#_Toc154291917)

[Question 22 28](#_Toc154291918)

[Question 23: 29](#_Toc154291919)

[Question 24: 30](#_Toc154291920)

[Question 25: 31](#_Toc154291921)

[Question 26: 32](#_Toc154291922)

[Question 27: 33](#_Toc154291923)

[Question 28: 34](#_Toc154291924)

[Question 29: 35](#_Toc154291925)

[Question 30: 36](#_Toc154291926)

[Question 31 37](#_Toc154291927)

[Question 32: 38](#_Toc154291928)

[Question 33: 39](#_Toc154291929)

[Question 34: 40](#_Toc154291930)

[Question 36: 43](#_Toc154291931)

[Question 37: 44](#_Toc154291932)

[Question 38: 45](#_Toc154291933)

[Question 39: 46](#_Toc154291934)

[Question 40: 47](#_Toc154291935)

[Question 42: 50](#_Toc154291936)

[Question 43: 51](#_Toc154291937)

[Question 44: 52](#_Toc154291938)

[Question 45: 53](#_Toc154291939)

[Question 46: 54](#_Toc154291940)

[Question 47: 55](#_Toc154291941)

[Question 48: 56](#_Toc154291942)

[Question 50 57](#_Toc154291943)

[Question 50: 58](#_Toc154291944)

[Question 51: 59](#_Toc154291945)

[Question 52: 60](#_Toc154291946)

[Question 53: 61](#_Toc154291947)

[Question 54: 62](#_Toc154291948)

[Question 55: 63](#_Toc154291949)

[Question 57 65](#_Toc154291950)

[Question 59: 67](#_Toc154291951)

[Question 60: 68](#_Toc154291952)

[Question 61 69](#_Toc154291953)

[Question 62 71](#_Toc154291954)

[Question 63 73](#_Toc154291955)

[Question 64: 74](#_Toc154291956)

[Question 65: 75](#_Toc154291957)

[Question 66 76](#_Toc154291958)

[Question 67 77](#_Toc154291959)

[Question 68 78](#_Toc154291960)

[Question 69 79](#_Toc154291961)

[Question 70 80](#_Toc154291962)

[Question 71 81](#_Toc154291963)

[Question 72 82](#_Toc154291964)

[Question 73: 83](#_Toc154291965)

[Question 74 84](#_Toc154291966)

[Question 75: 85](#_Toc154291967)

[Question 76 86](#_Toc154291968)

[Question 77 87](#_Toc154291969)

[Question 78 88](#_Toc154291970)

[Question 79: 89](#_Toc154291971)

[Question 80: 90](#_Toc154291972)

[Question 81 91](#_Toc154291973)

[Question 82 92](#_Toc154291974)

[Question 83: 93](#_Toc154291975)

[Question 84: 94](#_Toc154291976)

[Question 85: 95](#_Toc154291977)

[Question 86: 96](#_Toc154291978)

[Question 87: 97](#_Toc154291979)

[Question 88: 98](#_Toc154291980)

[Question 89: 99](#_Toc154291981)

[Question 90 100](#_Toc154291982)

[Question 91 101](#_Toc154291983)

[Question 92: 102](#_Toc154291984)

[Question 93: 103](#_Toc154291985)

[Question 94: 105](#_Toc154291986)

[Question 95: 106](#_Toc154291987)

[Question 96 107](#_Toc154291988)

[Question 97 109](#_Toc154291989)

[Question 98 110](#_Toc154291990)

[Question 99 111](#_Toc154291991)

[Question 100 113](#_Toc154291992)

[Question 101 114](#_Toc154291993)

[Question 102 115](#_Toc154291994)

[Question 103 116](#_Toc154291995)

[Question 104 117](#_Toc154291996)

[Question 105 118](#_Toc154291997)

[Question 106 119](#_Toc154291998)

[Question 107: 120](#_Toc154291999)

[Question 108 121](#_Toc154292000)

[Question 109 122](#_Toc154292001)

[Question 110 123](#_Toc154292002)

[Question 111 124](#_Toc154292003)

[Question 102 125](#_Toc154292004)

[Question 113 126](#_Toc154292005)

[Question 114 127](#_Toc154292006)

[Question 115 128](#_Toc154292007)

[Question 116 129](#_Toc154292008)

[Question 117 130](#_Toc154292009)

[Question 118 131](#_Toc154292010)

[Question 119: 132](#_Toc154292011)

[Question 120 133](#_Toc154292012)

## Question 1

A data engineer wants to pass multiple parameters from a Databricks Job to a notebook. They already configured the key and value of each parameter in the configurations of the job. Which of the following utilities can the data engineer use to read the passed parameters inside the notebook ?

* **dbutils.secrets**
* **dbutils.library**
* **dbutils.fs**
* **dbutils.notebook**
* **dbutils.widgets**

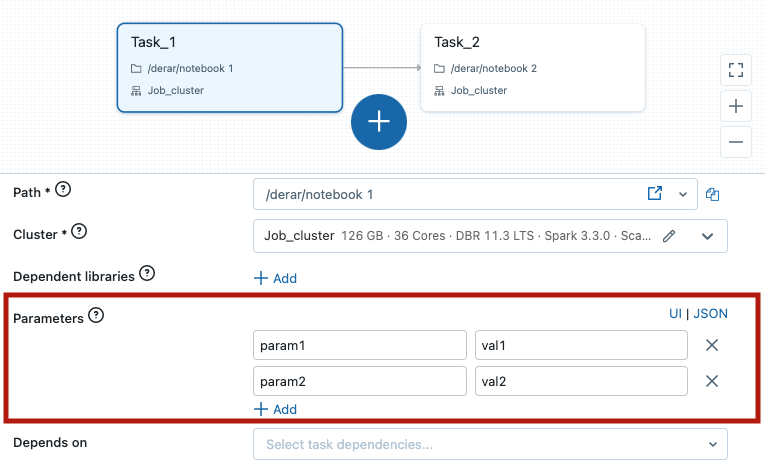
**Explanation**

dbutils.widgets allow you to add parameters to your notebooks.

Example: Adding a parameter named ‘param1’

1. ​​dbutils.widgets.text("param1", "default")
2. param1 = dbutils.widgets.get("param1")

The defined parameters can be passed from Databricks Jobs by adding them to the Parameters section in the task configuration.



Reference: <https://docs.databricks.com/notebooks/widgets.html>

## Question 2

Bottom of Form

Top of Form

Which of the following describes the minimal permissions a data engineer needs to view the metrics, driver logs, and Spark UI of an existing cluster ?

* **“Can Attach To” privilege on the cluster**
* **“Can Restart” privilege on the cluster**
* **“Can Manage” privilege on the cluster**
* **Cluster creation allowed + “Can Attach To” privileges on the cluster**
* **Cluster creation allowed + “Can Restart” privileges on the cluster**

**Explanation**

You can configure two types of cluster permissions:

1- The ‘Allow cluster creation’ entitlement controls your ability to create clusters.

2- Cluster-level permissions control your ability to use and modify a specific cluster. There are four permission levels for a cluster: No Permissions, Can Attach To, Can Restart, and Can Manage. The table lists the abilities for each permission:



Reference:

<https://docs.databricks.com/security/auth-authz/access-control/cluster-acl.html>

Bottom of Form

## Question 3

For production Databricks jobs, which of the following cluster types is recommended to use?

* **All-purpose clusters**
* **Production clusters**
* **Job clusters**
* **On-premises clusters**
* **Serverless clusters**

**Explanation**

Job Clusters are dedicated clusters for a job or task run. A job cluster auto terminates once the job is completed, which saves cost compared to all-purpose clusters.

In addition, Databricks recommends using job clusters in production so that each job runs in a fully isolated environment.

Reference: <https://docs.databricks.com/workflows/jobs/jobs.html#choose-the-correct-cluster-type-for-your-job>

## Question 4

The data engineering team has a Delta Lake table created with following query:

1. CREATE TABLE target
2. AS SELECT \* FROM source

A data engineer wants to drop the source table with the following query:

DROP TABLE source

Which statement describes the result of running this drop command ?

* **An error will occur indicating that other tables are based on this source table**
* **Both the target and source tables will be dropped**
* **No table will be dropped until CASCADE keyword is added to the command**
* **Only the source table will be dropped, but the target table will be no more queryable**
* **Only the source table will be dropped, while the target table will not be affected**

**Explanation**

CREATE TABLE AS SELECT statements, or CTAS statements create new Delta tables and populate them using the output of a SELECT query. So, when dropping the source table, the target table will not be affected.

Reference: (cf. AS query clause)

<https://docs.databricks.com/sql/language-manual/sql-ref-syntax-ddl-create-table-using.html>

## Question 5

Which of the following describes the minimal permissions a data engineer needs to start and terminate an existing cluster ?

* **“Can Attach To” privilege on the cluster**
* **“Can Restart” privilege on the cluster**
* **“Can Manage” privilege on the cluster**
* **Cluster creation allowed + “Can Attach To” privileges on the cluster**
* **Cluster creation allowed + “Can Restart” privileges on the cluster**

**Explanation**

You can configure two types of cluster permissions:

1- The ‘Allow cluster creation’ entitlement controls your ability to create clusters.

2- Cluster-level permissions control your ability to use and modify a specific cluster. There are four permission levels for a cluster: No Permissions, Can Attach To, Can Restart, and Can Manage. The table lists the abilities for each permission:



Reference: <https://docs.databricks.com/security/auth-authz/access-control/cluster-acl.html>

## Question 6

The data engineering team has a Delta Lake table created with following query:

CREATE TABLE customers\_clone

LOCATION 'dbfs:/mnt/backup'

AS SELECT \* FROM customers

A data engineer wants to drop the table with the following query:

DROP TABLE customers\_clone

Which statement describes the result of running this drop command ?

* **An error will occur as the table is deep cloned from the customers table**
* **An error will occur as the table is shallowly cloned from the customers table**
* **Only the table's metadata will be deleted from the catalog, while the data files will be kept in the storage**
* **Both the table's metadata and the data files will be deleted**
* **The table will not be dropped until VACUUM command is run**

**Explanation**

External (unmanaged) tables are tables whose data is stored in an external storage path by using a LOCATION clause.

When you run DROP TABLE on an external table, only the table's metadata is deleted, while the underlying data files are kept.

Reference: <https://docs.databricks.com/lakehouse/data-objects.html#what-is-an-unmanaged-table>

## Question 7

Which of the following describes the minimal permissions a data engineer needs to edit the configurations of an existing cluster ?

* **“Can Restart” privilege on the cluster**
* **“Can Manage” privilege on the cluster**
* **Cluster creation allowed + “Can Restart” privileges on the cluster**
* **Cluster creation allowed + “Can Manage” privileges on the cluster**
* **Only administrators can edit the configurations on existing clusters**

**Explanation**

You can configure two types of cluster permissions:

1- The ‘Allow cluster creation’ entitlement controls your ability to create clusters.

2- Cluster-level permissions control your ability to use and modify a specific cluster. There are four permission levels for a cluster: No Permissions, Can Attach To, Can Restart, and Can Manage. The table lists the abilities for each permission:



Reference:

<https://docs.databricks.com/security/auth-authz/access-control/cluster-acl.html>

## Question 8:

Given the following code block in a notebook

db\_password = dbutils.secrets.get(scope="dev", key="database\_password")

print (db\_password)

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

* **An interactive input box will appear in the notebook**
* **The string "REDACTED" will be printed.**
* **The error message “Secrets can not be printed” will be shown**
* **The string value of the password will be printed in plain text.**
* **If the user has “Can Read” permission, the string value of the password will be printed in plain text. Otherwise, the string "REDACTED" will be printed.**

**Explanation**

Databricks Secrets allows you to securely store your credentials and reference them in notebooks and jobs. To prevent accidentally printing a secret to standard output buffers or displaying the value during variable assignment, Databricks redacts secret values that are read using dbutils.secrets.get(). When displayed in notebook cell output, the secret values are replaced with [REDACTED] string.

Reference:

<https://docs.databricks.com/security/secrets/redaction.html>

<https://docs.databricks.com/security/secrets/index.html>

## Question 9

A junior data engineer is using the %sh magic command to run some legacy code. A senior data engineer has recommended refactoring the code instead.

Which of the following could explain why a data engineer may need to avoid using the %sh magic command ?

* **%sh restarts the Python interpreter. This clears all the variables declared in the notebook**
* **%sh executes shell code only on the local driver machine which leads to significant performance overhead.**
* **%sh can not access storage to persist the output**
* **All the above reasons explain why %sh may need to be avoided**
* **None of these reasons correctly describe why %sh may need to be avoided**

**Explanation**

Databricks support the %sh auxiliary magic command to run shell code in notebooks. This command runs only on the Apache Spark driver, and not on the worker nodes.

## Question 10:

Given a Delta table ‘products’ with the following schema:

name STRING, category STRING, expiration\_date DATE,  price FLOAT

When executing the below query:

1. SELECT \* FROM products
2. WHERE price > 30.5

Which of the following will be leaverged by the query optimizer to identify the data files to load?

* **Columns statistics in the Hive metastore**
* **Columns statistics in the metadata of Parquet files**
* **Files statistics in the Delta transaction log**
* **Files statistics in the in the Hive metastore**
* **None of the above. All data files are fully scanned to identify the ones to load**

**Explanation**

In the Transaction log, Delta Lake captures statistics for each data file of the table. These statistics indicate per file:

* Total number of records
* Minimum value in each column of the first 32 columns of the table
* Maximum value in each column of the first 32 columns of the table
* Null value counts for in each column of the first 32 columns of the table

When a query with a selective filter is executed against the table, the query optimizer uses these statistics to generate the query result. it leverages them to identify data files that may contain records matching the conditional filter.

For the SELECT query in the question, The transaction log is scanned for min and max statistics for the price column.

Reference: <https://docs.databricks.com/delta/data-skipping.html>

## Question 11

The data engineering team has a table ‘orders\_backup’ that was created using Delta Lake’s SHALLOW CLONE functionality from the table ‘orders’. Recently, the team started getting an error when querying the ‘orders\_backup’ table indicating that some data files are no longer present.

Which of the following correctly explains this error ?

* **The VACUUM command was run on the orders table**
* **The VACUUM command was run on the orders\_backup table**
* **The OPTIMIZE command was run on the orders table**
* **The OPTIMIZE command was run on the orders\_backup table**
* **The REFRESH command was run on the orders\_backup table**

**Explanation**

With Shallow Clone, you create a copy of a table by just copying the Delta transaction logs.

That means that there is no data moving during Shallow Cloning.

Running the VACUUM command on the source table may purge data files referenced in the transaction log of the clone. In this case, you will get an error when querying the clone indicating that some data files are no longer present.

Reference:

<https://docs.databricks.com/delta/clone.html>

## Question 12

A data engineer has a Delta Lake table named ‘orders\_archive’ created using the following command:

CREATE TABLE orders\_archive

DEEP CLONE orders

They want to sync up the new changes in the orders table to the clone.

Which of the following commands can be run to achieve this task ?

* **REFRESH orders\_archive**
* **SYNC orders\_archive**
* **INSERT OVERWRITE orders\_archive**

**SELECT \* FROM orders**

* **CREATE OR REPLACE TABLE orders\_archive**

**DEEP CLONE orders**

* **DROP TABLE orders\_archive;**

**CREATE TABLE orders\_archive**

**DEEP CLONE orders**

**Explanation**

Cloning can occur incrementally. Executing the CREATE OR REPLACE TABLE command can sync changes from the source to the target location.

Now, If you run DESCRIBE HISTORY orders\_archive, you will see a new version of CLONE operation occurred on the table.

\* The last choice in the question is incorrect since dropping the table will lead to removing all the table's history.

Reference:

<https://docs.databricks.com/delta/clone.html>

## Question 13:

The data engineering team has a Delta Lake table named ‘daily\_activities’ that is completely overwritten each night with new data received from the source system.

For auditing purposes, the team wants to set up a post-processing task that uses Delta Lake Time Travel functionality to determine the difference between the new version and the previous version of the table. They start by getting the current table version via this code:

current\_version = spark.sql("SELECT max(version) FROM (DESCRIBE HISTORY daily\_activities)").collect()[0][0]

Which of the following queries can be used by the team to complete this task ?

