Contents

[Question 1 2](#_Toc156624528)

[Question 2: 3](#_Toc156624529)

[Question 3: 4](#_Toc156624530)

[Question 4 5](#_Toc156624531)

[Question 5: 7](#_Toc156624532)

[Question 6: 8](#_Toc156624533)

[Question 7: 9](#_Toc156624534)

[Question 8: 10](#_Toc156624535)

[Question 9: 11](#_Toc156624536)

[Question 10: 12](#_Toc156624537)

[Question 11: 14](#_Toc156624538)

[Question 12: 15](#_Toc156624539)

[Question 13: 16](#_Toc156624540)

[Question 14: 17](#_Toc156624541)

[Question 15: 19](#_Toc156624542)

[Question 16: 20](#_Toc156624543)

[Question 17: 21](#_Toc156624544)

[Question 18: 22](#_Toc156624545)

[Question 19: 23](#_Toc156624546)

[Question 20: 25](#_Toc156624547)

[Question 21: 26](#_Toc156624548)

[Question 22: 29](#_Toc156624549)

[Question 23: 30](#_Toc156624550)

[Question 24: 31](#_Toc156624551)

[Question 25: 32](#_Toc156624552)

[Question 26: 35](#_Toc156624553)

[Question 27: 36](#_Toc156624554)

[Question 28: 37](#_Toc156624555)

[Question 29: 38](#_Toc156624556)

[Question 30: 39](#_Toc156624557)

[Question 31: 40](#_Toc156624558)

[Question 32: 41](#_Toc156624559)

[Question 33: 42](#_Toc156624560)

[Question 34: 43](#_Toc156624561)

[Question 35: 44](#_Toc156624562)

[Question 36: 45](#_Toc156624563)

[Question 37: 46](#_Toc156624564)

[Question 38: 47](#_Toc156624565)

[Question 39: 49](#_Toc156624566)

[Question 40 50](#_Toc156624567)

[Question 41 51](#_Toc156624568)

[Question 42: 52](#_Toc156624569)

[Question 44: 54](#_Toc156624570)

[Question 45: 55](#_Toc156624571)

[Question 46: 56](#_Toc156624572)

[Question 47: 57](#_Toc156624573)

[Question 48: 59](#_Toc156624574)

[Question 49: 60](#_Toc156624575)

[Question 50: 62](#_Toc156624576)

[Question 51: 64](#_Toc156624577)

[Question 52: 65](#_Toc156624578)

[Question 53: 67](#_Toc156624579)

[Question 54: 68](#_Toc156624580)

[Question 55: 69](#_Toc156624581)

[Question 56: 70](#_Toc156624582)

[Question 57: 71](#_Toc156624583)

[Question 58: 72](#_Toc156624584)

[Question 59: 73](#_Toc156624585)

[Question 60: 76](#_Toc156624586)

## Question 1

A data engineer found a CSV file that contained the records of the company’s old customers with some of the columns having PII data, as well. The data engineer wants to convert this file to a Delta table and runs the following statement:

**CONVERT TO DELTA csv.`path\_to\_csv`**

The above command fails to execute. What is the reason behind this failure?

* **csv.`path\_to\_csv` should be replaced with `path\_to\_csv`**
* **CONVERT TO DELTA is an invalid command.**
* **CONVERT TO DELTA can only be used with partitioned data.**
* **CONVERT TO DELTA can only be used with parquet data.**
* **Databricks restricts the conversion of CSV files with PII data to a Delta table.**

**Explanation**

**CONVERT TO DELTA** is an important command that can be used to **convert parquet file(s) to a Delta table**. This helps in accessing the data as a table and taking advantage of Delta tables.

If you try to convert a CSV-based table to a Delta table using **CONVERT TO TABLE**, you will get the following **error**:

**CONVERT TO DELTA only supports parquet tables, but you are trying to convert a csv source.**

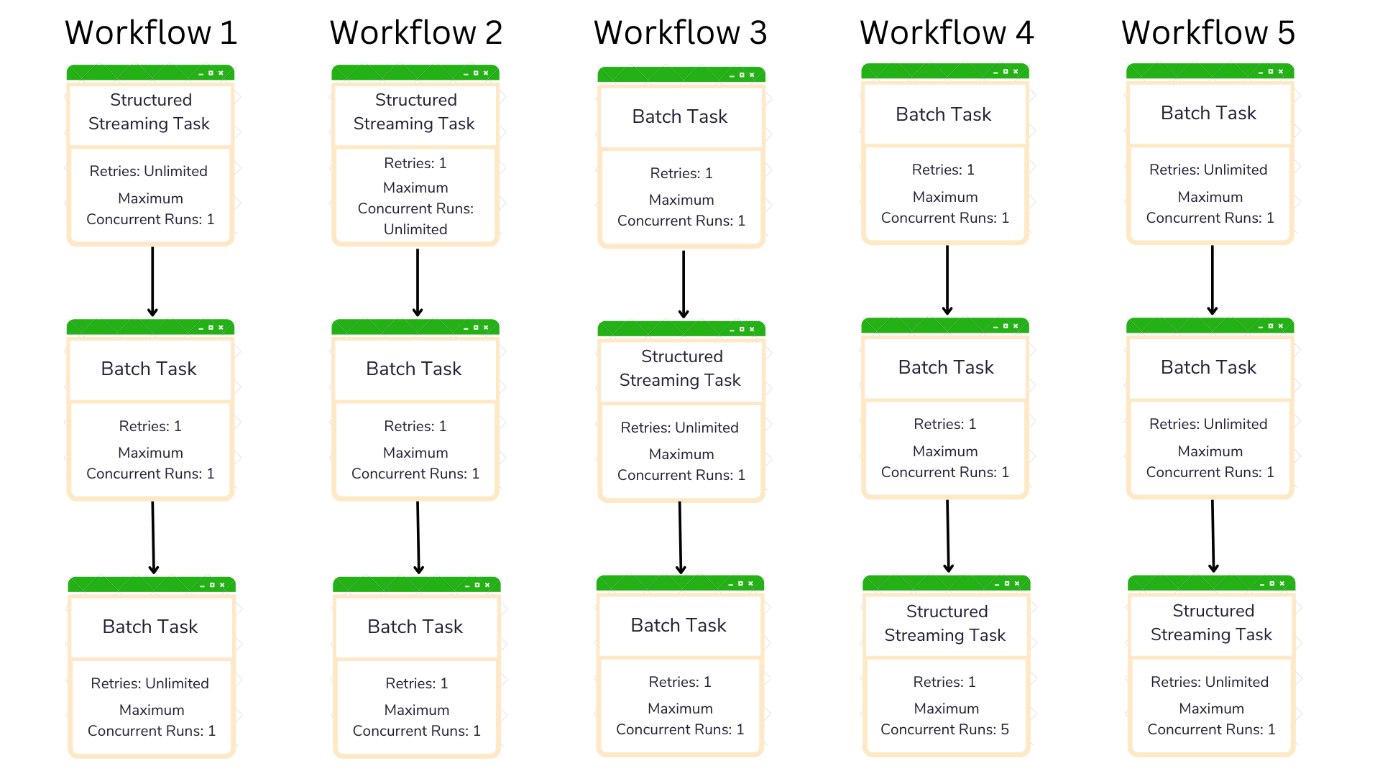
[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%201.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Using CONVERT TO DELTA in Databricks](https://docs.databricks.com/sql/language-manual/delta-convert-to-delta.html#:~:text=Either%20an%20optionally%20qualified%20table%20identifier%20or%20a%20path%20to%20a%20parquet%20or%20iceberg%20file%20directory.)