* **SELECT \* FROM daily\_activities**

**UNION**

**SELECT \* FROM daily\_activities AS VERSION = {current\_version-1}**

* **SELECT \* FROM daily\_activities**

**UNION ALL**

**SELECT \* FROM daily\_activities@v{current\_version-1}**

* **SELECT \* FROM daily\_activities**

**INTERSECT**

**SELECT \* FROM daily\_activities AS VERSION = {current\_version-1}**

* **SELECT \* FROM daily\_activities**

**EXCEPT**

**SELECT \* FROM daily\_activities@v{current\_version-1}**

* **SELECT \* FROM daily\_activities**

**MINUS**

**SELECT \* FROM daily\_activities AS VERSION = {current\_version-1}**

**Explanation**

Each operation that modifies a Delta Lake table creates a new table version. You can use history information to audit operations or query a table at a specific point in time using:

* Version number

SELECT \* FROM my\_table@v36

SELECT \* FROM my\_table VERSION AS OF 36

* Timestamp

SELECT \* FROM my\_table TIMESTAMP AS OF "2019-01-01"

Using the EXCEPT set operator, you can get the difference between the new version and the previous version of the table

Reference:

<https://docs.databricks.com/delta/history.html>

<https://docs.databricks.com/sql/language-manual/sql-ref-syntax-qry-select-setops.html>

## Question 14:

The data engineering team wants to build a pipeline that receives customers data as change data capture (CDC) feed from a source system. The CDC events logged at the source contain the data of the records along with metadata information. This metadata indicates whether the specified record was inserted, updated, or deleted. In addition to a timestamp column identified by the field update\_time indicating the order in which the changes happened. Each record has a primary key identified by the field customer\_id.

In the same batch, multiple changes for the same customer could be received with different update\_time. The team wants to store only the most recent information for each customer in the target Delta Lake table.

Which of the following solutions meets these requirements?

* **Enable Delta Lake's Change Data Feed (CDF) on the target table to automatically merge the received CDC feed**
* **Use MERGE INTO to upsert the most recent entry for each customer\_id into the table**
* **Use MERGE INTO with SEQUENCE BY clause on the update\_time for ordering how operations should be applied**
* **Use dropDuplicates function to remove duplicates by customer\_id, then merge the duplicate records into the table.**
* **Use the option mergeSchema when writing the CDC data into the table to automatically merge the changed data with its most recent schema.**

**Explanation**

MERGE INTO command allows you to upsert data from a source table, view, or DataFrame into a target Delta table. Delta Lake supports inserts, updates, and deletes in merge operations.

Reference:

<https://docs.databricks.com/delta/merge.html>

<https://docs.databricks.com/sql/language-manual/delta-merge-into.html>

## Question 15:

A data engineer is using a foreachBatch logic to upsert data in a target Delta table.

The function to be called at each new microbatch processing is displayed below with a blank:

def upsert\_data(microBatchDF, batch\_id):

microBatchDF.createOrReplaceTempView("sales\_microbatch")

sql\_query = """

MERGE INTO sales\_silver a

USING sales\_microbatch b

ON a.item\_id=b.item\_id

AND a.item\_timestamp=b.item\_timestamp

WHEN NOT MATCHED THEN INSERT \*

"""

Which option correctly fills in the blank to execute the sql query in the function on a cluster with Databricks Runtime below 10.5 ?

* + **spark.sql(sql\_query)**
  + **batch\_id.sql(sql\_query)**
  + **microBatchDF.sql(sql\_query)**
  + **microBatchDF.sparkSession.sql(sql\_query)**
  + **microBatchDF.\_jdf.sparkSession().sql(sql\_query)**

**Explanation**

Usually, we use spark.sq() function to run SQL queries. However, in this particular case, the spark session can not be accessed from within the microbatch process. Instead, we can access the local spark session from the microbatch dataframe.

For clusters with Databricks Runtime version below 10.5, the syntax to access the local spark session is:

microBatchDF.\_jdf.sparkSession().sql(sql\_query)

Reference: <https://docs.gcp.databricks.com/structured-streaming/delta-lake.html#language-python>

## Question 16:

The data engineering team has a singleplex bronze table called ‘orders\_raw’ where new orders data is appended every night. They created a new Silver table called ‘orders\_cleaned’ in order to provide a more refined view of the orders data.

The team wants to create a batch processing pipeline to process all new records inserted in the orders\_raw table and propagate them to the orders\_cleaned table.

Which solution minimizes the compute costs to propagate this batch of data?

* **Use time travel capabilities in Delta Lake to compare the latest version of orders\_raw with one version prior, then write the difference to the orders\_cleaned table.**
* **Use Spark Structured Streaming to process the new records from orders\_raw in batch mode using the trigger availableNow option**
* **Use Spark Structured Streaming's foreachBatch logic to process the new records from orders\_raw using trigger(processingTime=”24 hours")**
* **Use batch overwrite logic to reprocess all records in orders\_raw and overwrite the orders\_cleaned table**
* **Use insert-only merge into the orders\_cleaned table using orders\_raw data based on a composite key**

**Explanation**

Databricks supports trigger(availableNow=True) for Delta Lake and Auto Loader sources. This functionality consumes all available records in an incremental batch.

There is also the trigger(once=True) option for incremental batch processing. However, this setting is now deprecated in the newer Databricks Runtime versions.

NOTE: You may still see this option in the current certification exam version. However, Databricks recommends you use trigger(availableNow=True) for all future incremental batch processing workloads.

Reference:

<https://docs.databricks.com/structured-streaming/triggers.html#configuring-incremental-batch-processing>

## Question 17:

The data engineering team has a Silver table called ‘sales\_cleaned’ where new sales data is appended in near real-time.

They want to create a new Gold-layer entity against the ‘sales\_cleaned’ table to calculate the year-to-date (YTD) of the sales amount. The new entity will have the following schema:

**country\_code STRING, category STRING, ytd\_total\_sales FLOAT, updated TIMESTAMP**

It’s enough for these metrics to be recalculated once daily. But since they will be queried very frequently by several business teams, the data engineering team wants to cut down the potential costs and latency associated with materializing the results.

Which of the following solutions meets these requirements?

* **Define the new entity as a view to avoid persisting the results each time the metrics are recalculated**
* **Define the new entity as a global temporary view since it can be shared between notebooks or jobs that share computing resources.**
* **Configuring a nightly batch job to recalculate the metrics and store them as a table overwritten with each update.**
* **Create multiple tables, one per business team so the metrics can be queried quickly and efficiently.**
* **All the above solutions meet the required requirements since Databricks uses the Delta Caching feature**

**Explanation**

Data engineers must understand how materializing results is different between views and tables on Databricks, and how to reduce total compute and storage cost associated with each materialization depending on the scenario.

Consider using a view when:

* Your query is not complex. Because views are computed on demand, the view is re-computed every time the view is queried. So, frequently querying complex queries with joins and subqueries increases compute costs
* You want to reduce storage costs. Views do not require additional storage resources.

Consider using a gold table when:

* Multiple downstream queries consume the table, so you want to avoid re-computing complex ad-hoc queries every time.
* Query results should be computed incrementally from a data source that is continuously or incrementally growing.

## Question 18:

A data engineer wants to calculate predictions using a MLFlow model logged in a given “model\_url”. They want to register the model as a Spark UDF in order to apply it to a test dataset.

Which code block allows the data engineer to register the MLFlow model as a Spark UDF ?

* **predict\_udf = mlflow.pyfunc.spark\_udf(spark, "model\_url")**
* **predict\_udf = mlflow.spark\_udf(spark, "model\_url")**
* **predict\_udf = mlflow.udf(spark, "model\_url")**
* **predict\_udf = pyfunc.spark\_udf(spark, "model\_url")**
* **predict\_udf = mlflow.pyfunc(spark, "model\_url")**

**Explanation**

Mlflow.pyfunc.spark\_udf function allows to register a MLFlow model as a Apache Spark UDF. It needs at least 2 parameters:

* spark: A SparkSession object
* model\_uri: the location, in URI format, of the MLflow model

Once the Spark UDF created, it can be applied to a dataset to calculate the predictions:

1. predict\_udf = mlflow.pyfunc.spark\_udf(spark, "model\_url")
2. pred\_df = data\_df.withColumn("prediction", predict\_udf(\*column\_list))

Sample Notebook:

<https://docs.databricks.com/en/_extras/notebooks/source/mlflow/mlflow-quick-start-inference.html>

Reference:

<https://docs.databricks.com/en/mlflow/models.html>

<https://mlflow.org/docs/latest/python_api/mlflow.pyfunc.html#mlflow.pyfunc.spark_udf>

## Question 20

“A Delta Lake’s functionality that automatically compacts small files during individual writes to a table by performing two complementary operations on the table”

Which of the following is being described in the above statement?

* **Optimized writes**
* **Auto compaction**
* **Auto Optimize**
* **OPTIMIZE command**
* **REORG TABLE command**

**Explanation**

Auto Optimize is a functionality that allows Delta Lake to automatically compact small data files of Delta tables. This can be achieved during individual writes to the Delta table.

Auto optimize consists of 2 complementary operations:

- Optimized writes: with this feature enabled, Databricks attempts to write out 128 MB files for each table partition.

- Auto compaction: this will check after an individual write, if files can further be compacted. If yes, it runs an OPTIMIZE job with 128 MB file sizes (instead of the 1 GB file size used in the standard OPTIMIZE)

Reference:

<https://docs.databricks.com/delta/tune-file-size.html>

## Question 20:

The data engineering team has a large external Delta table where new changes are merged very frequently. They enabled Optimized writes and Auto Compaction on the table in order to automatically compact small data files to target files of size 128 MB. However, when they look at the table directory, they see that most data files are smaller than 128 MB.

Which of the following likely explains these smaller file sizes ?

* **Optimized Writes and Auto Compaction have no effect on large Delta tables. The table needs to be partitioned so Auto Compaction can be applied at partition level.**
* **Optimized Writes and Auto Compaction have no effect on external tables. The table needs to be managed in order to store the information of file sizes in the Hive metastore.**
* **Optimized Writes and Auto Compaction automatically generate smaller data files to reduce the duration of future MERGE operations.**
* **Auto compaction supports Auto Z-Ordering which is more expensive than just compaction**
* **The team needs to look at the table’s \_auto\_optimize directory, where all new compacted files are written.**

**Explanation**

Having many small files can help minimize rewrites during some operations like merges and deletes. For such operations, Databricks can automatically tune the file size of Delta tables. As a result, it can generate data files smaller than the default 128MB. This helps in reducing the duration of future MERGE operations.

Reference:

<https://docs.databricks.com/delta/tune-file-size.html#autotune-file-size-based-on-workload>

## Question 21:

Which statement regarding streaming state in Stream-Stream Joins is correct?

* **Stream-Stream Joins are not stateful. Spark does not buffers past inputs as a streaming state for the input streams**
* **Spark buffers past inputs as a streaming state only for the left input stream, so that it can match future right inputs with past left inputs.**
* **Spark buffers past inputs as a streaming state only for the right input stream, so that it can match future left inputs with past right inputs.**
* **Spark buffers past inputs as a streaming state for both input streams, so that it can match every future input with past inputs.**
* **Stream-Stream Joins does not support limiting the state information using watermarks.**

**Explanation**

When performing stream-stream join, Spark buffers past inputs as a streaming state for both input streams, so that it can match every future input with past inputs. This state can be limited by using watermarks.

Reference:

<https://www.databricks.com/blog/2018/03/13/introducing-stream-stream-joins-in-apache-spark-2-3.html>

## Question 22

Which statement regarding static Delta tables in Stream-Static joins is correct?

* **Static Delta tables must be small enough to be broadcasted to all worker nodes in the cluster.**
* **Static Delta tables need to be partitioned in order to be used in stream-static join.**
* **Static Delta tables need to be refreshed with REFRESH TABLE command for each microbatch of a stream-static join**
* **The latest version of the static Delta table is returned each time it is queried by a microbatch of the stream-static join**
* **The latest version of the static Delta table is returned only for the first microbatch of the stream-static join. Then, it will be cached to be used by any upcoming microbatch.**

**Explanation**

Stream-static joins take advantage of Delta Lake guarantee that the latest version of the static delta table is returned each time it is queried in a join operation with a data stream.

Reference:

<https://docs.databricks.com/structured-streaming/delta-lake.html#performing-stream-static-joins>

## Question 23:

A data engineer has the following streaming query with a blank:

***spark.readStream***

***.table("orders\_cleaned")***

***.groupBy(***

***"order\_timestamp",***

***"author")***

***.agg(***

***count("order\_id").alias("orders\_count"),***

***avg("quantity").alias("avg\_quantity"))***

***.writeStream***

***.option("checkpointLocation", "dbfs:/path/checkpoint")***

***.table("orders\_stats")***

For handling late-arriving data, they want to maintain the streaming state information for 30 minutes.

Which option correctly fills in the blank to meet this requirement ?

* **.trigger(processingTime=”30 minutes")**
* **.awaitTermination("order\_timestamp", "30 minutes")**
* **.awaitWatermark("order\_timestamp", "30 minutes")**
* **.withWatermark("order\_timestamp", "30 minutes")**
* **.window("order\_timestamp", "30 minutes")**

**Explanation**

pyspark.sql.DataFrame.withWatermark function allows you to only track state information for a window of time in which we expect records could be delayed.

Reference:

<https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.sql.DataFrame.withWatermark.html>

## Question 24:

Given the following streaming query:

spark.readStream

.table("orders\_cleaned")

.withWatermark("order\_timestamp", "10 minutes")

.groupBy(

window("order\_timestamp", "5 minutes").alias("time"),

"author")

.agg(

count("order\_id").alias("orders\_count"),

avg("quantity").alias("avg\_quantity"))

.writeStream

.option("checkpointLocation", "dbfs:/path/checkpoint")

.table("orders\_stats")

Which of the following statements best describe this query ?

* **It calculates business-level aggregates for each non-overlapping ten-minute interval. Incremental state information is maintained for 5 minutes for late-arriving data.**
* **It calculates business-level aggregates for each non-overlapping five-minute interval. Incremental state information is maintained for 10 minutes for late-arriving data.**
* **It calculates business-level aggregates for each overlapping five-minute interval. Incremental state information is maintained for 10 minutes for late-arriving data.**
* **It calculates business-level aggregates for each overlapping ten-minute interval. Incremental state information is maintained for 5 minutes for late-arriving data.**
* **None of the above statements correctly describe this query**

**Explanation**

Pyspark.sql.functions.window function bucketizes rows into one or more time windows given a timestamp specifying column. In this query, we are performing aggregations per order\_timestamp for each non-overlapping five minute interval.

pyspark.sql.DataFrame.withWatermark function allows you to only track state information for a window of time in which we expect records could be delayed. Here we define a watermark of 10 minutes.

Reference:

<https://spark.apache.org/docs/3.1.3/api/python/reference/api/pyspark.sql.functions.window.html>

<https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.sql.DataFrame.withWatermark.html>

## Question 25:

Which statement regarding checkpointing in Spark Structured Streaming is **Not** correct?

* **Checkpoints stores the current state of a streaming job to cloud storage**
* **Checkpointing allows the streaming engine to track the progress of a stream processing**
* **Checkpoints can be shared between separate streams**
* **To specify the checkpoint in a streaming query, we use the checkpointLocation option.**
* **Checkpointing with write-ahead logs mechanism ensure fault-tolerant stream processing**

**Explanation**

Checkpoints cannot be shared between separate streams. Each stream needs to have its own checkpoint directory to ensure processing guarantees.

Reference:

<https://www.databricks.com/blog/2022/12/12/streaming-production-collected-best-practices.html>

## Question 26:

Which of the following statements best describes Delta Lake Auto Compaction?