## Question 2:

A workflow consists of three tasks - 2 batch tasks and 1 structured streaming task. Which of the following workflows is best suited to complete the job while providing fault tolerance?

****

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

**Explanation**

Let us look at each workflow one by one:

The first task in **Workflow 1** and **Workflow 2** is a **structured streaming** task. Since the structured streaming task is **always running**(if not stopped manually), the other tasks dependent on this task **cannot** **start**. So, the first two workflows **cannot** **be** **completed**.

Similarly, in the **third workflow**, the **last task** is dependent on a **structured streaming task**, which makes the workflow **impossible**to complete.

In **Workflow** **4**, the streaming task is the **final task** of the workflow but the **number of retries is just 1**. As the question demands that the flow should be fault tolerant, the number of retries should be set as **unlimited**. Hence, **Workflow 5** ticks all the boxes.

Also note, according to Databricks, the number of **maximum concurrent runs** should always be **set as 1**in case of a **streaming** task.

More Info: [Placing a streaming task in Databricks Workflows](https://docs.databricks.com/workflows/jobs/jobs.html#:~:text=Since%20a%20streaming%20task%20runs%20continuously%2C%20it%20should%20always%20be%20the%20final%20task%20in%20a%20job)

## Question 3:

A Databricks engineer uses the following Databricks CLI command to start an already existing cluster but did not get any response.

**databricks clusters start --cluster-id 1198-132537-dht25rtr**

* **If the command is successful, no output is displayed.**
* **The cluster should be used in place of clusters**
* **Starting a cluster is the only unsupported operation using Databricks CLI.**
* **Cluster name should be used instead of cluster-id to start the cluster.**
* **Cluster name should also be added to the command to start the cluster.**

**Explanation**

Simply, if the command is successfully processed, there will be no output displayed.

More Info: [Starting a cluster using Databricks CLI](https://docs.databricks.com/dev-tools/cli/clusters-cli.html#:~:text=start%20%2D%2Dcluster%2Did%201234%2D567890%2Dbatch123-,If%20successful%2C%20no%20output%20is%20displayed,-.)

## Question 4

A data engineer wants to create a new job that intends to run a Python file located at **dbfs:/fetch\_matches.py** The python file accepts **year** and **month** as command-line arguments. These arguments are further used in the python file for extracting information about all the soccer matches played in that month and year. Assuming **matches** is an existing cluster having **1198-132537-dht25rtr** as its ID, which of the following JSON workloads should be posted to the **2.0/jobs/create endpoint** of Databricks REST API to create the job with the **year** as **2019** and **month** as **11**?

{

"name": "Get All Matches",

"existing\_cluster": "matches",

"python\_task": {

"python\_file": "dbfs:/fetch\_matches.py",

"parameters": ["2019", "11"]

}

}

{

"name": "Get All Matches",

"existing\_cluster\_id": "1198-132537-dht25rtr",

"spark\_python\_task": {

"python\_file": "dbfs:/fetch\_matches.py",

"arguments": "["2019", "11"]"

}

}

* {

"existing\_cluster\_id": "1198-132537-dht25rtr",

"spark\_python\_task": {

"python\_file": "dbfs:/fetch\_matches.py",

"parameters": ["2019", "11"]

}

}

* {

"existing\_cluster": "matches",

"spark\_python\_task": {

"python\_file\_path": "dbfs:/fetch\_matches.py",

"arguments": ["2019", "11"]

}

}

* {

"name": "Get All Matches",

"existing\_cluster\_id": "1198-132537-dht25rtr",

"python\_task": {

"python\_file\_path": "dbfs:/fetch\_matches.py",

"parameters": "["2019", "11"]"

}

}

**Explanation**

This question tests your **Databricks REST API** knowledge in detail. While creating a job through **2.0/jobs/create** endpoint, the following types of tasks are supported:

1. notebook\_task
2. spark\_jar\_task
3. spark\_python\_task
4. spark\_submit\_task
5. pipeline\_task

According to the question, you need to select **spark\_python\_task** to create the job with a **python task**. While creating a **spark\_python\_task**, the mandatory components of the JSON workload include the **path of the python file** and the **cluster details**. As you need to use an **already created cluster** for the job, the **ID of the cluster** is required(and **not** the name).

Correct code block:

{

"existing\_cluster\_id": "1198-132537-dht25rtr",

"spark\_python\_task": {

"python\_file": "dbfs:/fetch\_matches.py",

"parameters": ["2019", "11"]

}

}

**Exam Tip** - Also note, the **name of the job is optional**. If you do not provide the name, the job will be named **Untitled**, by default.

More Info: [Request structure to create a Job using Databricks REST API](https://docs.databricks.com/dev-tools/api/2.0/jobs.html#request-structure) | [Spark Python Task in a Job](https://docs.databricks.com/dev-tools/api/2.0/jobs.html#sparkpythontask:~:text=parameters-,An%20array%20of%20STRING,-Command%20line%20parameters)

## Question 5:

Which of the following languages is not supported on High concurrency clusters(*now known as Shared Clusters*)?

* **SQL and Scala**
* **Python and R**
* **Scala and R**
* **Scala and Python**
* **SQL and Python**

**Explanation**

Explanation: A **high concurrency cluster** cannot be used to run **R and Scala** code. To know the reason, you can refer to the below documentation.

More Info: [Why Scala is not supported in a High Concurrency Cluster](https://docs.databricks.com/archive/compute/configure.html#:~:text=High%20Concurrency%20clusters%20can%20run%20workloads%20developed%20in%20SQL%2C%20Python%2C%20and%20R.%20The%20performance%20and%20security%20of%20High%20Concurrency%20clusters%20is%20provided%20by%20running%20user%20code%20in%20separate%20processes%2C%20which%20is%20not%20possible%20in%20Scala.) | [Shared access mode limitations](https://docs.databricks.com/clusters/configure.html#shared-limitations)

## Question 6:

A data engineering team is working on a complex pipeline with trillions of rows in each table. They decide to persist in some of the frequently used DataFrames to fasten the processing of the queries. One of the data engineers ran the **persist()** command over the DataFrame and immediately checked the Spark UI’s Storage Tab but was unable to find the information about the persisted DataFrame. What could be the possible reason?

* **DataFrames persisted through persist() are not visible in Storage Tab, only the ones persisted through cache() are visible in the Storage Tab of Spark UI.**
* **The information about the persisted DataFrame can be found only in Ganglia metrics.**
* **The DataFrame information should have been visible in the Storage Tab of Spark UI immediately after running the persist() command. If the DataFrame information is not present, it means that the command was not executed successfully.**
* **Since persist() is lazily evaluated, performing an action on the cached DataFrame is necessary to view the results.**
* **persist() caches the DataFrame in memory which cannot be viewed in Spark UI.**

**Explanation**

Both **cache()** and **persist()** are **lazily evaluated** which means you would need to perform an **action** like **df.count()** or **df.show()** to view the cached DataFrame in **Spark UI**.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%206.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Persist() in Spark](https://spark.apache.org/docs/3.0.0-preview/api/python/pyspark.sql.html?highlight=cache#pyspark.sql.DataFrame.persist:~:text=Sets%20the%20storage%20level%20to%20persist%20the%20contents%20of%20the%20DataFrame%20across%20operations%20after%20the%20first%20time%20it%20is%20computed.)

## Question 7:

A scheduled production job consists of two tasks. The first task reads the data from the Kafka source while the second task loads the data to a mounted location after performing transformations. The data engineer tries to unmount the mount point by using **unmount()** command, in a new notebook, while the first task is still running. What will be the effect of using the above command?

* **The command will be executed immediately but the job will be completed without any error.**
* **The command will be executed immediately and the job will be canceled automatically as some of the tasks from the job are accessing the mount location.**
* **The command will wait until the job is completed.**
* **The command will be executed immediately and the first task will be completed while the second task will fail.**
* **The command will fail with an error message.**

**Explanation**

In the above scenario, the command will be **executed immediately**. As the first task is **not** using the **mount point,** it will be completed **without any errors** but the **second task** will run until the **reference to the mount point** is encountered. Since the **mount point** is now **unmounted**, the second task will **fail** with the following error message:

**Path does not exist**

More Info: [https://docs.databricks.com/dbfs/mounts.html#:~:text=Important-,Unmounting%20a%20mount%20point%20while%20jobs%20are%20running%20can%20lead%20to%20errors.%20Ensure%20that%20production%20jobs%20do%20not%20unmount%20storage%20as%20part%20of%20processing,-](https://docs.databricks.com/dbfs/mounts.html#:~:text=Important-,Unmounting%20a%20mount%20point%20while%20jobs%20are%20running%20can%20lead%20to%20errors.%20Ensure%20that%20production%20jobs%20do%20not%20unmount%20storage%20as%20part%20of%20processing,-.)

## Question 8:

A data engineer is assigned the task of creating a table using a **venues.csv** file stored at **dbfs:/FileStore/data/** The data engineer executes the following SQL statement and the table is created successfully.

CREATE TABLE venues

(name STRING, area INT)

USING CSV

LOCATION 'dbfs:/FileStore/data/'

Now, the data engineer tries to add a record to the table using **INSERT INTO** command. Which of the following would be the output of the **INSERT INTO** command?

* **The record will be inserted in the venues table and a new CSV file will be added in dbfs:/FileStore/data/ directory.**
* **The record will not be inserted in the table and an error message will be displayed.**
* **The record will be inserted in the table as well as the venues.csv file.**
* **The record will not be inserted in the table but an OK message will be displayed.**
* **The record will be inserted in the venues.csv file but not in the venues table.**

**Explanation**

As the table is created using **dbfs:/FileStore/data/** as **LOCATION**, new records **can be added** to the table and a **new CSV file** will be added to **dbfs:/FileStore/data/** directory.

However, if the table is created using **dbfs:/FileStore/data/venues.csv** as **LOCATION**, adding new records to the table will **result in an error**.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%208.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

## Question 9:

A Data Engineer has been assigned a task to take a pull of the **master** branch and create a new branch to work on the required changes. The data engineer has made all the changes and now wants to create a pull request in Databricks Repos. Which of the following Repos permissions should be granted to the data engineer to accomplish the tasks?

* **No permissions required**
* **Can Read permission**
* **Pull requests cannot be created using Databricks Repos**
* **Can Edit permission**
* **Can Manage permission**

**Explanation**

The **Databricks Repos** helps in integrating the notebooks with various **Git** providers like **GitHub**, **GitLab** and **Bitbucket**. Some of the tasks that **can be performed** using the **Databricks Repos** include **pushing** and **pulling** from a GitHubrepository, **creating** **branches** and **notebooks** etc.

However, some of the tasks **cannot** be performed in **Databricks Repos.** These tasks include **creation of a pull request**, **resolving merge conflicts** etc.

More Info: [Databricks Repos](https://docs.databricks.com/repos/index.html#:~:text=For%20following%20tasks,a%20pull%20request)

## Question 10:

Which of the following is a valid payload to create a new job using Databricks REST API?

* {

"name": "new\_job"

}

**{**

**"existing\_cluster\_id": "8522-150723-rmn4l016",**

**"notebook\_task": {**

**"notebook\_path": "path/to/notebook"**

**}**

**}**

* {

"name": "new\_job",

"notebook\_task": {

"notebook\_path": "path/to/notebook"

}

}

* **All three will return an error**
* **All three will be successful**

**Explanation**

Let’s look at the first three options:

{

“name”: “new\_job”

}

**INCORRECT!** This will return an **error** as the **cluster details** are **missing** from the payload. Either an **existing cluster ID** or the **new cluster details** need to be added to the payload.

{

"existing\_cluster\_id": "8522-150723-rmn4l016",

"notebook\_task": {

"notebook\_path": "path/to/notebook"

}

}

**CORRECT!** This payload will create a new job named **Untitled**. The job will have one **notebook task** which will run the notebook at **path/to/notebook** location. Also note, although the job will be created, it will need to be **manually** **triggered** as there is **no** schedule mentioned in the payload. To create a **scheduled job**, the payload should be updated to:

{

"existing\_cluster\_id": "8522-150723-rmn4l016",

"notebook\_task": {

"notebook\_path": "path/to/notebook"

},

"schedule": {

"quartz\_cron\_expression": "0 15 \* \* \* ?",

"timezone\_id": "IST"

}

}

{

"name": "new\_job",

"notebook\_task": {

"notebook\_path": "path/to/notebook"

}

}

**INCORRECT!** **Cluster details** are missing.

More Info: [Create a job using Databricks REST API](https://docs.databricks.com/api/workspace/jobs/create)

## Question 11:

Which of the following permission levels cannot be set while granting cluster permission to a group of users in Databricks?