* **Auto Compaction occurs after a write to a table has succeeded to check if files can further be compacted; if yes, it runs an OPTIMIZE job with Z-Ordering toward a file size of 128 MB.**
* **Auto Compaction occurs after a write to a table has succeeded to check if files can further be compacted; if yes, it runs an OPTIMIZE job without Z-Ordering toward a file size of 128 MB.**
* **Auto Compaction occurs after a write to a table has succeeded to check if files can further be compacted; if yes, it runs an OPTIMIZE job with Z-Ordering toward a file size of 1 GB.**
* **Auto Compaction occurs after a write to a table has succeeded to check if files can further be compacted; if yes, it runs an OPTIMIZE job without Z-Ordering toward a file size of 1 GB.**
* **None of the above statements correctly describe Delta Lake Auto Compaction.**

**Explanation**

Auto Compaction is part of the Auto Optimize feature in Databricks. it checks after an individual write, if files can further be compacted, if yes, it runs an OPTIMIZE job with 128 MB file sizes instead of the 1 GB file size used in the standard OPTIMIZE

Auto compaction does not support Z-Ordering as Z-Ordering is significantly more expensive than just compaction.

Reference:

<https://docs.databricks.com/delta/tune-file-size.html#auto-compaction-for-delta-lake-on-databricks>

## Question 27:

Which of the following statements best describes Auto Loader ?

* **Auto loader allows applying Change Data Capture (CDC) feed to update tables based on changes captured in source data.**
* **Auto loader monitors a source location, in which files accumulate, to identify and ingest only new arriving files with each command run. While the files that have already been ingested in previous runs are skipped.**
* **Auto loader allows cloning a source Delta table to a target destination at a specific version.**
* **Auto loader defines data quality expectations on the contents of a dataset, and reports the records that violate these expectations in metrics.**
* **Auto loader enables efficient insert, update, deletes, and rollback capabilities by adding a storage layer that provides better data reliability to data lakes.**

**Explanation**

Auto Loader incrementally and idempotently processes new data files as they arrive in cloud storage and load them into a target Delta Lake table.

Reference: <https://docs.databricks.com/ingestion/auto-loader/index.html>

## Question 28:

Which of the following functions can a data engineer use to return a new DataFrame containing the distinct rows from a given DataFrame based on multiple columns?

* **pyspark.sql.DataFrame.drop**
* **pyspark.sql.DataFrame.distinct**
* **pyspark.sql.DataFrame.dropDuplicates**
* **pyspark.sql.DataFrame.na.drop**
* **pyspark.sql.DataFrame.dropna**

**Explanation**

pyspark.sql.DataFrame.dropDuplicates function returns a new DataFrame with duplicate rows removed, optionally only considering certain columns.

Reference:

<https://spark.apache.org/docs/3.1.2/api/python/reference/api/pyspark.sql.DataFrame.dropDuplicates.html>

## Question 29:

Which of the following approaches allows to correctly perform streaming deduplication ?

* **De-duplicate records within each batch, and then append the result into the target table**
* **De-duplicate records within each batch, and then merge the result into the target table using insert-only merge**
* **De-duplicate records within each batch, rank the result, and then insert only records having rank = 1 into the target table**
* **De-duplicate records in all batches with watermarking, and then overwrite the target table by the result**
* **None of the above approaches allows to correctly perform streaming deduplication**

**Explanation**

To perform streaming deduplication, we use dropDuplicates() function to eliminate duplicate records within each new micro batch. In addition, we need to ensure that records to be inserted are not already in the target table. We can achieve this using insert-only merge.

Reference:

<https://spark.apache.org/docs/3.1.2/api/python/reference/api/pyspark.sql.DataFrame.dropDuplicates.html>

<https://docs.databricks.com/delta/merge.html#data-deduplication-when-writing-into-delta-tables>

## Question 30:

A junior data engineer is testing the following code block to get the newest entry for each item added in the ‘sales’ table since the last table update.

from pyspark.sql import functions as F

from pyspark.sql.window import Window

window = Window.partitionBy("item\_id").orderBy(F.col("item\_time").desc())

ranked\_df = (spark.readStream

.table("sales")

.withColumn("rank", F.rank().over(window))

.filter("rank == 1")

.drop("rank")

)

display(ranked\_df)

However, the command fails when executed.

Which statement explains the cause of this failure?

* **The query output can not be displayed. They should use spark.writeStream to persist the query result.**
* **Watermarking is missing. It should be added to allow tracking state information for the window of time.**
* **Non-time-based window operations are not supported on streaming DataFrames. They need to be implemented inside a foreachBatch logic instead.**
* **The item\_id field is not unique. Records must be de-duplicated on the item\_id using dropDuplicates function**
* **The item\_id field is not unique. The drop("rank") must be called before applying the rank function in order to drop any duplicate record.**

**Explanation**

If you try to call such a window operation on a streaming DataFrames, this will generate an error indicating that “Non-time-based window operations are not supported on streaming DataFrames”.

Instead, these window operations need to be implemented inside a foreachBatch logic.

## Question 31

Given the following query on the Delta table ‘customers’ on which Change Data Feed is enabled:

spark.readStream

.option("readChangeFeed", "true")

.option("startingVersion", 0)

.table ("customers")

.filter (col("\_change\_type").isin(["update\_postimage"]))

.writeStream

.option ("checkpointLocation", "dbfs:/checkpoints")

.trigger (availableNow=True)

.table("customers\_updates")

Which statement describes the results of this query each time it is executed ?

* **Newly updated records will be merged into the target table, modifying previous entries with the same primary keys.**
* **Newly updated records will be appended to the target table.**
* **Newly updated records will overwrite the target table.**
* **The entire history of updated records will be appended to the target table at each execution, which leads to duplicate entries.**
* **The entire history of updated records will overwrite the target table at each execution.**

**Explanation**

Databricks supports reading table’s changes captured by CDF in streaming queries using spark.readStream. This allows you to get only the new changes captured since the last time the streaming query was run.

The query in the question then appends the data to the target table at each execution since it’s using the default writing mode, which is ‘append’.

Reference:

<https://docs.databricks.com/delta/delta-change-data-feed.html#read-changes-in-streaming-queries>

## Question 32:

Given the following query on the Delta table customers on which Change Data Feed is enabled:

spark.read

.option("readChangeFeed", "true")

.option("startingVersion", 0)

.table ("customers")

.filter(col("\_change\_type").isin(["update\_postimage"]))

.write

.mode(“overwrite”)

.table("customers\_updates")

Which statement describes the results of this query each time it is executed ?

* **Newly updated records will be merged into the target table, modifying previous entries with the same primary keys.**
* **Newly updated records will be appended to the target table.**
* **Newly updated records will overwrite the target table.**
* **The entire history of updated records will be appended to the target table at each execution, which leads to duplicate entries.**
* **The entire history of updated records will overwrite the target table at each execution.**

**Explanation**

Reading table’s changes, captured by CDF, using spark.read means that you are reading them as a static source. So, each time you run the query, all table’s changes (starting from the specified startingVersion) will be read.

The query in the question then writes the data in mode “overwrite” to the target table, which completely overwrites the table at each execution.

Reference:

<https://docs.databricks.com/delta/delta-change-data-feed.html#read-changes-in-batch-queries>

## Question 33:

The data engineering team maintains a Type 1 table that is overwritten each night with new data received from the source system.

A junior data engineer has suggested enabling the Change Data Feed (CDF) feature on the table in order to identify those rows that were updated, inserted, or deleted.

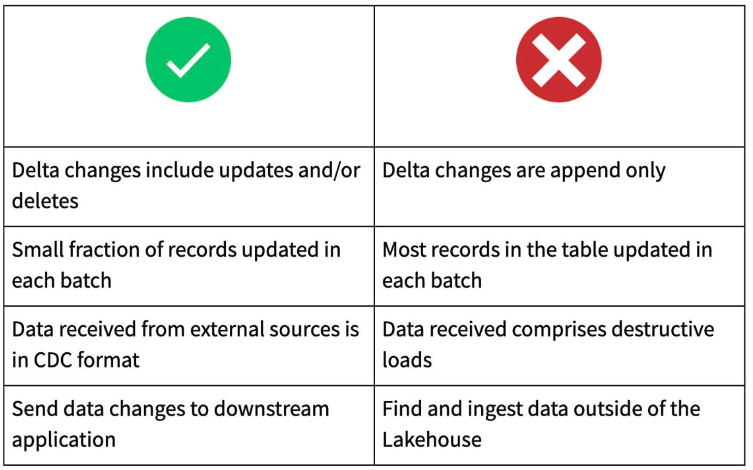
Which response to the junior data engineer's suggestion is correct?

* **CDF can not be enabled on existing tables. It can only be enabled on newly created tables.**
* **Table’s data changes captured by CDF can only be read in streaming mode**
* **CDF is useful when only a small fraction of records are updated in each batch**
* **CDF is useful when the table is a Slowly Changing Dimension (SCD) of Type 2**
* **All the above are correct responses to the data engineer's suggestion**

**Explanation**

Generally speaking, we use CDF for sending incremental data changes to downstream tables in a multi-hop architecture. So, use CDF when only small fraction of records updated in each batch. Such updates are usually received from external sources in CDC format. If most of the records in the table are updated, or if the table is overwritten in each batch, like in the question, don’t use CDF.

Here is some guidance for when to use CDF (from the below reference link)



Reference:

<https://www.databricks.com/blog/2021/06/09/how-to-simplify-cdc-with-delta-lakes-change-data-feed.html>

## Question 34:

A data engineer wants to ingest input json data into a target Delta table. They want the data ingestion to happen incrementally in near real-time.

Which option correctly meets the specified requirement ?

* **spark.readStream**

**.format("autoloader")**

**.option ("autoloader.format", "json")**

**.load(source\_path)**

**.writeStream**

**.option("checkpointLocation", checkpointPath)**

**.start("target\_table")**

* **spark.readStream**

**.format("autoloader")**

**.option ("autoloader.format", "json")**

**.load(source\_path)**

**.writeStream**

**.option("checkpointLocation", checkpointPath)**

**.trigger(real-time=True)**

**.start("target\_table")**

* **spark.readStream**

**.format("cloudFiles")**

**.option ("cloudFiles.format", "json")**

**.load(source\_path)**

**.writeStream**

**.option("checkpointLocation", checkpointPath)**

**.start("target\_table")**

* **spark.readStream**

**.format("cloudFiles")**

**.option ("cloudFiles.format", "json")**

**.load(source\_path)**

**.writeStream**

**.trigger(real-time=True)**

**.start("target\_table")**

* **spark.readStream**

**.format("cloudFiles")**

**.option ("cloudFiles.format", "json")**

**.load(source\_path)**

**.writeStream**

**.trigger(availableNow=True)**

**.start("target\_table")**

**Explanation**

In order to ingest input json data into a target Delta table, we use Autoloader. Auto Loader is based on Spark Structured Streaming and provides a Structured Streaming source called ‘cloudFiles’.

If you want the data ingestion to happen incrementally in near real-time, you can use the default trigger method which is trigger(processingTime=”500ms"). This allows the processing of data in micro-batches at a fixed interval of half a second.

Reference:

<https://docs.databricks.com/ingestion/auto-loader/index.html>

<https://docs.databricks.com/structured-streaming/triggers.html#what-is-the-default-trigger-interval>

Bottom of Form

Top of Form

Question 35:

Given the following Structured Streaming query:

(spark.table("orders")

.withColumn("total\_after\_tax", col("total")+col("tax"))

.writeStream

.option("checkpointLocation", checkpointPath)

.outputMode("append")

.\_\_\_\_\_\_\_\_\_\_\_\_\_

.table("new\_orders")

)

Fill in the blank to make the query executes a micro-batch to process data every 2 minutes

* **trigger(once=”2 minutes”)**
* **trigger(processingTime=”2 minutes")**
* **processingTime(”2 minutes")**
* **trigger(”2 minutes")**
* **trigger()**

**Explanation**

In Spark Structured Streaming, in order to process data in micro-batches at a user-specified intervals, you can use the processingTime trigger method. This allows you to specify a time duration as a string. By default, it’s “500ms”.

Reference: <https://docs.databricks.com/structured-streaming/triggers.html#configure-structured-streaming-trigger-intervals>

## Question 36:

Which statement regarding Delta Lake File Statistics is correct?

* **By default, Delta Lake captures statistics in the Hive metastore on the first 16 columns of each table.**
* **By default, Delta Lake captures statistics in the Hive metastore on the first 32 columns of each table.**
* **By default, Delta Lake captures statistics in the transaction log on the first 16 columns of each table.**
* **By default, Delta Lake captures statistics in the transaction log on the first 32 columns of each table.**
* **By default, Delta Lake captures statistics in both Hive metastore and transaction log for each added data file.**

**Explanation**

Delta Lake automatically captures statistics in the transaction log for each added data file of the table. By default, Delta Lake collects the statistics on the first 32 columns of each table. These statistics indicate per file:

* Total number of records
* Minimum value in each column of the first 32 columns of the table
* Maximum value in each column of the first 32 columns of the table
* Null value counts for in each column of the first 32 columns of the table

These statistics are leveraged for data skipping based on query filters.

Reference: <https://docs.databricks.com/delta/data-skipping.html>

## Question 37:

A data engineer uses the following SQL query:

GRANT USAGE ON DATABASE sales\_db TO finance\_team

Which of the following is the benefit of the USAGE privilege ?

* **Gives read access on the database**
* **Gives full permissions on the entire database**
* **Gives the ability to view database objects and their metadata**
* **No effect! but it's required to perform any action on the database**
* **USAGE privilege is not part of the Databricks governance model**

**Explanation**

The USAGE does not give any abilities, but it's an additional requirement to perform any action on a schema (database) object.

Reference: <https://docs.databricks.com/security/access-control/table-acls/object-privileges.html#privileges>

## Question 38:

The data engineering team is using the LOCATION keyword for every new Delta Lake table created in the Lakehouse.

Which of the following describes the purpose of using the LOCATION keyword in this case ?

* **The LOCATION keyword is used to configure the created Delta Lake tables as managed tables.**
* **The LOCATION keyword is used to configure the created Delta Lake tables as external tables.**
* **The LOCATION keyword is used to define the created Delta Lake tables in an external database.**
* **The LOCATION keyword is used to define the created Delta Lake tables in a database over a JDBC connection.**
* **The LOCATION keyword is used to set a default schema and checkpoint location for the created Delta Lake tables.**

**Explanation**

External (unmanaged) tables are tables whose data is stored in an external storage path by using a LOCATION clause.

Reference: <https://docs.databricks.com/lakehouse/data-objects.html#what-is-an-unmanaged-table>

## Question 39:

A data engineer wants to create a Delta Lake table for storing user activities of a website. The table has the following schema:

**user\_id LONG, page STRING, activity\_type LONG, ip\_address STRING, activity\_time TIMESTAMP, activity\_date DATE**

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

* **user\_id**
* **activity\_type**
* **page**
* **activity\_time**
* **activity\_date**

**Explanation**

When choosing partitioning columns, it's good to consider the fact that records with a given value (the activities of a given user) will continue to arrive indefinitely. In such a case, we use a datetime column for partitioning. This allows your partitions to be optimized, and allows you to easily archive those partitions of previous time periods, if necessary.

## Question 40:

The data engineering team has a large Delta table named ‘users’. A recent query on the table returned some entries with negative values in the ‘age’ column.

To avoid this issue and enforce data quality, a junior data engineer decided to add a CHECK constraint to the table with the following command:

ALTER TABLE users ADD CONSTRAINT valid\_age CHECK (age> 0);

However, the command fails when executed.

Which statement explains the cause of this failure?

* **The syntax for adding the CHECK constraint is incorrect. Instead, the command should be:**

**ALTER TABLE users ADD CONSTRAINT ON COLUMN age (CHECK > 0)**

* **The users table already exists; CHECK constraints can only be added during table creation using CREATE TABLE command.**
* **The users table already contains rows that violate the new constraint; all existing rows must satisfy the constraint before adding it to the table.**
* **The users table is not partitioned on the age column. CHECK constraints can only be added on partitioning columns.**
* **The users table already contains rows; CHECK constraints can only be added on empty tables**

**Explanation**

ADD CONSTRAINT command verifies that all existing rows in the table satisfy the constraint before adding it to the table. Otherwise, the command failed with an error that says some rows in the table violate the new CHECK constraint.