* **Can Attach To**
* **Can Restart**
* **Can Manage**
* **Can Start**
* **No permissions**

**Explanation**

The 4 levels of permissions are as follows:

1. No permission
2. **Can Attach To**
3. **Can Restart**
4. **Can Manage**

Clearly, **Can Start** is not a valid permission for a cluster.

More Info: [Cluster Level permissions in Databricks](https://docs.databricks.com/security/access-control/cluster-acl.html#cluster-level-permissions)

## Question 12:

A team of data engineers is working on their respective notebooks attached to a common cluster. While creating the cluster, the admin updated the Spark config properties by adding **spark.sql.autoBroadcastJoinThreshold 100b** One of the data engineers runs the following command on their notebook named **show\_orders**:

**spark.conf.set("spark.sql.autoBroadcastJoinThreshold", '1024b')**

Another data engineer from the team, who is working on a notebook named **fetch\_results** runs the following commands:

1. %run ./show\_orders
2. spark.sparkContext.getConf().get("spark.sql.autoBroadcastJoinThreshold")

Assuming all commands run successfully, which of the following statements explains the output?

* **The output will be '1024b' as the config property defined in the notebook prevails over the config property set during the cluster creation.**
* **The output will be '100b' because the config property was changed in another notebook.**
* **The output will be '100b' as the config property defined during the cluster creation cannot be altered at the context level in the notebook.**
* **There will be no output because there is no print statement.**
* **The output will be None because the default value of the config property is None.**

**Explanation**

Once the **config property** is defined during the **cluster** **creation**, it **cannot** be altered at the **context** **level**. To change the already defined **config property** at the context level, you will need to **edit** the cluster’s spark properties using Cluster UI.

**IMPORTANT** - If you run **spark.conf.get("spark.sql.autoBroadcastJoinThreshold")** after running **spark.conf.set("spark.sql.autoBroadcastJoinThreshold", '1024b')** the result will be **'1024b'** as this change is made at the **notebook level**. But once you try to read the config property using **getConf()** method of  **sparkContext** (as asked in the question), the output will be **'100b'**

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2012.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Spark Config properties in Databricks Clusters](https://docs.databricks.com/clusters/configure.html#:~:text=volume%20types.-,Spark%20configuration,-To%20fine%20tune)

## Question 13:

Which of the following SQL statements can be used to create a Delta table with change data feed enabled?

* **CREATE TABLE versions (software string, version string) TBLPROPERTIES (delta.enableChangeDataCapture, True)**
* **Change Data Feed is auto-enabled for all the Delta tables.**
* **CREATE TABLE versions (software string, version string) TBLPROPERTIES (delta.enableChangeDataFeed = true)**
* **CREATE TABLE versions (software string, version string) PROPERTIES (delta.enableChangeDataCapture, True)**
* **CREATE TABLE versions (software string, version string) TABLE\_PROPERTIES (delta.changeDataFeedEnabled = true)**

**Explanation**

As **Change Data Feed** is **not** auto-enabled, it needs to be **enabled** for a Delta table **explicitly** by using **TBLPROPERTIES** keyword. It is used to add a property to a Delta table. For adding the **Change Data Capture** ability to a Delta table, you can add **delta.enableChangeDataFeed = true** in table properties.

Also note, the **CDC** **logs** will be available for only those records which are **inserted**, **updated** or **deleted** after this property is **enabled.**

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2013.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Enabling Change Data Feed for Delta tables](https://docs.databricks.com/delta/delta-change-data-feed.html#:~:text=You%20must%20explicitly,%3D%20true))

## Question 14:

A data engineer runs the following query to load the data in the **downstream** table.

spark.readStream.format('delta') \

.table('upstream') \

.writeStream \

.option('checkpointLocation', '/tmp/upDown') \

.format('delta') \

.trigger(once = True) \

.table('downstream')

After a few days, the **upstream** table was dropped and re-created as one of the columns needed to be removed to be compliant with the data protection rules. As new data arrived in the **upstream** table, the query was re-run but it failed. Which of the following changes should be done to the query to run it successfully keeping the historical data intact in the **downstream** table?

* **.drop('deleted\_column\_name') should be added to the query before writing the data to the downstream table.**
* **.option('mergeSchema', 'true') should be added to the query and .option('checkpointLocation', '/tmp/upDown') should be removed.**
* **.drop('deleted\_column\_name') should be added and the .option('checkpointLocation', '/tmp/upDown') should be removed.**
* **The checkpoint location should be changed.**
* **.outputMode('append') should be added to the streaming query.**

**Explanation**

Let us look at the options one by one:

***.drop(‘deleted\_column\_name')****should be added to the query before writing the data to the****downstream****table.*

**INCORRECT!** As the column is already **dropped** from the **upstream** table, adding **drop** operation will have no effect on the query.

***.option('mergeSchema', 'true')****should be added to the query and****.option('checkpointLocation', '/tmp/upDown')****should be removed.*

**INCORRECT!** A **checkpoint** is necessary for all **streaming** queries and hence, should **not** be removed.

***.drop('deleted\_column\_name')****should be added and the****.option('checkpointLocation', '/tmp/upDown')****should be removed.*

**INCORRECT!** As discussed earlier, adding **drop** operation will have **no** effect on the query as the column is already **deleted** from the **upstream** table.

*The checkpoint location should be changed.*

**CORRECT!** The checkpoint location **cannot** be the same as the **previous** query and needs to be **changed** as the table was **dropped** and **recreated**. Moreover, as the question demands, the **historical** data in the **downstream** table will remain **intact.**

***.outputMode('append')****should be added to the streaming query.*

**INCORRECT!** The **default** mode is **append**. So, adding this will have **no** effect on the functionality of the **query.**

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2014.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Checkpointing in Spark](https://docs.databricks.com/structured-streaming/async-checkpointing.html)

## Question 15:

A data engineer has been using a Delta table for the testing of one of the newly launched products. Two new data engineers need to add some more test cases to the delta table but the original table cannot be accessed by them. Both the data engineers request a copy of the original table. The first data engineer was given a copy of the original table using the **DEEP CLONE** while the second one got the copy using the **SHALLOW CLONE**. Which scenario explains the insertion of records by the data engineers into the copies of the original table?

* **The insertions made by the first data engineer will be reflected in the original table whereas the insertions made by the second data engineer will not affect the original table.**
* **Both the data engineers can add data to the copies of the original table received by them but the data will not be reflected back to the original table.**
* **The first data engineer can make the insertions to the table whereas the second data engineer cannot.**
* **Both the data engineers get a read-only copy of the original table and thus, cannot add data to the copies of the original table received by them.**
* **The insertions made by both the data engineers will be reflected in the original table.**

**Explanation**

Once the copies are created using **DEEP** and **SHALLOW** clone, the data engineers can **insert**, **update** or **delete** the **data** from their **copies.**

*Now the question arises if the insertions are reflected in the original table or not.*

The simple answer to this question is - **NO!**

So, the correct answer would be the **second** option which states that the data engineers can insert **records** in the copies they received but the original table will **not** be affected.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2015.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Delta table cloning in Databricks](https://docs.databricks.com/optimizations/clone.html#:~:text=Any%20changes%20made%20to%20either%20deep%20or%20shallow%20clones%20affect%20only%20the%20clones%20themselves%20and%20not%20the%20source%20table)

## Question 16:

Statistics need to be collected for the first 50 columns of a Delta table having 73 columns in total. Which of the following property must be altered in order to force statistics collection on the first 50 columns for future appends?

* **delta.dataSkippingNumIndexedCols**
* **spark.sql.delta.dataSkippingNumIndexedCols**
* **delta.dataSkippingCols**
* **spark.sql.dataSkippingNumIndexedCols**
* **The default value is 50; no need to change any property.**

**Explanation**

**Data Skipping** is an essential part of Delta tables and results in **faster** processing of **queries.**The default value for **delta.dataSkippingNumIndexedCols** is **32**, which means that the statistics data is collected for the **first** 32 columns only. Also, note that **increasing** this value will add to the **overhead** while writing the data to the Delta table.

More Info: [Data Skipping using Z-order](https://docs.databricks.com/delta/data-skipping.html#:~:text=You%20can%20change%20this%20value%20using%20the%20table%20property%20delta.dataSkippingNumIndexedCols)

## Question 17:

The following code intends to use Auto Loader for ingesting JSON files from a cloud location:

spark.readStream.format("cloudFiles") \

.schema(schema) \

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

.option("cloudFiles.schemaEvolutionMode", "addNewColumns") \

.load(source)

Which of the following is true if a file with an added column arrives at the source location?

* **The schema will evolve and the stream will continue to run.**
* **The stream will fail and the schema will not be evolved.**
* **The stream will continue and the new column will be ignored.**
* **The schema will be evolved but the stream will fail.**
* **The stream will fail and the new column is added to \_rescued\_data column.**

**Explanation**

Syntactically, adding **.option("cloudFiles.schemaEvolutionMode", "addNewColumns")** will not make any difference to the query as this is the **default** schema evolution mode in **Auto Loader.**

In the **addNewColumns**(default) mode, as soon as a **new** column is detected, the stream **fails** but the schema is **evolved.**

More Info: [Schema evolution in Auto Loader](https://docs.databricks.com/ingestion/auto-loader/schema.html#:~:text=addNewColumns%20(default),evolve%20data%20types)

## Question 18:

Which of the following locations will be used for the events log if the storage setting has not been set for the Delta Live Table pipeline?

* **/pipelines/log**
* **/pipelines/system/events**
* **/{pipeline-id}/logs**
* **/pipelines/{pipeline-id}/system/events**
* **/system/logs**

**Explanation**

If the storage location is **not** set for a **DLT pipeline,** the **events log** will be stored in the **/pipelines/{pipeline-id}/system/events** location.

More Info: [Default events log location for DLT](https://docs.databricks.com/workflows/delta-live-tables/delta-live-tables-event-log.html#:~:text=If%20you%20have%20not%20configured%20the%20storage%20setting%2C%20the%20default%20event%20log%20location%20is%20/pipelines/%3Cpipeline%2Did%3E/system/events)

## Question 19:

A data engineer is exploring **distinct()** and **dropDuplicates()** methods in spark to de-duplicate a DataFrame. Which of the following statements is correct for the transformations used for de-duplication?

* **The distinct() method accepts column names as arguments to remove duplicates based on certain column(s).**
* **In Databricks, only dropDuplicates() method is supported as the distinct() method is now deprecated.**
* **dropDuplicates() and drop\_duplicates() can be used interchangeably.**
* **Both distinct() and dropDuplicates() can be used to drop duplicates based on certain column(s).**
* **dropDuplicates() method can only be used on an RDD whereas distinct() method can be used only on a DataFrame.**

**Explanation**

Let us go through all the options one by one:

*The****distinct()****method accepts column names as arguments to remove duplicates based on certain column(s).*

**INCORRECT! distinct()** method does **not** accept any arguments. Applying **distinct()** method on a DataFrame will return another DataFrame with only **distinct** rows based on **all** the **columns** of the DataFrame.

*In Databricks, only****dropDuplicates()****method is supported as the****distinct()****method is now deprecated.*

**INCORRECT!**Both **distinct()** and **dropDuplicates()** can be used in Databricks. **None** of the two methods is **deprecated.**

***dropDuplicates()****and****drop\_duplicates()****can be used interchangeably.*

**CORRECT!**As per the **official** Spark documentation, **drop\_duplicates()** is an **alias** for the **dropDuplicates()**

*Both****distinct()****and****dropDuplicates()****can be used to drop duplicates based on certain column(s).*

**INCORRECT!** The **dropDuplicates()** method accept **column** names as **arguments**but **distinct()** method does **not.**

***dropDuplicates()****method can only be used on an RDD whereas****distinct()****method can be used only on a DataFrame.*

**INCORRECT!** The **distinct()** method can be used on both **RDDs** as well as **DataFrames**whereas **dropDuplicates()** can **only** be used on **DataFrames.**

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2019.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [dropDuplicates() method](https://spark.apache.org/docs/3.1.2/api/python/reference/api/pyspark.sql.DataFrame.dropDuplicates.html) | [drop\_duplicates() method](https://spark.apache.org/docs/3.1.2/api/python/reference/api/pyspark.sql.DataFrame.drop_duplicates.html) | [distinct() method on RDDs](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.RDD.distinct.html) | [distinct() method on DataFrames](https://spark.apache.org/docs/3.1.2/api/python/reference/api/pyspark.sql.DataFrame.distinct.html)

## Question 20:

The following query runs throughout the soccer matches to load data into the performance table:

spark.readStream.format('delta') \

.table('scorecard') \

.writeStream \

.option('checkpointLocation', '/tmp/scores') \

.format('delta') \

.table('performance')

While the query is running, one of the users accidentally drops the **performance** table. Which statement describes the effect of dropping the table?