Reference:

<https://docs.databricks.com/tables/constraints.html#set-a-check-constraint-in-databricks>

Bottom of Form

Top of Form

Question 41:

A data engineer has added a CHECK constraint to the **sales** table using the following command:

ALTER TABLE sales ADD CONSTRAINT valid\_date CHECK (item\_date >= '2024-01-01');

In addition, they have added a comment on the **item\_date** column using the following command:

ALTER TABLE sales ALTER COLUMN item\_date COMMENT "Date must be newer than Jan 1, 2024";

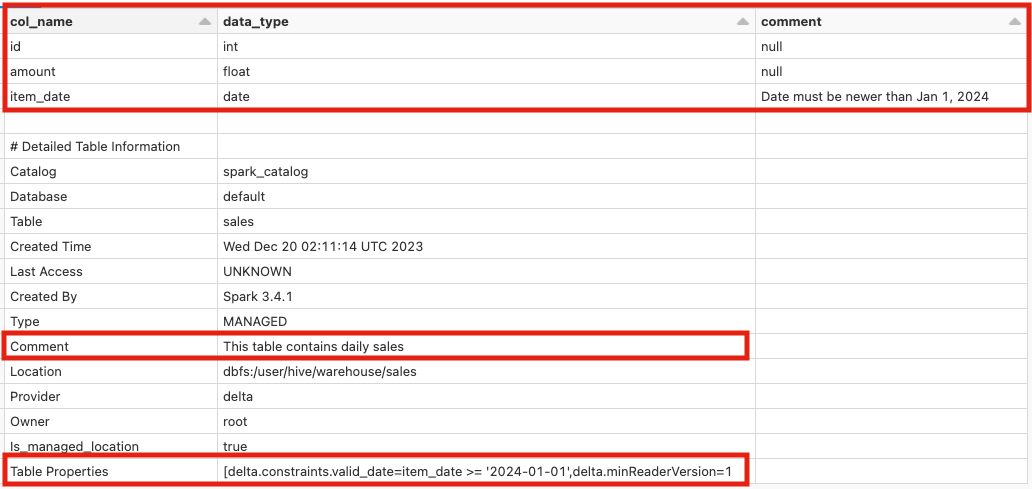
Which of the following commands allows the data engineer to verify that both the constraint and the column comment have been successfully added on the table ?

* **SHOW TBLPROPERTIES sales**
* **DESCRIBE TABLE sales**
* **DESCRIBE DETAIL sales**
* **DESCRIBE EXTENDED sales**
* **SHOW TABLES sales**

**Explanation**

DESCRIBE TABLE EXTENDED or simply DESCRIBE EXTENDED allows to show the added tables constraints in the ‘Table Properties’ field. It shows both the name and the actual condition of the check constraints.

In addition, DESCRIBE EXTENDED allows to show the comments on each column, and the comment on the table itself.



Reference:

* <https://docs.databricks.com/sql/language-manual/sql-ref-syntax-aux-describe-table.html>
* <https://docs.databricks.com/tables/constraints.html#set-a-check-constraint-in-databricks>

## Question 42:

Which of the following is the benefit of Delta Lake File Statistics ?

* **They are leveraged for process time forecasting when executing selective queries.**
* **They are leveraged for data skipping when executing selective queries.**
* **They are leveraged for data compression in order to improve Delta Caching.**
* **They are used as checksums to check data corruption in parquet files.**
* **None of the above statements correctly describe the benefit of Delta Lake File Statistics.**

**Explanation**

Delta Lake File Statistics indicate per file:

* Total number of records
* Minimum value in each column of the first 32 columns of the table
* Maximum value in each column of the first 32 columns of the table
* Null value counts for in each column of the first 32 columns of the table

These statistics are leveraged for data skipping based on query filters. For example, if you are querying the total number of records in a table, Delta will not calculate the count by scanning all data files. Instead, it will leverage these statistics to generate the query result

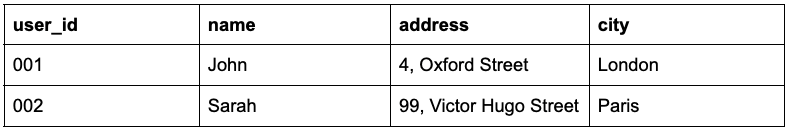
Reference:

<https://docs.databricks.com/delta/data-skipping.html>

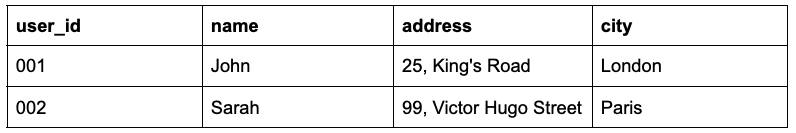
## Question 43:

Given the following two versions of a Delta Lake table before and after an update:

Before:

****

After:

****

Which SCD Type is this table ?

* **SCD Type 0**
* **SCD Type 1**
* **SCD Type 2**
* **It's a combination of Type 0 and Type 2 SCDs**
* **More information is needed to determine the SCD Type of this table**

**Explanation**

In a Type 1 SCD table the new data overwrites the existing one. Thus, the existing data is lost as it is not stored anywhere else.

Reference:

<https://en.wikipedia.org/wiki/Slowly_changing_dimension>

## Question 44:

The data engineering team created a new Databricks job for processing sensitive financial data. A financial analyst asked the team to transfer the "Owner" privilege of this job to the “finance” group.

A junior data engineer that has the “CAN MANAGE” permission on the job is attempting to make this privilege transfer via Databricks Job UI, but it keeps failing.

Which of the following explains the cause of this failure?

* **The "Owner" privilege is assigned at job creation to the creator and cannot be changed. The job must be re-created using the “finance” group’s credentials.**
* **Databricks Jobs UI doesn’t support changing the owners of jobs. Databricks REST API needs to be used instead.**
* **Having the “CAN MANAGE” permission is not enough to grant "Owner" privileges to a group. The data engineer must be the current owner of the job.**
* **Having the “CAN MANAGE” permission is not enough to grant "Owner" privileges to a group. The data engineer must be a workspace administrator.**
* **Groups can not be owners of Databricks jobs. The owner must be an individual user.**

**Explanation**

A job cannot have a group as an owner. If you try to set a group as the owner of a job, you get the error “Groups can not be owners”

Reference:

<https://docs.databricks.com/security/auth-authz/access-control/jobs-acl.html#job-permissions>

## Question 45:

The data engineering team noticed that a partitioned Delta Lake table is suffering greatly. They are experiencing slowdowns for most general queries on this table.

The team tried to run an OPTIMIZE command on the table, but this did not help to resolve the issue.

Which of the following likely explains the cause of these slowdowns?

* **The table has too many old data files that need to be purged. They need to run a VACUUM command instead.**
* **The table is over-partitioned or incorrectly partitioned. This requires a full rewrite of all data files to resolve the issue.**
* **They are applying the OPTIMIZE command on the whole table. It must be applied at each partition separately.**
* **They are applying the OPTIMIZE command without ZORDER. Z-ordering is needed on the partitioning columns.**
* **The transaction log is too large. Log files older than a certain age must be deleted or archived at partition boundaries.**

**Explanation**

Data that is over-partitioned or incorrectly partitioned will suffer greatly. Files cannot be combined or compacted across partition boundaries, so partitioned small tables increase storage costs and total number of files to scan. This leads to slowdowns for most general queries. Such an issue requires a full rewrite of all data files to remedy.

Reference:

<https://docs.databricks.com/tables/partitions.html>

## Question 46:

The data engineering team has the following query for processing customers’ requests to be forgotten:

DELETE FROM customers

WHERE customer\_id IN

(SELECT customer\_id FROM delete\_requests)

Which statement describes the results of executing this query ?

* **The identified records will be deleted from the customers tables, and their associated data files will be permanently purged from the table directory.**
* **The identified will be deleted from the both customers and delete\_requests tables, and their associated data files will be permanently purged from the tables directories.**
* **The identified records will be deleted from the customers table, but they will still be accessible in the table history until a VACUUM command is run.**
* **The identified records will be deleted from both customers and delete\_requests tables, but they will still be accessible in the table history until VACUUM commands are run.**
* **The identified records will be deleted from the customers tables, but they will still be accessible in the table history until updating the status of the requests in the delete\_requests table**

**Explanation**

Delete requests, also known as requests to be forgotten, require deleting user data that represent Personally Identifiable Information or PII, such as the name and the email of the user.

Because of how Delta Lake tables time travel are implemented, deleted values are still present in older versions of the data. Remember, deleting data does not delete the data files from the table directory. Instead, it creates a copy of the affected files without these deleted records. So, to fully commit these deletes, you need to run VACUUM commands on the customers table.

Reference:

<https://www.databricks.com/blog/2022/03/23/implementing-the-gdpr-right-to-be-forgotten-in-delta-lake.html>

## Question 47:

Given the following commands:

CREATE DATABASE db\_hr;

LOCATION '/mnt/hr\_external';

USE db\_hr;

CREATE TABLE employees;

In which of the following locations will the employees table be located?

* **dbfs:/user/hive/warehouse**
* **dbfs:/user/hive/warehouse/db\_hr.db**
* **/mnt/hr\_external**
* **/mnt/hr\_external/db\_hr.db**
* **More information is needed to determine the correct answer**

**Explanation**

Since we are creating the database here with the LOCATION keyword, it will be created as an external database under the specified location ‘/mnt/hr\_external’. The database folder has the extension (.db)

And since we are creating the table without specifying a location, the table becomes a managed table created under the database directory (in db\_hr.db folder)

Reference:<https://docs.databricks.com/sql/language-manual/sql-ref-syntax-ddl-create-schema.html>

## Question 48:

The data engineering team has a secret scope named ‘DataOps-Prod’ that contains all secrets needed by DataOps engineers in a production workspace.

Which of the following is the minimum permission required for the DataOps engineers to use the secrets in this scope ?

* **MANAGE permission on the “DataOps-Prod” scope**
* **READ permission on the “DataOps-Prod” scope**
* **MANAGE permission on each secret in the “DataOps-Prod” scope**
* **READ permission on each secret in the “DataOps-Prod” scope**
* **Workspace Administrator role**

**Explanation**

The secret access permissions are as follows:

* MANAGE - Allowed to change ACLs, and read and write to this secret scope.
* WRITE - Allowed to read and write to this secret scope.
* READ - Allowed to read this secret scope and list what secrets are available.

Each permission level is a subset of the previous level’s permissions (that is, a principal with WRITE permission for a given scope can perform all actions that require READ permission).

Reference:

<https://docs.databricks.com/security/auth-authz/access-control/secret-acl.html#permission-levels>

## Question 50

Which of the following is **Not** part of the Ganglia UI ?

* **Memory usage**
* **Overall workload of the cluster**
* **CPU usage**
* **Lifecycle events of the cluster**
* **Network performance**

**Explanation**

Lifecycle events of the cluster are not part of Ganglia UI.

Ganglia allows monitoring the performance of Databricks clusters. Ganglia UI provides you with the overall workload of the cluster, in addition to detailed metrics on memory, CPI, and Network usage.

Lifecycle events of the cluster are part of the Cluster Event log.

Reference:

<https://docs.databricks.com/clusters/clusters-manage.html#view-cluster-logs>

## Question 50:

In Spark UI, which of the following is **Not** part of the metrics displayed in a stage’s details page ?

* **Duration**
* **Spill (Memory)**
* **Spill (Disk)**
* **DBU Cost**
* **GC time**

**Explanation**

In Spark UI, the stage’s details page shows summary metrics for completed tasks. This includes:

* Duration of tasks.
* GC time: is the total JVM garbage collection time.
* Shuffle spill (memory): is the size of the deserialized form of the shuffled data in memory.
* Shuffle spill (disk): is the size of the serialized form of the data on disk.
* and others …

DBU Cost is not part of Spark UI. DBU stands for Databricks Unit and it is a unit of processing capability per hour for pricing purposes. This depends on your cluster configuration which tells you how much DBUs would be consumed if a virtual machine runs for an hour, and then pays for each DBU consumed.

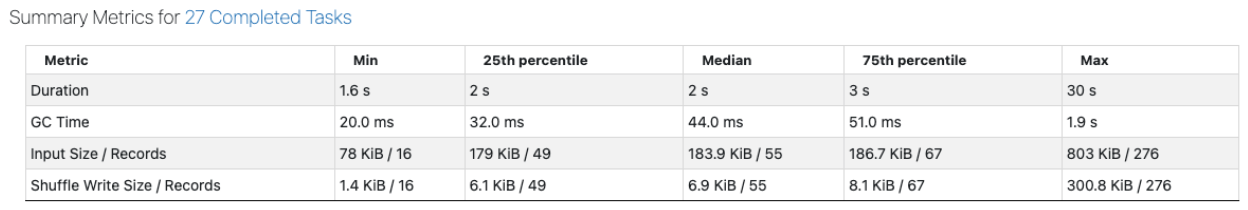
Reference:

<https://spark.apache.org/docs/latest/web-ui.html>

<https://www.databricks.com/product/pricing>

## Question 51:

A data engineer is analyzing a Spark job via the Spark UI. They have the following summary metrics for 27 completed tasks in a particular stage

****

Which conclusion can the data engineer draw from the above statistics ?

* **All task are operating over partitions with even amounts of data**
* **All task are operating over empty or near empty partitions**
* **All tasks are operating over partitions with larger skewed amounts of data.**
* **Number of tasks are operating over partitions with larger skewed amounts of data.**
* **Number of tasks are operating over empty or near empty partitions**

**Explanation**

Usually, if your computation was completely symmetric across tasks, you would see all of the statistics clustered tightly around the 50th percentile value.

Here, the “Max” metrics task took 10x the time and read about 5x the data of the 75th-percentile task. This suggests a number of “straggler” tasks that operating over partitions with larger skewed amounts of data.

## Question 52:

A data engineer is using Databricks REST API to send a GET request to the endpoint ‘api/2.1/jobs/runs/get’ to retrieve the run’s metadata of a multi-task job using its run\_id.

Which statement correctly describes the response structure of this API call?

* **Each task of this job run will have a unique task\_id**
* **Each task of this job run will have a unique run\_id**
* **Each task of this job run will have a unique job\_id**
* **Each task of this job run will have a unique orchestration\_id**
* **Tasks does not any unique identifier within a job run**

**Explanation**

Each task of this job run will have a unique run\_id to retrieve its output with endpoint ‘api/2.1/jobs/runs/get-output’

Reference:

<https://docs.databricks.com/dev-tools/api/latest/jobs.html#operation/JobsRunsGetOutput>

## Question 53:

A data engineer has noticed the comment ‘# Databricks notebook source’ on the first line of each Databricks Python file’s source code pushed to Github.

Which of the following explain the purpose of this comment ?

* **This comment makes it easier for humans to understand the source of the generated code from Databricks**
* **This comment establishes the Python files as Databricks notebooks**
* **This comment is used for Python auto-generated documentation**
* **This comment add the Python file to the search index in Databricks workspace**
* **There is no special purpose for this comment. Comments in Python are ignored by the interpreter.**

**Explanation**

You can convert Python, SQL, Scala, and R scripts to single-cell notebooks by adding a comment to the first cell of the file:

# Databricks notebook source

Reference:

<https://docs.databricks.com/notebooks/notebook-export-import.html#import-a-file-and-convert-it-to-a-notebook>

## Question 54:

Which of the following statements best describes DBFS ?

* **Database File System that organizes and maintains data files in Databricks workspace**
* **Database File System to interact with files in cloud-based object storage**
* **Abstraction on top of Databricks Lakehouse that provides an open solution to share data to any computing platform**
* **Abstraction on top of scalable object storage that maps Unix-like file system calls to native cloud storage API calls.**
* **None of the above correctly describes DBFS**

**Explanation**

The Databricks File System (DBFS) is a distributed file system mounted into a Databricks workspace and available on Databricks clusters. DBFS is an abstraction on top of scalable object storage that maps Unix-like filesystem calls to native cloud storage API calls.

Reference:

<https://docs.databricks.com/dbfs/index.html>

## Question 55:

A data engineer wants to install a Python wheel scoped to the current notebook’s session, so only the current notebook and any jobs associated with this notebook have access to that library.

Which of the following commands can the data engineer use to complete this task?