* **The query will be stopped immediately.**
* **When the new data is loaded into the scorecard table, the performance table will be created again but the previous data will be lost from the performance table.**
* **When the new data is loaded into the scorecard table, the performance table will be created again and the previous data will also be restored automatically using the Delta logs.**
* **The query will be stopped only when the new data is added to the scorecard table.**
* **Dropping a table that is currently used in a streaming query is not possible.**

**Explanation**

If a **downstream** table is **dropped** while the streaming query is **running**, the query will **continue** to run till the **next** batch of data **arrives** in the **upstream** table. As soon as the data arrives in the **upstream** table, the query reads the data using the **readStream** method. Once the data is read from the table, it tries to perform the **writeStream** step. As the **downstream** table is not present the following error message will be returned:

**Table or view not found: spark\_catalog.default.performance;**

Examining the other options, the query will **not** be **stopped** immediately **nor** the downstream table will be **created** automatically.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2020.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Spark Streaming with Delta](https://docs.databricks.com/structured-streaming/delta-lake.html)

## Question 21:

Which of the following correctly depicts the output when the below set of commands is executed:

CREATE TABLE versions (software STRING, version STRING) TBLPROPERTIES (delta.enableChangeDataFeed = true);

INSERT INTO versions VALUES ('IDE', '6.2.0');

UPDATE versions SET version = '5.1.0' WHERE software = 'IDE';

INSERT INTO versions VALUES ('IDE-1', '1.3.0');

DELETE FROM versions WHERE version = '1.3.0';

SELECT max(\_commit\_version), max(version) FROM table\_changes('versions', 2);

* +-----------------------+---------------+

| max(\_commit\_version) | max(version) |

+-----------------------+---------------+

| 3 | 6.2.0 |

+-----------------------+---------------+

* **+-----------------------+---------------+**

**| max(\_commit\_version) | max(version) |**

**+-----------------------+---------------+**

**| 4 | 6.2.0 |**

**+-----------------------+---------------+**

* +-----------------------+---------------+

| max(\_commit\_version) | max(version) |

+-----------------------+---------------+

| 4 | 5.1.0 |

+-----------------------+---------------+

* +-----------------------+---------------+

| max(\_commit\_version) | max(version) |

+-----------------------+---------------+

| 3 | 1.3.0 |

+-----------------------+---------------+

* +-----------------------+---------------+

| max(\_commit\_version) | max(version) |

+-----------------------+---------------+

| 3 | 5.1.0 |

|  |  |
| --- | --- |
|  |  |

**Explanation**

Let us look at the value of the **\_commit\_version**after each statement:

***CREATE TABLE versions (software STRING, version STRING) TBLPROPERTIES (delta.enableChangeDataFeed = true);***

Once the table is created, **\_commit\_version** takes the value **0**

***INSERT INTO versions VALUES ('IDE', '6.2.0');***

After a record is inserted in the table, the **first** version of the table is created.

***UPDATE versions SET version = '5.1.0' WHERE software = 'IDE';***

Updating a record will add **another** version to the table.

***INSERT INTO versions VALUES ('IDE-1', '1.3.0');***

Inserting another record into the table will make the value of **\_commit\_version** equal to **3**

***DELETE FROM versions WHERE version = '1.3.0';***

Deleting a record from the table will add **another** version which makes the **maximum** value of the **\_commit\_version** to be **4**

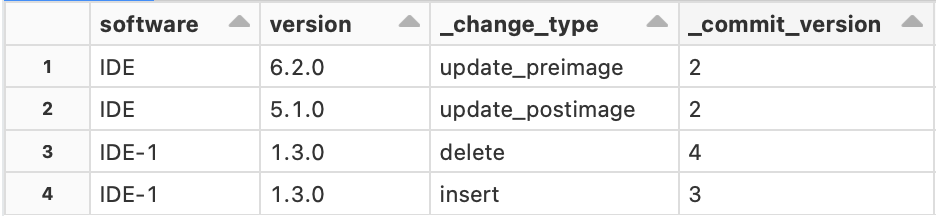
***SELECT max(\_commit\_version), max(version) FROM table\_changes('versions', 2)***

**max(\_commit\_version)** = 4

Before finding out the **max** value of the **version** column, reading about the [table\_changes](https://docs.databricks.com/sql/language-manual/functions/table_changes.html) function is highly suggested.

To find out the **maximum** value of the **version** column, let us look at the result of the following query:

***SELECT \* FROM table\_changes('versions', 2)***



The output contains all the **inserts** and **updates** starting from **version** **2** till the present version i.e. **4**

As you can see, **\_change\_type** column has a value - **update\_preimage**. This value is added to the table because the table stores **both** values for an updated record - the **updated** value as well as the **original** value. The updated value is identified by **update\_postimage** whereas the original value is identified by **update\_preimage** So, the maximum value of the **version** would be **6.2.0**

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2021.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Using \_change\_type column for Change Data Capture in Databricks](https://docs.databricks.com/delta/delta-change-data-feed.html#:~:text=_change_type,update_postimage%2C%20delete)

## Question 22:

Which of the following defines the difference between Z-ordering and bin-packing?

* **Z-ordering optimization tries to create similar-size files based on the number of rows whereas bin-packing optimization tries to create similar-size files based on their size on disk.**
* **Bin-packing optimization tries to create similar-size files based on the number of rows whereas Z-ordering optimization tries to create similar-size files based on their size on disk.**
* **Both bin-packing and Z-ordering optimization techniques try to create similar-size files based on their size on disk.**
* **Both bin-packing and Z-ordering optimization techniques try to create similar-size files based on the number of rows.**
* **Bin-packing optimization tries to create similar-size files based on the number of rows whereas Z-ordering optimization tries to create similar-size files based on the number of columns.**

**Explanation**

It is an **easy** picker for the exam.

**Z-ordering** - *Number of rows*

**Bin-packing** - *Size on disk*

More Info: [Bin packing vs Z-ordering](https://docs.databricks.com/sql/language-manual/delta-optimize.html#:~:text=Bin%2Dpacking%20optimization%20is%20idempotent,skew%20in%20optimize%20task%20times)

## Question 23:

A data engineer recently learned about the schema evolution modes in Databricks Auto Loader. Which of the following modes should be selected by them to ensure that the stream does not fail while ignoring the new columns?

* **dropNewColumns**
* **addNewColumns**
* **ignoreNewColumns**
* **none**
* **failOnNewColumns**

**Explanation**

Let us look at all the options one by one:

***dropNewColumns***

**INCORRECT!** This is **not** a valid schema evolution mode.

***addNewColumns***

**INCORRECT!** This is the **default** schema evolution mode. If you select this mode, **new** columns will be added to the **schema** but the stream would **fail.**

***ignoreNewColumns***

**INCORRECT!** This is an **invalid** mode.

***none***

**CORRECT!** If this mode is selected, the stream will **not** fail even if a **new** column is detected. The column will be **ignored** while the schema will **not** be evolved.

***failOnNewColumns***

**INCORRECT!** As the name suggests, the stream will **fail** if a **new** column is discovered.

More Info: [Auto Loader Schema evolution](https://docs.databricks.com/ingestion/auto-loader/schema.html#:~:text=none,to%20schema%20changes)

## Question 24:

Which of the following correctly depicts the usage of **withWatermark** method in a streaming job?

* **withWatermark is used for adding watermarks to the streaming tables for fault tolerance.**
* **Late arrival of data can be handled using withWatermark method.**
* **withWatermark enables the user to perform faster joins in a streaming application.**
* **Both late creation and the arrival of data can be handled using withWatermark method.**
* **To enable transformations on late and early-arriving data, withWatermark can be used.**

**Explanation**

withWatermark method can be used for handling **late-arriving** data in any **streaming** query.

More Info: [Handling late-arriving Data in Spark Streaming](https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html#:~:text=).count()-,Handling%20Late%20Data%20and%20Watermarking,-Now%20consider%20what)

## Question 25:

A data engineer performs the following always-on streaming query to update the downstream table **mobiles**

spark.readStream.format('delta') \

.option('readChangeFeed', 'true') \

.table('customers') \

.select('mobile') \

.writeStream \

.format('delta') \

.option('checkpointLocation', '/tmp/cust/\_checkpoints/') \

.table('mobiles')

Assuming that before starting the query, both the tables(**mobiles** and **customers**) were empty, what will be the number of records in the **mobiles** table after the execution of the following statements:

INSERT INTO customers VALUES ('Tom', ‘8635875’);

INSERT INTO customers VALUES ('Paul', ‘24984567');

INSERT INTO customers VALUES ('Ben', ‘79327492’);

UPDATE customers SET mobile = '8474738' WHERE name = ‘Ben’;

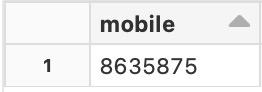
DELETE FROM customers WHERE mobile = ‘24984567’;

* **3 records**
* **4 records**
* **5 records**
* **6 records**
* **7 records**

**Explanation**

Let us look at the status of **mobiles** table after each statement:

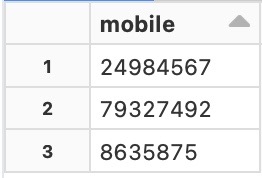
***INSERT INTO customers VALUES ('Tom', '8635875');***



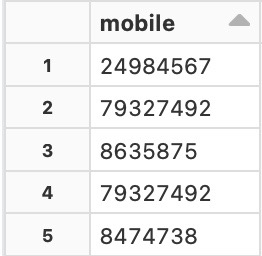
**INSERT INTO customers VALUES ('Paul', '24984567');**



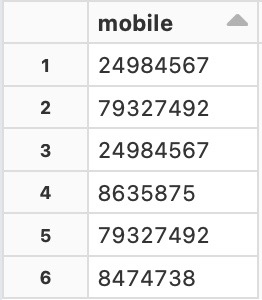
***INSERT INTO customers VALUES ('Ben', '79327492');***



***UPDATE customers SET mobile = '8474738' WHERE name = 'Ben';***



***DELETE FROM customers WHERE mobile = '24984567';***



As you can notice, each **insert** and **delete** statement adds **one** record to the table whereas the **update** command adds **2** records. This is because of the fact that there is **no** filter applied on the basis of the **\_change\_type** column in the query and when you **update** a record, the value **before** the update is also **preserved.**

Also note, when **UPDATE** or **DELETE** statement is executed, the **mobiles** table will have duplicates.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2025.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Using \_change\_type column for Change Data Capture in Databricks](https://docs.databricks.com/delta/delta-change-data-feed.html#:~:text=_change_type,update_postimage%2C%20delete)

## Question 26:

A data engineer is trying to access the transaction log for **ratings** table which contains weekly ratings for TV shows across the country. Which of the following folders will contain the logs for this table?

* **\_delta\_log**
* **\_ratings\_log**
* **\_transaction\_log**
* **\_log**
* **\_log\_ratings**

**Explanation**

Each **delta table** has a designated directory named **\_delta\_log** which stores the **transaction log**for the table. As soon as you **add**, **update** or **delete** records from the table, a new **JSON** file is added to the **\_delta\_log** directory.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2026.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Transaction log for Delta table](https://www.databricks.com/blog/2019/08/21/diving-into-delta-lake-unpacking-the-transaction-log.html#:~:text=When%20a%20user%20creates,%2C%20and%20so%20on)

## Question 27:

A streaming query has been started while the target table already contains some records. Which output mode would be selected for the query given that the output mode is not specified by the data engineer?

* **As the output mode is mandatory, the query will fail.**
* **The complete mode will be auto-selected.**
* **The default output mode i.e update mode will be selected.**
* **As the output mode is omitted, the append mode will be selected.**
* **The output mode depends on the type of data being loaded.**

**Explanation**

Output mode is specified using **.outputMode()** in a Spark **streaming** query. There are **three** types of **output** **modes** supported in a Spark Streaming query:

1. **Complete Mode:**Entire table is **replaced** with the **current** micro-batch of **data.**
2. **Append Mode:**Only the **new** records are **added** to the target table.
3. **Update Mode:** Only the rows **changed** since the last micro-batch of data are **updated** in the target table.

Also, note that the **default** mode is **Append** mode. So, if the output mode is not specified the query will **not** fail and the **append** mode will be auto-selected.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2027.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Output modes in Spark streaming](https://docs.databricks.com/getting-started/streaming.html#:~:text=Complete%20Mode%3A%20The,to%20Append%20mode) | [Default output mode in Spark streaming](https://docs.databricks.com/structured-streaming/delta-lake.html#:~:text=By%20default%2C%20streams%20run%20in%20append%20mode%2C%20which%20adds%20new%20records%20to%20the%20table)

## Question 28:

A SQL notebook that needs to be scheduled to run daily, has the following set of statements in one of its cells which tries to drop a table named **billing** if it exists, before creating and inserting some values into the table.