* **%fs install my\_package.whl**
* **%pip install my\_package.whl**
* **%python install my\_package.whl**
* **%whl install my\_package**
* **Python wheels can not be installed at the notebook level. They can only be installed at the cluster level.**

**Explanation**

‘%pip install’ allows you to install a Python wheel scoped to the current notebook’s session. This library will be only accessible in the current notebook and any jobs associated with this notebook.

Reference:

<https://docs.databricks.com/libraries/notebooks-python-libraries.html>

Bottom of Form

Top of Form

Question 56:

Which of the following statements correctly describes the sys.path Python variable ?

* **The sys.path variable contains a list of all the parameters passed to a Python notebook.**
* **The sys.path variable contains a list of all the necessary dependencies for a Python notebook.**
* **The sys.path variable contains a list of directories where the Python interpreter searches for modules.**
* **The sys.path variable contains the full pathname of the current working directory of a Python notebook.**
* **The sys.path variable is an alias for os.path**

**Explanation**

The sys.path variable contains a list of directories where the Python interpreter searches for modules.

To import modules from another directory, you must add it to sys.path

1. import sys
2. sys.path.append("/path/to/dir")

Reference:

<https://docs.databricks.com/files/workspace-modules.html#import-python-and-r-modules-1>

## Question 57

Which of the following statements correctly describes assertions in unit testing ?

* **An assertion is a boolean expression that checks if two code blocks are integrated logically and interacted as a group.**
* **An assertion is a boolean expression that checks if assumptions made in the code remain true while development**
* **An assertion is a command that logs failed units of code in production for later debugging and analysis**
* **An assertion is a command that shows the differences between the current version of a code unit and the most recently edited version**
* **An assertion is a set of actions that simulates a user experience to ensure that the application can run properly under real-world scenarios**

**Explanation**

Assertions are boolean expressions that enable you to test the assumptions you have made in your code. They are used in unit tests to check if certain assumptions remain true while you're developing your code.

assert func() == expected\_value

Reference:

<https://docs.databricks.com/notebooks/testing.html>

Bottom of Form

Top of Form

Question 58:

Which of the following statements correctly describes End-to-End Testing ?

* **It’s an approach to simulate a user experience to ensure that the application can run properly under real-world scenarios**
* **It’s an approach to test the interaction between subsystems of an application to ensure that modules work properly as a group.**
* **It’s an approach to test individual units of code to determine whether they still work as expected if new changes are made to them in the future**
* **It’s an approach to verify if each feature of the application works as per the business requirements**
* **It’s an approach to measure the reliability, speed, scalability, and responsiveness of an application**

**Explanation**

End-to-end testing is an approach to ensure that your application can run properly under real-world scenarios. The goal of this testing is to simulate a real user experience from start to finish.

## Question 59:

When running an existing job via Databricks REST API, which of the following represents the globally unique identifier of the newly triggered run ?

* **job\_id**
* **run\_id**
* **run\_key**
* **task\_id**
* **task\_key**

**Explanation**

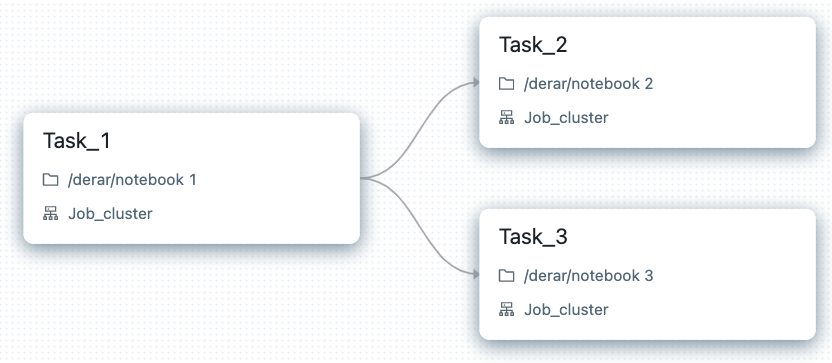
Running an existing job via the endpoint ‘/api/2.1/jobs/run-now’ returns the run\_id of the triggered run. This represents the globally unique identifier of this newly triggered run.

Reference:

<https://docs.databricks.com/dev-tools/api/latest/jobs.html#operation/JobsRunNow>

## Question 60:

Given the following multi-task job

****

If there is an error in the notebook 2 that is associated with Task 2, which statement describes the run result of this job ?

* **Task 1 will succeed. Task 2 will partially fail. Task 3 will be skipped**
* **Task 1 will succeed. Task 2 will completely fail. Task 3 will be skipped**
* **Tasks 1 and 3 will succeed, while Task 2 will partially fail**
* **Tasks 1 and 3 will succeed, while Task 2 will completely fail**
* **All tasks will completely fail**

**Explanation**

If a task fails during a job run, only the dependent tasks, if any, will be skipped. Parallel tasks will run and complete.

The failure of a task will always be partial, which means that the operations in the notebook before the code failure will be successfully run and committed, while the operations after the code failure will be skipped.

Reference:

<https://docs.databricks.com/workflows/jobs/repair-job-failures.html>

## Question 61

Which of the following describes the minimal permissions a data engineer needs to attach a notebook to an existing cluster ?

* **“Can Attach To” privilege on the cluster**
* **“Can Restart” privilege on the cluster**
* **“Can Manage” privilege on the cluster**
* **Cluster creation allowed + “Can Attach To” privileges on the cluster**
* **Cluster creation allowed + “Can Restart” privileges on the cluster**

**Explanation**

You can configure two types of cluster permissions:

1- The ‘Allow cluster creation’ entitlement controls your ability to create clusters.

2- Cluster-level permissions control your ability to use and modify a specific cluster. There are four permission levels for a cluster: No Permissions, Can Attach To, Can Restart, and Can Manage. The table lists the abilities for each permission:



Reference:

<https://docs.databricks.com/security/auth-authz/access-control/cluster-acl.html>

## Question 62

Which of the following describes the minimal permissions a data engineer needs to start an existing cluster, and attach a notebook to it?

* **“Can Attach To” privilege on the cluster**
* **“Can Restart” privilege on the cluster**
* **“Can Manage” privilege on the cluster**
* **Cluster creation allowed + “Can Attach To” privileges on the cluster**
* **Cluster creation allowed + “Can Restart” privileges on the cluster**

**Explanation**

You can configure two types of cluster permissions:

1- The ‘Allow cluster creation’ entitlement controls your ability to create clusters.

2- Cluster-level permissions control your ability to use and modify a specific cluster. There are four permission levels for a cluster: No Permissions, Can Attach To, Can Restart, and Can Manage. The table lists the abilities for each permission:



Reference:

<https://docs.databricks.com/security/auth-authz/access-control/cluster-acl.html>

## Question 63

The data engineering team has a Delta Lake table created with following query:

CREATE TABLE customers\_clone

AS SELECT \* FROM customers

A data engineer wants to drop the table with the following query:

DROP TABLE customers\_clone

Which statement describes the result of running this drop command ?

* **An error will occur as the table is deep cloned from the customers table**
* **An error will occur as the table is shallow cloned from the customers table**
* **Only the table's metadata will be deleted from the catalog, while the data files will be kept in the storage**
* **Both the table's metadata and the data files will be deleted**
* **The table will not be dropped until VACUUM command is run**

**Explanation**

The table is created without the LOCATION clause, which means that it’s a managed table. Managed tables are tables whose metadata and data are managed by Databricks.

When you run DROP TABLE on a managed table, both the metadata and the underlying data files are deleted.

Reference:

<https://docs.databricks.com/lakehouse/data-objects.html#what-is-a-managed-table>

## Question 64:

For production Structured Streaming jobs, which of the following retry policies is recommended to use ?

* **Unlimited Retries, with 1 Maximum Concurrent Run**
* **Unlimited Retries, with Unlimited Concurrent Runs**
* **No Retries, with 1 Maximum Concurrent Run**
* **No Retries, with Unlimited Concurrent Runs**
* **1 Retry, with 1 Maximum Concurrent Run**

**Explanation**

In order to restart streaming queries on failure, it’s recommended to configure Structured Streaming jobs with the following job configuration:

* Retries: Set to Unlimited.
* Maximum concurrent runs: Set to 1. There must be only one instance of each query concurrently active.
* Cluster: Set this always to use a new job cluster and use the latest Spark version (or at least version 2.1). Queries started in Spark 2.1 and above are recoverable after query and Spark version upgrades.
* Notifications: Set this if you want email notification on failures.
* Schedule: Do not set a schedule.
* Timeout: Do not set a timeout. Streaming queries run for an indefinitely long time.

Reference:

<https://docs.databricks.com/structured-streaming/query-recovery.html#configure-structured-streaming-jobs-to-restart-streaming-queries-on-failure>

## Question 65:

A data engineer has a MLFlow model logged in a given “model\_url”. They have registered the model as a Spark UDF using the following code:

predict\_udf = mlflow.pyfunc.spark\_udf(spark, "model\_url")

The data engineer wants to apply this model UDF to a test dataset loaded in the “test\_df” DataFrame in order to calculate predictions in a new column “prediction”

Which of the following code blocks allows the data engineer to accomplish this task ?

* **test\_df.apply(predict\_udf, \*column\_list).select(“record\_id”, “prediction")**
* **test\_df.select(“record\_id”, predict\_udf(\*column\_list).allas("prediction"))**
* **predict\_udf(“record\_id”, test\_df).select(“record\_id”, “prediction")**
* **mlflow.pyfunc.map(predict\_udf, test\_df, “record\_id”).allas("prediction")**
* **mlflow.pyfunc.map(predict\_udf, test\_df, “record\_id”).allas("prediction")**

**Explanation**

In PySpark Dataframe, you can create a new column based on function return value. This can be achieved by calling the function using either:

* **Dataframe.withColumn** method:

test\_df.withColumn("prediction", predict\_udf(\*column\_list))

* Or using **Dataframe.select** method:

test\_df.select( predict\_udf(\*column\_list).allas("prediction") )

Dataframe.select allows also to select one or more columns:

test\_df.select(“record\_id”, predict\_udf(\*column\_list).allas("prediction"))

or it can be expanded to include all columns using “\*”

test\_df.select(“\*”, predict\_udf(\*column\_list).allas("prediction"))

Reference:

<https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.sql.DataFrame.select.html>

<https://spark.apache.org/docs/3.1.3/api/python/reference/api/pyspark.sql.DataFrame.withColumn.html>

## Question 66

In Delta Lake tables, which of the following is the file format for the transaction log ?

* **Delta**
* **Parquet**
* **JSON**
* **Hive-specific format**
* **Both, Parquet and JSON**

**Explanation**

Delta Lake builds upon standard data formats. Delta lake table gets stored on the storage in one or more data files in Parquet format, along with transaction logs in JSON format.

In addition, Databricks automatically creates Parquet checkpoint files every 10 commits to accelerate the resolution of the current table state.

Reference:

* <https://docs.databricks.com/delta/index.html>
* <https://www.databricks.com/blog/2019/08/21/diving-into-delta-lake-unpacking-the-transaction-log.html>

## Question 67

Which of the following describes the minimal permissions a data engineer needs to modify permissions of an existing cluster ?

* **“Can Restart” privilege on the cluster**
* **“Can Manage” privilege on the cluster**
* **Cluster creation allowed + “Can Restart” privileges on the cluster**
* **Cluster creation allowed + “Can Manage” privileges on the cluster**
* **Only administrators can modify permissions on existing clusters**

**Explanation**

You can configure two types of cluster permissions:

1- The ‘Allow cluster creation’ entitlement controls your ability to create clusters.

2- Cluster-level permissions control your ability to use and modify a specific cluster. There are four permission levels for a cluster: No Permissions, Can Attach To, Can Restart, and Can Manage. The table lists the abilities for each permission:



<https://docs.databricks.com/security/auth-authz/access-control/cluster-acl.html>

## Question 68

Which of the following is the default target file size when compacting small files of a Delta table by manually running OPTIMIZE command ?

* **64 MB**
* **128 MB**
* **256 MB**
* **512 MB**
* **1024 MB**

**Explanation**

The OPTIMIZE command compact small data files into larger ones. The default value is 1073741824, which sets the size to 1 GB.

Reference: <https://docs.databricks.com/sql/language-manual/delta-optimize.html>

## Question 69

A junior data engineer is using the following code to de-duplicate raw streaming data and insert them in a target Delta table

spark.readStream

.table("orders\_raw")

.dropDuplicates(["order\_id", "order\_timestamp"])

.writeStream

.option("checkpointLocation", "dbfs:/checkpoints")

.table("orders\_unique")

A senior data engineer pointed out that this approach is not enough for having distinct records in the target table when there are late-arriving, duplicate records.

Which of the following could explain the senior data engineer’s remark?

* **Watermarking is also needed to only track state information for a window of time in which we expect records could be delayed.**
* **A ranking function is also needed to ensure processing only the most recent records**
* **A window function is also needed to apply deduplication for each non-overlapping interval.**
* **The new records need also to be deduplicated against previously inserted data into the table.**
* **More information is needed to determine the correct response**

**Explanation**

To perform streaming deduplication, we use dropDuplicates() function to eliminate duplicate records within each new micro batch. In addition, we need to ensure that records to be inserted are not already in the target table. We can achieve this using insert-only merge.

Reference:

<https://spark.apache.org/docs/3.1.2/api/python/reference/api/pyspark.sql.DataFrame.dropDuplicates.html>

<https://docs.databricks.com/delta/merge.html#data-deduplication-when-writing-into-delta-tables>

## Question 70

“A feature built into Delta Lake that allows to automatically generate CDC feeds about Delta Lake tables”

Which of the following is being described in the above statement?

* **Auto Optimize**
* **Optimized writes**
* **Spark Watermarking**
* **Slowly Changing Dimension (SCD)**
* **Change Data Feed (CDF)**

**Explanation**

Change Data Feed ,or CDF, is a new feature built into Delta Lake that allows it to automatically generate CDC feeds about Delta Lake tables.

CDF records row-level changes for all the data written into a Delta table. This includes the row data along with metadata indicating whether the specified row was inserted, deleted, or updated.

Reference:

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

## Question 71

A data engineer uses the following SQL query:

GRANT MODIFY ON TABLE employees TO hr\_team

Which of the following describes the ability given by the MODIFY privilege ?

* **It gives the ability to add data from the table**
* **It gives the ability to delete data from the table**
* **It gives the ability to modify data in the table**
* **All the above abilities are given by the MODIFY privilege**
* **None of these options correctly describe the ability given by the MODIFY privilege**

**Explanation**

The MODIFY privilege gives the ability to add, delete, and modify data to or from an object.

Reference: <https://docs.databricks.com/security/access-control/table-acls/object-privileges.html#privileges>

## Question 72

Which of the following statements regarding the retention policy of Delta lake CDF is correct ?

* **Running the VACUUM command on the table deletes CDF data as well**
* **Running the VACUUM command on the table does not deletes CDF data**
* **Running the VACUUM command on the table does not deletes CDF data unless CASCADE clause is set to true**
* **CDF data files can be purged by running VACUUM CHANGES command**
* **CDF data files can never be permanently purged from Delta Lake**

**Explanation**

Databricks records change data for UPDATE, DELETE, and MERGE operations in the \_change\_data folder under the table directory.

The files in the \_change\_data folder follow the retention policy of the table. Therefore, if you run the VACUUM command, change data feed data is also deleted.

Reference:

<https://docs.databricks.com/delta/delta-change-data-feed.html#change-data-storage>

## Question 73:

Given the following Structured Streaming query:

(spark.readStream

.table("orders")

.writeStream

.option("checkpointLocation", checkpointPath)

.table("Output\_Table")

)

Which of the following is the trigger Interval for this query ?

* **Every half second**
* **Every half min**
* **Every half hour**
* **The query will run in batch mode to process all available data at once, then the trigger stops.**
* **More information is needed to determine the correct response**

**Explanation**

By default, if you don’t provide any trigger interval, the data will be processed every half second. This is equivalent to trigger(processingTime=”500ms")

Reference: <https://docs.databricks.com/structured-streaming/triggers.html#what-is-the-default-trigger-interval>

## Question 74

A data engineer run the following CTAS statement in a SQL notebook attached to an All-purpose cluster:

CREATE TABLE course\_students

AS ( SELECT c.course\_name, t.student\_id, t.student\_name

FROM courses c

LEFT JOIN (

SELECT s.student\_id, s.student\_name, e.course\_id

FROM students s

INNER JOIN enrollments e

ON s.student\_id = e.student\_id

) t

ON c.course\_id = t.course\_id

WHERE c.active = true

)

Which statement describes the resulting course\_students table ?

* **It’s a virtual table that has no physical data. The SELECT statement will be executed each time the course\_students table is queried.**
* **It’s a cluster-scoped virtual table. The SELECT statement will be executed only the first time the course\_students table is queried. The query output will be stored in the memory of the currently active cluster.**
* **It’s a Delta Lake table. The SELECT statement will be executed at the table creation, and its output will be stored in Delta format on the underlying storage.**
* **It’s a cluster-scoped table. The SELECT statement will be executed at the table creation, but its output will be stored in the memory of the currently active cluster.**
* **It’s a session-scoped table. The SELECT statement will be executed at the table creation, but its output will be stored in the cache of the current active Spark session.**

**Explanation**

CREATE TABLE AS SELECT statements, or CTAS statements create new Delta tables and populate them using the output of a SELECT query. So, The query result is stored in Delta format in the directory of the newly created table.

Reference: (cf. AS query clause)

<https://docs.databricks.com/sql/language-manual/sql-ref-syntax-ddl-create-table-using.html>

## Question 75:

A data engineer has a streaming job that updates a Delta table named ‘user\_activities’ by the results of a join between a streaming Delta table ‘activity\_logs’ and a static Delta table ‘users’.

They noticed that adding new users into the ‘users’ table does not automatically trigger updates to the ‘user\_activities’ table, even when there were activities for those users in the ‘activity\_logs’ table.

Which of the following likely explains this issue ?

* **The users table must be refreshed with REFRESH TABLE command for each microbatch of this join**
* **This stream-static join is not stateful by default unless they set the spark configuration delta.statefulStreamStaticJoin to true.**
* **The streaming portion of this stream-static join drives the join process. Only new data appearing on the streaming side of the join will trigger the processing.**
* **The static portion of the stream-static join drives this join process only in batch mode.**
* **In Delta Lake, static tables can not be joined with streaming tables.**

**Explanation**

In stream-static join, the streaming portion of this join drives this join process. So, only new data appearing on the streaming side of the join will trigger the processing. While, adding new records into the static table will not automatically trigger updates to the results of the stream-static join.

Reference: <https://docs.databricks.com/structured-streaming/delta-lake.html#performing-stream-static-joins>

## Question 76

Given the following query on the Delta table ‘customers’ on which Change Data Feed is enabled:

spark.read

.option("readChangeFeed", "true")

.option("startingVersion", 0)

.table ("customers")

.filter (col("\_change\_type").isin(["update\_postimage"]))

.write

.mode("append")

.table("customers\_updates")

Which statement describes the result of this query each time it is executed ?

* **Newly updated records will be merged into the target table, modifying previous entries with the same primary keys.**
* **Newly updated records will be appended to the target table.**
* **Newly updated records will overwrite the target table.**
* **The entire history of updated records will be appended to the target table at each execution, which leads to duplicate entries.**
* **The entire history of updated records will overwrite the target table at each execution.**

**Explanation**

Reading table’s changes, captured by CDF, using spark.read means that you are reading them as a static source. So, each time you run the query, all table’s changes (starting from the specified startingVersion) will be read.

The query in the question then appends the data to the target table at each execution since it’s using the ‘append’ writing mode.

Reference:

<https://docs.databricks.com/delta/delta-change-data-feed.html#read-changes-in-batch-queries>

## Question 77

A data engineer wants to use Autoloader to ingest input data into a target table, and automatically evolve the schema of the table when new fields are detected.

They use the below query with a blank:

spark.readStream

.format("cloudFiles")

.option("cloudFiles.format", "json")

.option("cloudFiles.schemaLocation", checkpointPath)

.load(source\_path)

.writeStream

.option("checkpointLocation", checkpointPath)

.\_\_\_\_\_\_\_\_\_\_\_

.start("target\_table")

Which option correctly fills in the blank to meet the specified requirement ?

* **option("cloudFiles.schemaEvolutionMode", "addNewColumns")**
* **option("cloudFiles.mergeSchema", True)**
* **option("mergeSchema", True)**
* **schema(schema\_definition, mergeSchema=True)**
* **Autoloader can not automatically evolve the schema of the table when new fields are detected**

**Explanation**

Schema evolution is a feature that allows adding new detected fields to the table. It’s activated by adding .option('mergeSchema', 'true') to your .write or .writeStream Spark command.

Reference:

<https://docs.databricks.com/delta/update-schema.html#add-columns-with-automatic-schema-update>

## Question 78

Given the following query:

spark.table("stream\_sink")

.filter("recent = true")

.dropDuplicates(["item\_id", "item\_timestamp"])

.write

.mode ("overwrite")

.table("stream\_data\_stage")

Which statement describes the result of executing this query ?

* **An incremental job will overwrite the stream\_sink table by those deduplicated records from stream\_data\_stage that have been added since the last time the job was run.**
* **An incremental job will overwrite the stream\_data\_stage table by those deduplicated records from stream\_sink that have been added since the last time the job was run.**
* **A batch job will overwrite the stream\_sink table by deduplicated records calculated from all “recent” items in the stream\_data\_stage table**
* **A batch job will overwrite the stream\_data\_stage table by deduplicated records calculated from all “recent” items in the stream\_sink table**
* **A batch job will overwrite the stream\_data\_stage table by those deduplicated records from stream\_sink that have been added since the last time the job was run.**

**Explanation**

Reading a Delta table using spark.table() function means that you are reading it as a static source. So, each time you run the query, all records in the current version of the ‘stream\_sink’ table will be read, filtered and deduplicated.

There is no difference between spark.table() and spark.read.table() function. Actually, spark.read.table() internally calls spark.table().

The query in the question then writes the data in mode “overwrite” to the ‘stream\_data\_stage’ table, which completely overwrites the table at each execution.

Reference:

<https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrameReader.table.html>

## Question 79:

The data engineering team has a Delta table named ‘users’. A recent CHECK constraint has been added to the table using the following command:

ALTER TABLE users

ADD CONSTRAINT valid\_age CHECK (age> 0);

The team attempted to insert a batch of new records to the table, but there were some records with negative age values which caused the write to fail because of the constraint violation.

Which statement describes the outcome of this batch insert?

* **All records except those that violate the table constraint have been inserted in the table. Records violating the constraint have been ignored.**
* **All records except those that violate the table constraint have been inserted in the table. Records violating the constraint have been recorded into the transaction log.**
* **Only records processed before reaching the first violating record have been inserted in the table**
* **None of the records have been inserted into the table.**
* **The outcome depends on the action defined using the ON VIOLATION clause in the table properties.**

**Explanation**

Write operations failed because of the constraint violation. However, ACID guarantees on Delta Lake ensure that all transactions are atomic. That is, they will either succeed or fail completely. So in this case, none of these records have been inserted into the table, even the ones that don't violate the constraints.

Reference:

<https://www.databricks.com/glossary/acid-transactions>

## Question 80:

Which statement regarding Delta Lake File Statistics is **Not** correct?

* **The statistics are leveraged for data skipping when executing selective queries.**
* **The statistics are generally uninformative for string fields with very high cardinality.**
* **Delta Lake captures statistics in the transaction log for each added data file**
* **By default, Delta Lake captures statistics on the first 32 columns of the table.**
* **Nested fields do not count when determining the first 32 columns in the table.**

**Explanation**

Delta Lake automatically captures statistics in the transaction log for each added data file of the table. By default, Delta Lake collects the statistics on the first 32 columns of each table. Nested fields count when determining the first 32 columns

Example: 4 struct fields with 8 nested fields will total to the 32 columns.

Reference: <https://docs.databricks.com/delta/data-skipping.html>

## Question 81

Which of the following is **Not** a valid Delta Lake File Statistics ?

* **The total number of records in the added data file.**
* **The minimum value in each of the first 32 columns**
* **The maximum value in each of the first 32 columns**
* **The average value for each of the first 32 columns**
* **The number of null values for each of the first 32 columns**

**Explanation**

Delta Lake automatically captures statistics in the transaction log for each added data file of the table. These statistics indicate per file:

* Total number of records
* Minimum value in each column of the first 32 columns of the table
* Maximum value in each column of the first 32 columns of the table
* Null value counts for in each column of the first 32 columns of the table

The average value in the columns is Not part of Delta Lake File Statistics

Reference: <https://docs.databricks.com/delta/data-skipping.html>

## Question 82

Which of the following definitions correctly describes a Slowly Changing Dimension of Type 0 ?

* **It’s a table where no changes are allowed.**
* **It’s a table where history will be kept in the additional column**
* **It’s a table where the new arriving data overwrites the existing one.**
* **It’s a table that stores and manages both current and historical data over time.**
* **It’s a table that maintains current records, while older records are stored in another table.**

**Explanation**

Type 0 SCD tables never change. Tables of this type are usually static. For example, static lookup tables.

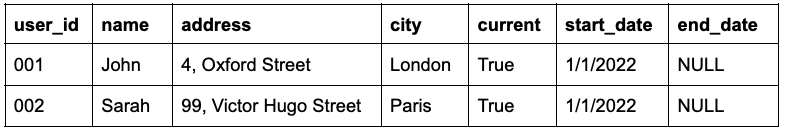
Reference:

<https://en.wikipedia.org/wiki/Slowly_changing_dimension>

## Question 83:

Given the following two versions of a Delta Lake table before and after an update:

Before:

****

After:

****

Which SCD Type is this table ?

* **SCD Type 0**
* **SCD Type 1**
* **SCD Type 2**
* **It's a combination of Type 0 and Type 2 SCDs**
* **More information is needed to determine the SCD Type of this table**

**Explanation**

In a Type 2 SCD table, a new record is added with the changed data values, and this new record becomes the current active record, while the old record is marked as no longer active. So, Type 2 SCD retains the full history of values

Reference:

<https://en.wikipedia.org/wiki/Slowly_changing_dimension>

## Question 84:

The data engineering team maintains a Delta Lake table of SCD Type 1. A junior data engineer noticed a folder named ‘\_change\_data’ in the table directory, and wants to understand what this folder is used for.

Which of the following describes the purpose of this folder ?

* **The ‘\_change\_data’ folder is a metadata directory to track any update to the table definition**
* **The ‘\_change\_data’ folder is the default directory to track the evolution in schema definition**
* **All SCD Type 1 tables have the ‘\_change\_data’ folder to track the updates applied on the table’s data.**
* **Optimized Writes feature is enabled on the table. The ‘\_change\_data’ folder is the location where the optimized data is stored**
* **CDF feature is enabled on the table. The ‘\_change\_data’ folder is the location where CDF data is stored**

**Explanation**

Databricks records change data for UPDATE, DELETE, and MERGE operations in the \_change\_data folder under the table directory.

The files in the \_change\_data folder follow the retention policy of the table. Therefore, if you run the VACUUM command, change data feed data is also deleted.

Reference:

<https://docs.databricks.com/delta/delta-change-data-feed.html#change-data-storage>

## Question 85:

A data engineer has the following streaming query with a blank:

spark.readStream

.table("orders\_cleaned")

.groupBy(

"author")

.agg(

count("order\_id").alias("orders\_count"),

avg("quantity").alias("avg\_quantity"))

.writeStream

.option("checkpointLocation", "dbfs:/path/checkpoint")

.table("orders\_stats")

They want to calculate the orders count and average quantity for each non-overlapping 15-minute interval.

Which option correctly fills in the blank to meet this requirement ?

* **trigger(processingTime=”15 minutes")**
* **window("order\_timestamp", "15 minutes")**
* **withWindow("order\_timestamp", "15 minutes")**
* **withWatermark("order\_timestamp", "15 minutes")**
* **interval("order\_timestamp", "15 minutes")**

**Explanation**

Pyspark.sql.functions.window function bucketizes rows into one or more time windows given a timestamp specifying column.

Reference:

<https://spark.apache.org/docs/3.1.3/api/python/reference/api/pyspark.sql.functions.window.html>

## Question 86:

The data engineering team is looking for a simple solution to share part of a large Delta Lake table with the data science team. Only department-specific columns in the table need to be shared, but with different names. In addition, there is some sensitive data that must be filtered out before sharing.

Which of the following objects can be created to meet the specified requirements ?

* **A new Delta Table created using DEEP CLONE from the existing table**
* **A new Delta Table created using SHALLOW CLONE from the existing table**
* **A new Delta Table created using CTAS statement on the existing table**
* **A stored view on the existing table**
* **An ad-hoc query on the existing table**

**Explanation**

The solution in this case is to create a view on the table where the required columns can be renamed, and the sensitive data that can be filtered out with the WHERE clause.

Reference: <https://docs.databricks.com/sql/language-manual/sql-ref-syntax-ddl-create-view.html>

## Question 87:

A junior data engineer has created the table ‘orders\_backup’ as a copy of the table “orders”. Recently, the team started getting an error when querying the orders\_backup indicating that some data files are no longer present. The transaction logs for the orders tables show a recent run of VACUUM command.

Which of the following explains how the data engineer created the orders\_backup table ?

* **The orders\_backup table was created using CTAS statement from orders table**
* **The orders\_backup table was created using CRAS statement from orders table**
* **The orders\_backup table was created using Delta Lake’s SHALLOW CLONE functionality from the orders table**
* **The orders\_backup table was created using Delta Lake’s DEEP CLONE functionality from the orders table**
* **The orders\_backup table was created by fully copying the orders table’s directory**

**Explanation**

With Shallow Clone, you create a copy of a table by just copying the Delta transaction logs.

That means that there is no data moving during Shallow Cloning.

Running the VACUUM command on the source table may purge data files referenced in the transaction log of the clone. In this case, you will get an error when querying the clone indicating that some data files are no longer present.

Reference:

<https://docs.databricks.com/delta/clone.html>

## Question 88:

Which of the following statements regarding cloning tables on Databricks

is correct ?

* **Any changes made to either deep or shallow clones affect only the clones themselves and not the source table.**
* **Any changes made to deep clones affect only the clones themselves and not the source table. While, changes made to shallow clones affect the source table.**
* **Any changes made to shallow clones affect only the clones themselves and not the source table. While, changes made to deep clones affect the source table.**
* **Changes made to either deep or shallow clones affect the source table.**
* **Changes made to either deep or shallow clones affect the source table unless CASCADE option is False.**

**Explanation**

In either case, deep or shallow cloning, data modifications applied to the cloned version of the table will be tracked and stored separately from the source, so it will not affect the source table.

Reference:

<https://docs.databricks.com/delta/clone.html#clone-types-1>

## Question 89:

A data engineer wants to optimize the following join operation by allowing the smaller dataFrame to be sent to all executor nodes in the cluster:

1. largeDF.join(smallerDF, [“key”], "inner")

Which of the following functions can be used to mark a dataFrame as small enough to fit in memory on all executors ?

* **pyspark.sql.functions.distribute**
* **pyspark.sql.functions.explode**
* **pyspark.sql.functions.broadcast**
* **pyspark.sql.functions.diffuse**
* **pyspark.sql.functions.shuffle**

**Explanation**

Pyspark.sql.functions.broadcast function marks a DataFrame as small enough for use in broadcast joins.

Reference:

<https://spark.apache.org/docs/3.1.3/api/python/reference/api/pyspark.sql.functions.broadcast.html>

## Question 90

The data engineering team wants to create a multiplex bronze Delta table from a Kafka source. The Delta Table has the following schema:

key BINARY, value BINARY, topic STRING, partition LONG, offset LONG, timestamp LONG

Since the “value” column contains Personal Identifiable Information (PII) for some topics, the team wants to apply Access Control Lists (ACLs) at partition boundaries to restrict access to this PII data.

Based on the above schema and the specified requirement, which column is a good candidate for partitioning ?

* **key**
* **value**
* **topic**
* **partition**
* **timestamp**

**Explanation**

Table partitioning helps improve security. You can separate sensitive and nonsensitive data into different partitions and apply different security controls to the sensitive data.

\* Personally Identifiable Information or PII represents any information that allows identifying individuals by either direct or indirect means, such as the name and the email of the user.

## Question 91

The data engineering team wants to know if the tables that they maintain in the Lakehouse are over-partitioned.

Which of the following is an indicator that a Delta Lake table is over-partitioned ?

* **If the number of partitions in the table are too low**
* **If most partitions in the table have less than 1 GB of data**
* **If most partitions in the table have more than 1 GB of data**
* **If the partitioning columns are fields of low cardinality**
* **If the data in the table continues to arrive indefinitely.**

**Explanation**

Data that is over-partitioned or incorrectly partitioned will suffer greatly. Files cannot be combined or compacted across partition boundaries, so partitioned small tables increase storage costs and total number of files to scan. This leads to slowdowns for most general queries.

If most partitions in a table have less than 1GB of data, the table is likely over-partitioned

Reference:

<https://docs.databricks.com/tables/partitions.html>

## Question 92:

A data engineer is using the following spark configurations in a pipeline to enable Optimized Writes and Auto Compaction:

spark.conf.set("spark.databricks.delta.optimizeWrite.enabled", True)

spark.conf.set("spark.databricks.delta.autoCompact.enabled", True)

They also want to enable Z-order indexing with Auto Compaction to leverage data skipping on all the pipeline’s tables.

Which of the following solutions allows the data engineer to complete this task ?

* Use spark.conf.set("spark.databricks.delta.autoZorder.enabled", True)
* Use spark.conf.set("spark.databricks.delta.autoCompact.zorder.enabled", True)
* **Z-order indexing with Auto Compaction can only be enabled on each table separately using:**

**ALTER TABLE table\_name**

**SET TBLPROPERTIES (delta.autoOptimize.zorder.enabled = true)**

* **There is no need for extra configurations. Z-Ordering is enabled by default with Auto Compaction**
* **There is no way to enable Z-order indexing with Auto Compaction since it does not support Z-Ordering**

**Explanation**

Auto Compaction does not support Z-Ordering as Z-Ordering is significantly more expensive than just compaction.

Reference:

<https://docs.databricks.com/delta/tune-file-size.html#when-to-run-optimize>

## Question 93:

The data engineering team maintains the following join logic between three Delta tables:

df\_students = spark.table("students")

df\_courses = spark.table("courses")

df\_enrollments = spark.table("enrollments")

df\_join\_1 = (df\_students.join(df\_enrollments, df\_students.student\_id == df\_enrollments.student\_id)

.select(df\_students.student\_id,

df\_students.student\_name,

df\_enrollments.course\_id)

)

df\_join\_2 = (df\_join\_1.join(df\_courses, df\_join\_1.course\_id == df\_courses.course\_id)

.select(df\_join\_1.student\_id,

df\_join\_1.student\_name,

df\_courses.course\_name)

)

(df\_join\_2.write

.mode("overwrite")

.table("students\_courses\_details"))

Which statement describes the result of this code block each time it is executed ?

* **All records in the current version of the source tables will be considered in the join operations. The matched records will overwrite the students\_courses\_details table.**
* **All records in the current version of the source tables will be considered in the join operations. The unmatched records will overwrite the students\_courses\_details table.**
* **Only newly added records to any of the source tables will be considered in the join operations. The matched records will overwrite the students\_courses\_details table.**
* **Only newly added records to any of the source tables will be considered in the join operations. The unmatched records will overwrite the students\_courses\_details table.**
* **These join operations are stateful, meaning that they will wait for unmatched records to be added to the source tables prior to calculating the results.**

**Explanation**

The query reads three static Delta tables using spark.table() function, which means that all records in the current version of these tables will be read and considered in the join operations.

There is no difference between spark.table() and spark.read.table() function. Actually, spark.read.table() internally calls spark.table().

The pyspark.sql.DataFrame.join() function performs inner join operation by default, so the matched records will be written to the target table. In our case, the query writes the data in mode “overwrite” to the target table, which completely overwrites the table.

Reference:

<https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrameReader.table.html>

<https://spark.apache.org/docs/3.1.2/api/python/reference/api/pyspark.sql.DataFrame.join.html>

## Question 94:

The data engineering team has a large Delta Lake table named ‘user\_posts’ which is partitioned over the ‘year’ column. The table is used as an input streaming source in a streaming job. The streaming query is displayed below with a blank:

spark.readStream

.table("user\_posts")

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

.groupBy(

"post\_category", "post\_date")

.agg(

count("psot\_id").alias("posts\_count"),

sum("likes").alias("total\_likes"))

.writeStream

.option("checkpointLocation", "dbfs:/path/checkpoint")

.table("psots\_stats")

They want to remove previous 2 years data from the table without breaking the append-only requirement of streaming sources.

Which option correctly fills in the blank to enable stream processing from the table after deleting the partitions ?

* **.withWatermark("year", "INTERVAL 2 YEARS")**
* **.window("year", "INTERVAL 2 YEARS")**
* **.option(“year”, "ignoreDeletes")**
* **.option("ignoreDeletes", “year”)**
* **.option("ignoreDeletes", True)**

**Explanation**

Partitioning on datetime columns can be leveraged when removing data older than a certain age from the table. For example, you can decide to delete previous years data. In this case, file deletion will be cleanly along partition boundaries.

However, if you are using this table as a streaming source, deleting data breaks the append-only requirement of streaming sources, which makes the table no more streamable. To avoid this, you can use the ignoreDeletes option when streaming from this table. This option enables streaming processing from Delta tables with partition deletes.

option("ignoreDeletes", True)

Reference:

<https://docs.databricks.com/structured-streaming/delta-lake.html#ignore-updates-and-deletes>

## Question 95:

A data engineer created a new table along with a comment using the following query:

CREATE TABLE payments

COMMENT "This table contains sensitive information"

AS SELECT \* FROM bank\_transactions

Which of the following commands allows the data engineer to review the comment of the table ?

* **SHOW TABLES payments**
* **SHOW TBLPROPERTIES payments**
* **SHOW COMMENTS payments**
* **DESCRIBE TABLE payments**
* **DESCRIBE EXTENDED payments**

**Explanation**

DESCRIBE TABLE EXTENDED or simply DESCRIBE EXTENDED allows you to show none only table’s comment, but also columns’ comments, and other custom table properties

Reference:

<https://docs.databricks.com/sql/language-manual/sql-ref-syntax-aux-describe-table.html>

## Question 96

Which of the following commands allows data engineers to perform an insert-only merge?

* **MERGE INTO orders**

**USING new\_orders**

**ON orders.orders\_id = new\_orders.orders\_id**

**WHEN MATCHED**

**INSERT \***

* **MERGE INTO orders**

**USING new\_orders**

**ON orders.orders\_id = new\_orders.orders\_id**

**WHEN NOT MATCHED**

**INSERT \***

* **MERGE INTO orders**

**USING new\_orders**

**ON orders.orders\_id = new\_orders.orders\_id**

**WHEN MATCHED**

**INSERT \***

**WHEN NOT MATCHED**

**IGNORE \***

* **MERGE INTO orders**

**USING new\_orders**

**ON orders.orders\_id = new\_orders.orders\_id**

**WHEN NOT MATCHED**

**INSERT \***

**WHEN MATCHED**

**IGNORE \***

* **MERGE INTO orders**

**USING new\_orders**

**ON orders.orders\_id = new\_orders.orders\_id**

**WHEN NOT MATCHED**

**INSERT \***

**ELSE IGNORE \***

**Explanation**

The syntax for insert-only merge:

MERGE INTO target\_table

USING soruce\_table

ON merge\_condition

WHEN NOT MATCHED

INSERT \*

You just need to specify the NOT MATCHED clause, which inserts a row when a source row does not match any target row based on the merge\_condition (merge keys). Records that have the same keys as an existing record in the table will be simply ignored.

Reference: <https://docs.databricks.com/sql/language-manual/delta-merge-into.html>

## Question 97

Which of the following is considered a limitation when using the MERGE INTO command ?

* **Merge can not be performed in streaming jobs unless it uses Watermarking**
* **Merge can not be performed if multiple source rows matched and attempted to modify the same target row in the table**
* **Merge can not be performed if single source row matched and attempted to modify the multiple target rows in the table**
* **Merge does not support records deletion. It supports only upsert operations.**
* **All the above are considered limitations of the MERGE INTO command**

**Explanation**

Merge operation can not be performed if multiple source rows matched and attempted to modify the same target row in the table. The result may be ambiguous as it is unclear which source row should be used to update or delete the matching target row.

For such an issue, you need to preprocess the source table to eliminate the possibility of multiple matches.

Reference:

<https://docs.databricks.com/error-messages/index.html#delta_multiple_source_row_matching_target_row_in_merge>

## Question 98

A data engineer is using a foreachBatch logic to upsert data in a target Delta table.

The function to be called at each new microbatch processing is displayed below with a blank:

def upsert\_data(microBatchDF, batch\_id):

microBatchDF.createOrReplaceTempView("sales\_microbatch")

sql\_query = """

MERGE INTO sales\_silver a

USING sales\_microbatch b

ON a.item\_id=b.item\_id AND a.item\_timestamp=b.item\_timestamp

WHEN NOT MATCHED THEN INSERT \*

"""

Which option correctly fills in the blank to execute the sql query in the function on a cluster with recent Databricks Runtime above 10.5 ?

* **spark.sql(sql\_query)**
* **batch\_id.sql(sql\_query)**
* **microBatchDF.sql(sql\_query)**
* **microBatchDF.sparkSession.sql(sql\_query)**
* **microBatchDF.\_jdf.sparkSession().sql(sql\_query)**

**Explanation**

Usually, we use spark.sq() function to run SQL queries. However, in this particular case, the spark session can not be accessed from within the microbatch process. Instead, we can access the local spark session from the microbatch dataframe.

For clusters with recent Databricks Runtime version above 10.5, the syntax to access the local spark session is:

microBatchDF.sparkSession.sql(sql\_query)

Reference: <https://docs.gcp.databricks.com/structured-streaming/delta-lake.html#language-python>

## Question 99

A data engineer has been asked to develop a nightly batch job for workforce productivity analytics. The job will process events of employees productivity of the previous day, and store the performance of each employee in the Delta table “employees\_performance“. The table has the following schema:

"date DATE, employee\_id STRING, rating DOUBLE"

The data engineering team wants data to be stored in the table with the ability to compare employees' performance across time.

Which of the following code blocks accomplishes this task ?

* **performance\_df.write.format("delta").saveAsTable("employees\_performance")**
* **performance\_df.write.mode("append").saveAsTable("employees\_performance")**
* **performance\_df.write.mode("overwrite").saveAsTable("employees\_performance")**
* **performance\_df.write.option("dateFormat", "yyyy-MM-dd").saveAsTable("employees\_performance")**
* **performance\_df.write.partitionBy("date").saveAsTable("employees\_performance")**

**Explanation**

**DataFrameWriter.mode** defines the writing behaviour when data or table already exists.

Options include:

* append: Append contents of the DataFrame to existing data.
* overwrite: Overwrite existing data.
* error or errorifexists: Throw an exception if data already exists.
* ignore: Silently ignore this operation if data already exists.

This errorifexists or error is the default save mode. If the table already exists, it will throw the error message **Error: pyspark.sql.utils.AnalysisException: table already exists**.

The "employees\_performance" table has a **date** column. So, in order to be able to compare employees' performance across time, each new batch of data with new date should be appended into the table using the append mode.

Reference:

<https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrameWriter.mode.html>

## Question 100

The data engineering team has a large Delta table named ‘user\_messages’ with the following schema:

msg\_id INT, user\_id INT, msg\_time TIMESTAMP, msg\_title STRING, msg\_body STRING

The msg\_body field represents user messages in free-form text. The table has a performance issue when it’s queried with filters on this field.

Which of the following could explain the reason for this performance issue ?

* **The table does not leverage file skipping because it's not partitioned on the msg\_body column.**
* **The table does not leverage file skipping because it's not optimized with Z-ORDER on the msg\_body column.**
* **The table does not leverage file skipping because Delta Lake statistics are uninformative for string fields with very high cardinality**
* **The table does not leverage file skipping because Delta Lake statistics are only captured on the first 3 columns in a table.**
* **The table does not leverage file skipping because Delta Lake statistics are not captured on columns of type STRING**

**Explanation**

Delta Lake automatically captures statistics in the transaction log for each added data file of the table. By default, Delta Lake collects the statistics on the first 32 columns of each table. However, statistics are generally uninformative for string fields with very high cardinality (such as free text fields).

Calculating statistics on free-form text fields (like user messages, product reviews, etc) can be time consuming. So, you need to omit these fields from statistics collection by setting them later in the schema after the first 32 columns.

Reference: <https://docs.databricks.com/delta/data-skipping.html>

## Question 101

The data engineering team has a pipeline that ingest Kafka source data into a Multiplex bronze table. This Delta table is partitioned based on the topic and month columns.

A new data engineer notices that the ‘user\_activity’ topic contains Personal Identifiable Information (PII) that needs to to be deleted every two months based on the company’s Service-Level Agreement (SLA).

Which statement describes how table partitioning can help to meet

this requirement ?

* **Table partitioning allows immediate files deletion without running VACUUM command**
* **Table partitioning allows delete queries to leverage partition boundaries.**
* **Table partitioning reduces query latency when deleting large data files**
* **Table partitioning does not allow to time travel the PII data after deletion**
* **None of the above statements is correct. Table partitioning can not help to meet the specified requirement.**

**Explanation**

Partitioning on datetime columns can be leveraged when removing data older than a certain age from the table. For example, you can decide to delete previous months data. In this case, file deletion will be cleanly along partition boundaries.

Similarly, data could be archived and backed up at partition boundaries to a cheaper storage tier. This drives a huge savings on cloud storage.

Reference: <https://delta.io/blog/2023-01-18-add-remove-partition-delta-lake/>

## Question 102

Given the following commands:

CREATE DATABASE db\_hr;

USE db\_hr;

CREATE TABLE employees;

In which of the following locations will the employees table be located?

* **dbfs:/user/hive/warehouse**
* **dbfs:/user/hive/warehouse/db\_hr.db**
* **dbfs:/user/hive/warehouse/db\_hr**
* **dbfs:/user/hive/databases/db\_hr.db**
* **More information is needed to determine the correct answer**

**Explanation**

Since we are creating the database here without specifying a location, the database will be created in the default warehouse directory under dbfs:/user/hive/warehouse. The database folder has the extension (.db)

And since we are creating the table also without specifying a location, the table becomes a managed table created under the database directory (in db\_hr.db folder)

Reference:<https://docs.databricks.com/sql/language-manual/sql-ref-syntax-ddl-create-schema.html>

## Question 103

The data engineering team has a dynamic view with following definition:

CREATE VIEW students\_vw AS

SELECT \* FROM students

WHERE

CASE

WHEN is\_member("instructors") THEN TRUE

ELSE is\_active IS FALSE

END

Which statement describes the results returned by querying this view?

* + **Members of the instructors group will only see the records of active students. While users that are not members of the specified group will only see the records of inactive students.**
  + **Members of the instructors group will only see the records of active students. While users that are not members of the specified group will see null values for the records of inactive students**
  + **Only members of the instructors group will see the records of all students no matter if they are active or not. While users that are not members of the specified group will only see the records of inactive students**
* **Only members of the instructors group will see the records of all students no matter if they are active or not. While users that are not members of the specified group will see null values for the records of inactive students**
* **Only members of the instructors group will see the records of all students no matter if they are active or not. While users that are not members of the specified group will see “REDACTED” values for the records of inactive students**

**Explanation**

Only members of the instructors group will have full access to the underlying data since the WHERE condition will be True for every record. On the other hand, users that are not members of the specified group will only be able to see records of students with active status = false.

Reference:

<https://docs.databricks.com/data-governance/table-acls/object-privileges.html#dynamic-view-functions>

## Question 104

Which of the following commands can a data engineer use to grant full permissions to the HR team on the table employees ?

* **GRANT FULL PRIVILEGES ON TABLE employees TO hr\_team**
* **GRANT FULL PRIVILEGES ON TABLE hr\_team TO employees**
* **GRANT ALL PRIVILEGES ON TABLE employees TO hr\_team**
* **GRANT ALL PRIVILEGES ON TABLE hr\_team TO employees**
* **GRANT SELECT, MODIFY, CREATE, READ\_METADATA ON TABLE employees TO hr\_team**

**Explanation**

ALL PRIVILEGES is used to grant full permissions on an object to a user or group of users. It is translated into all the below privileges:

* SELECT
* CREATE
* MODIFY
* USAGE
* READ\_METADATA

Reference: <https://docs.databricks.com/security/access-control/table-acls/object-privileges.html#privileges>

## Question 105

The data engineering team has a secret scope named “prod-scope” that contains sensitive secrets in a production workspace.

A data engineer in the team is writing a security and compliance documentation, and wants to explain who could use the secrets in this secret scope.

Which of the following roles is able to use the secrets in the specified secret scope ?

* **Workspace Administrators**
* **Secret creators**
* **Users with MANAGE permission on the secret scope**
* **Users with READ permission on the secret scope**
* **All the above are able to use the secrets in the secret scope**

**Explanation**

Administrators\*, secret creators, and users granted access permission can use Databricks secrets. The secret access permissions are as follows:

* MANAGE - Allowed to change ACLs, and read and write to this secret scope.
* WRITE - Allowed to read and write to this secret scope.
* READ - Allowed to read this secret scope and list what secrets are available.

Each permission level is a subset of the previous level’s permissions (that is, a principal with WRITE permission for a given scope can perform all actions that require READ permission).

\* Workspace administrators have MANAGE permissions to all secret scopes in the workspace.

Reference:

<https://docs.databricks.com/security/secrets/index.html>

<https://docs.databricks.com/security/auth-authz/access-control/secret-acl.html#permission-levels>

## Question 106

In Spark UI, which of the following SQL metrics is displayed on the query’s details page?

* + **Query duration**
  + **Query execution time**
  + **Succeeded Jobs**
  + **Spill size**
  + **Number of input rows**

**Explanation**

In Spark UI, the query’s details page displays general information about the query execution time, its duration, the list of associated jobs, and the query execution DAG.

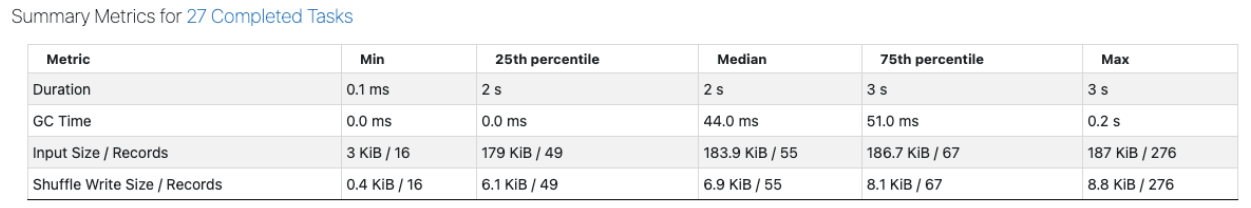
In addition, it shows SQL metrics in the block of physical operators. The SQL metrics can be useful when we want to dive into the execution details of each operator. For example, “number of output rows” can answer how many rows are output after a Filter operator, “Spill size” which is the number of bytes spilled to disk from memory in the operator.

Reference:

<https://spark.apache.org/docs/latest/web-ui.html>

## Question 107:

A data engineer is analyzing a Spark job via the Spark UI. They have the following summary metrics for 27 completed tasks in a particular stage

****

Which conclusion can the data engineer draw from the above statistics ?

* **All task are operating over partitions with even amounts of data**
* **All task are operating over empty or near empty partitions**
* **All tasks are operating over partitions with larger skewed amounts of data.**
* **Number of tasks are operating over partitions with larger skewed amounts of data.**
* **Number of tasks are operating over near empty partitions**

**Explanation**

Usually, if your computation was completely symmetric across tasks, you would see all of the statistics clustered tightly around the 50th percentile value.

Here, we see the distribution is reasonable, except that we have a bunch of “Min” values near zero. This suggests that we have almost empty partitions.

## Question 108

A data engineer has heard recently that users who have access to Databricks Secrets could be able to display the values of secrets in notebooks.

Which of the following could be a workaround to print the value of a Databricks secret in plain text ?

* **db\_password = dbutils.secrets.get("prod-scope", "db-password")**

**display(db\_password)**

* **db\_password = dbutils.secrets.get("prod-scope", "db-password", redacted=False)**

**print(db\_password)**

* **db\_password = dbutils.secrets.get("prod-scope", "db-password")**

**print(db\_password, redacted=False)**

* **db\_password = dbutils.secrets.get("prod-scope", "db-password")**

**for char in db\_password:**

**print(char)**

* **There is no workaround to print secrets values in plain text in notebooks. A string "REDACTED" will always be displayed when trying to print out a secret value.**

**Explanation**

Databricks redacts secret values that are read using dbutils.secrets.get(). When displayed in notebook cell output, the secret values are replaced with [REDACTED] string.

However, is there a workaround to print the values of Databricks secrets in plain text by Iterating through the secret and printing each character.

Reference:

<https://docs.databricks.com/security/secrets/index.html>

## Question 109

A data engineer wanted to create the job ‘process-sales’ using Databricks REST API.

However, they sent by mistake 2 POST requests to the endpoint ‘api/2.1/jobs/create’

Which statement describes the result of these requests ?

* **Only the first job will be created in the workspace. The second request will fail with an error indicating that a job named “process-sales” is already created.**
* **The second job will overwrite the previous one created using the first request.**
* **2 jobs will be created in the workspace, but the second one will be renamed to “process-sales (1)”**
* **2 jobs named “process-sales” will be created in the workspace with the same job\_id**
* **2 jobs named “process-sales” will be created in the workspace, but with different job\_id**

**Explanation**

Sending the same job definition in multiple POST requests to the endpoint ‘api/2.1/jobs/create’ will create a new job for each request, but each job will have its own unique job\_id.

Reference:

<https://docs.databricks.com/dev-tools/api/latest/jobs.html#operation/JobsCreate>

## Question 110

A data engineer wants to use Databricks REST API to retrieve the metadata of a job run using its run\_id.

Which of the following REST API calls achieves this requirement ?

* **Send POST request to the endpoint ‘api/2.1/jobs/runs/get’**
* **Send GET request to the endpoint ‘api/2.1/jobs/runs/get’**
* **Send POST request to the endpoint ‘api/2.1/jobs/runs/get-output’**
* **Send GET request to the endpoint ‘api/2.1/jobs/runs/get-output’**
* **Send GET request to the endpoint ‘api/2.1/jobs/runs/get-metadata’**

**Explanation**

Sending GET requests to the endpoint ‘/api/2.1/jobs/runs/get’ allows us to retrieve the metadata of a job run using its run\_id.

Reference:

<https://docs.databricks.com/dev-tools/api/latest/jobs.html#operation/JobsRunsGet>

## Question 111

Which of the following commands prints the current working directory of a notebook in Databricks Repos ?

* **%sh pwd**
* **print(sys.path)**
* **os.path.abspath()**
* **os.environ['PYTHONPATH']**
* **In Databricks Repos, the working directory of any notebook is ‘/databricks/driver’**

**Explanation**

The %sh magic command allows you to run shell code in a notebook.

The pwd is an acronym for print working directory.

Reference:

<https://docs.databricks.com/notebooks/notebooks-code.html#mix-languages>

<https://en.wikipedia.org/wiki/Pwd>

## Question 102

Which of the following establishes a Python file as a notebook in Databricks ?

* **The magic command %databricks on the first line of the file’s source code**
* **The comment ‘# Databricks notebook source’ on the first line of the file’s source code**
* **The import of the dbutils.notebook module in the file’s source code**
* **The creation of a spark session using SparkSession.builder.getOrCreate() in the file’s source code**
* **None of the above statements is correct. Notebooks are binary files with .pydb file extension.**

**Explanation**

You can convert Python, SQL, Scala, and R scripts to single-cell notebooks by adding a comment to the first cell of the file:

# Databricks notebook source

Reference:

<https://docs.databricks.com/notebooks/notebook-export-import.html#import-a-file-and-convert-it-to-a-notebook>

## Question 113

A data engineer wants to upload a CSV file from local system storage to the DBFS of a Databricks workspace. They have Databricks CLI already configured on the local system.

Which of the following Databricks CLI commands can the data engineer use to complete this task ?

* **workspace**
* **fs**
* **jobs**
* **configure**
* **libraries**

**Explanation**

Databricks CLI provides the ‘fs’ utility to interact with DBFS.

Usage:

1. databricks fs -h

The ‘fs cp’ command allows copying files to and from DBFS. To upload a csv file from a local system to DBFS, use:

1. databricks fs cp C:\source\file.csv dbfs:/target/file.csv --overwrite

Reference:

<https://docs.databricks.com/dev-tools/cli/dbfs-cli.html>

## Question 114

Which of the following statements correctly describes Unit Testing ?

* + **It’s an approach to simulate a user experience to ensure that the application can run properly under real-world scenarios**
  + **It’s an approach to test the interaction between subsystems of an application to ensure that modules work properly as a group.**
  + **It’s an approach to test individual units of code to determine whether they still work as expected if new changes are made to them in the future**
  + **It’s an approach to verify if each feature of the application works as per the business requirements**
  + **It’s an approach to measure the reliability, speed, scalability, and responsiveness of an application**

**Explanation**

Unit testing is an approach to testing units of code, such as functions. So, If you make any changes to them in the future, you can use unit tests to determine whether they still work as you expect them to.

Assertions are used in unit tests to check if certain assumptions remain true while you're developing your code.

assert func() == expected\_value

Reference:

<https://docs.databricks.com/notebooks/testing.html>

## Question 115

Which of the following statements correctly describes Integration Testing ?

* + **It’s an approach to simulate a user experience to ensure that the application can run properly under real-world scenarios**
  + **It’s an approach to test the interaction between subsystems of an application to ensure that modules work properly as a group.**
  + **It’s an approach to test individual units of code to determine whether they still work as expected if new changes are made to them in the future**
  + **It’s an approach to verify if each feature of the application works as per the business requirements**
  + **It’s an approach to measure the reliability, speed, scalability, and responsiveness of an application**

**Explanation**

Integration Testing is an approach to testing the interaction between subsystems of an application. It tests that the software modules are integrated logically and tested as a group.

Reference:

<https://en.wikipedia.org/wiki/Integration_testing>

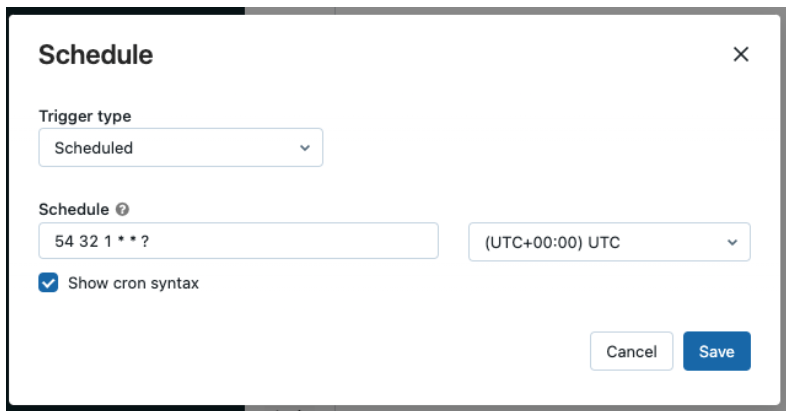
## Question 116

Which of the following describes Cron syntax in Databricks Jobs ?

* + **It’s an expression to represent the maximum concurrent runs of a job**
  + **It’s an expression to represent the retry policy of a job**
  + **It’s an expression to describe the email notification events (start, success, failure)**
  + **It’s an expression to represent the run timeout of a job**
  + **It’s an expression to represent complex job schedule that can be defined programmatically**

**Explanation**

To define a schedule for a Databricks job, you can either interactively specify the period and starting time, or write a Cron Syntax expression. The Cron Syntax allows to represent complex job schedule that can be defined programmatically



Reference:

<https://docs.databricks.com/workflows/jobs/schedule-jobs.html>

## Question 117

A data engineer has a Job with multiple tasks that takes more than 2 hours to complete. In the last run, the final task unexpectedly failed.

Which of the following actions can the data engineer perform to complete this run while minimizing the execution time ?

* **They can rerun this Job Run to execute all the tasks**
* **They can repair this Job Run so only the failed tasks will be re-executed**
* **They need to delete the failed Run, and start a new Run for the Job**
* **They can keep the failed Run, and simply start a new Run for the Job**
* **They can run the Job in Production mode which automatically retries execution in case of errors**

**Explanation**

You can repair failed multi-task jobs by running only the subset of unsuccessful tasks and any dependent tasks. Because successful tasks are not re-run, this feature reduces the time and resources required to recover from unsuccessful job runs.

Reference:

<https://docs.databricks.com/workflows/jobs/repair-job-failures.html>

## Question 118

As a general rule, before scheduling notebooks in production, which of the following commands should be removed from the code ?

* **Magic commands**
* **Overwrite table commands**
* **Markup language commands**
* **Display commands**
* **Import commands**

**Explanation**

Before scheduling notebooks in production, you may need to refactor your code. As a general rule, make sure you comment out:

* Unwanted files removal or tables dropping commands added during development
* Display actions or SQL queries added for debugging purposes

## Question 119:

Which of the following statements best describes the use of Python wheels in Databricks ?

* **A Python %wheel is a magic command allows to install Python packages on Databricks Clusters**
* **A Python wheel is a virtual environment for isolating the Python interpreter, libraries and modules in a notebook from other notebooks.**
* **A Python wheel is a repository for hosting, managing, and distributing Python binaries and artifacts in a Databricks workspace**
* **A Python wheel is a binary distribution format for installing custom Python code packages on Databricks Clusters**
* **A Python wheel is package installer tool alternative to ‘pip’**

**Explanation**

Python wheel is a binary distribution format for installing custom Python code packages on Databricks Clusters.

A wheel is a ZIP-format archive with the .whl extension.

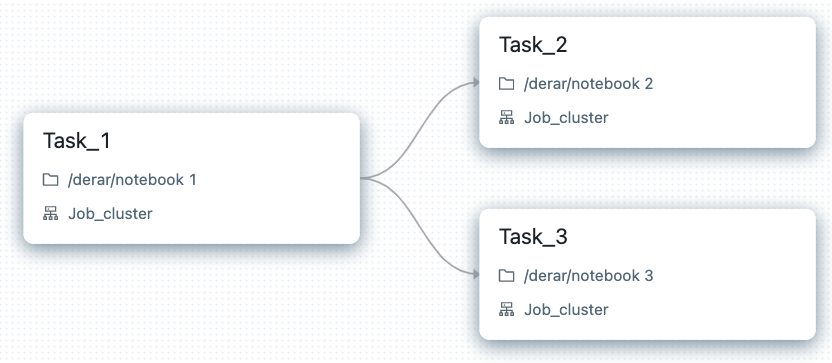
Reference:

<https://packaging.python.org/en/latest/specifications/binary-distribution-format/>

<https://docs.databricks.com/libraries/notebooks-python-libraries.html>

## Question 120

Given the following multi-task job

****

If there is an error in the notebook 1 that is associated with Task 1, which statement describes the run result of this job ?

* **Task 1 will partially fail. Tasks 2 and 3 will be skipped**
* **Task 1 will partially fail. Tasks 2 and 3 will run and succeed**
* **Task 1 will completely fail. Tasks 2 and 3 will be skipped**
* **Task 1 will completely fail. Tasks 2 and 3 will run and succeed**
* **All tasks will partially fail**

**Explanation**

If a task fails during a job run, all dependent tasks will be skipped.

The failure of a task will always be partial, which means that the operations in the notebook before the code failure will be successfully run and committed, while the operations after the code failure will be skipped.

Reference:

<https://docs.databricks.com/workflows/jobs/repair-job-failures.html>