DROP TABLE billing;

CREATE TABLE billing (item string, quantity int, price float);

INSERT INTO billing VALUES ('DP', 5, 500.00);

What changes(if any) should be made to the above statements to ensure that the **DROP TABLE** statement should not throw any error?

* **The order of the DROP and CREATE statements should be reversed.**
* **DROP TABLE IF TABLE EXISTS should be used in place of DROP TABLE**
* **DROP TABLE statement should be replaced with DROP IF TABLE EXISTS**
* **No changes are required, the statements will be executed without any errors.**
* **DROP TABLE IF EXISTS should be used instead of DROP TABLE**

**Explanation**

In the above set of statements, if the table **billing** is already existing, the cell will be executed **without** any **errors**. Whereas to ensure that the cell will work as per the requirement, the **DROP** statement should be replaced with **DROP TABLE IF EXISTS**

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2028.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Dropping a table in SQL](https://docs.databricks.com/sql/language-manual/sql-ref-syntax-ddl-drop-table.html#:~:text=%2D%2D%20Assumes%20a%20table%20named%20%60employeetable%60%20does,DROP%20TABLE%20IF%20EXISTS%20employeetable%3B)

## Question 29:

A streaming application is using AutoLoader to ingest new files as they come in an S3 location. To infer the schema, AutoLoader uses the first 50 GB of data or the first 1000 files, whichever is lesser. Which of the following configurations should be changed to set the default value to 500 files for all future queries using AutoLoader?

* **spark.databricks.cloudFiles.schemaInference.sampleSize.numFiles**
* **spark.sql.cloudFiles.schemaInference.sampleFileSize.numBytes**
* **spark.sql.cloudFiles.schemaInference.sampleSize.numFiles**
* **spark.databricks.cloudFiles.schemaInference.sampleFileSize.numBytes**
* **spark.databricks.sql.cloudFiles.schemaInference.sampleSize.numFiles**

**Explanation**

To re-iterate the learnings from the question itself, **AutoLoader** uses the first **50 GB** or first **1000 files** to infer the **schema**, whichever condition is met **first.**

For example, if there are **400 files** with a collective file size of **50 GB**, AutoLoader will infer the **schema** from those 400 files. Alternatively, if there are a total of **3400 files** that collectively sizes **40 GB**, AutoLoader will infer the **schema** from the first **1000 files only.**

These **default** settings can be changed using spark **configurations.** There are **two** types of configurations that can be used to change the **sample** **size** to infer the **schema**:

1. **spark.databricks.cloudFiles.schemaInference.sampleSize.numBytes** - It can be used to set the **limit** on the total **size**, which is **50 GB** by default.
2. **spark.databricks.cloudFiles.schemaInference.sampleSize.numFiles** - This is what is asked in the question. It is used to **limit** the **number of files** to infer the **schema**, which is **1000** by default.

More Info: [Auto Loader schema inference](https://docs.databricks.com/ingestion/auto-loader/schema.html#:~:text=To%20change%20the,(integer))

## Question 30:

A data engineer is working on a project that includes a DLT(Delta Live Table). Which of the following is not supported when a DLT(Delta Live Table) is used with Python?

* **pivot() operation**
* **import statements**
* **Creation of views**
* **read() function**
* **Python decorators**

**Explanation**

Let us take a look at each option one by one:

***pivot()****operation*

**NOT SUPPORTED! pivot()** operation is **not** supported in a Delta Live Table.

***%pip****magic command*

**SUPPORTED!** **%pip** magic command is used for **installing** Python libraries.

*Creation of views*

**SUPPORTED!** Both **tables** and **views** can be created in a **DLT**(Delta Live Table)

***read()****function*

**SUPPORTED!** **read()** function is used to create a **view** or a **table** from already **existing** data.

*Python decorators*

**SUPPORTED!** Python decorators like **@dlt.view** and **@dlt.table** are used to declare **views** and **tables.**

More Info: [Limitations of a DLT](https://docs.databricks.com/workflows/delta-live-tables/delta-live-tables-python-ref.html#:~:text=The%20pivot()%20function%20is%20not%20supported.%20The%20pivot%20operation%20in%20Spark%20requires%20eager%20loading%20of%20input%20data%20to%20compute%20the%20schema%20of%20the%20output.%20This%20capability%20is%20not%20supported%20in%20Delta%20Live%20Tables)

## Question 31:

Which of the following is the highest level of abstraction in the Databricks Lakehouse relational model?

* **Catalog**
* **Database**
* **Table**
* **View**
* **Schema**

**Explanation**

**Database Lakehouse architecture** consists of **5**objects:

1. Catalog
2. Database/Schema
3. Table
4. View
5. Functions

The **catalog** sits at the **top** of the **Lakehouse relational model** i.e it is the **highest** level of **abstraction.**

More Info: [Catalog object in the Databricks Lakehouse architecture](https://docs.databricks.com/lakehouse/data-objects.html" \l ":~:text=A%20catalog%20is%20the%20highest%20abstraction%20(or%20coarsest%20grain)%20in%20the%20Databricks%20Lakehouse%20relational%20model)

## Question 32:

A stocks managing firm stores the stocks prices data in a delta table. The data engineer from the firm needs to create a relational entity that can provide data to the end users for only those stocks that fall under the technical category. The relational entity should have physical storage and the end users should be able to refresh the data, as and when required. Which of the following relational entities can be used by the data engineer?

* **A temporary view that should be re-created daily.**
* **A materialized view should be used as it can be refreshed by the end users, as and when required.**
* **A delta table should be used as it has physical storage.**
* **A DataFrame should be used as filter transformations can be applied to it.**
* **Access to the original table should be given to all the users.**

**Explanation**

A **materialized view** is a special type of view that has **physical storage.** Once a materialized view is registered, the query results are stored. Whenever the **REFRESH** command is executed over the materialized view, it **recalculates** its query results to incorporate any **modifications** or **updates** that have occurred in the upstream datasets.

In essence, the materialized view can be brought **up-to-date** either **manually** or by using the **CRON schedule**, ensuring that it accurately reflects the most recent changes and data available from its source datasets.

More Info: [Materialized Views in Databricks](https://docs.databricks.com/sql/language-manual/sql-ref-syntax-ddl-create-materialized-view.html#:~:text=A%20materialized%20view%20is%20a%20view%20where%20precomputed%20results%20are%20available%20for%20query%20and%20can%20be%20updated%20to%20reflect%20changes%20in%20the%20input.%20Each%20time%20a%20materialized%20view%20is%20refreshed%2C%20query%20results%20are%20recalculated%20to%20reflect%20changes%20in%20upstream%20datasets.)

## Question 33:

A database **employees** was created by a data engineer with the following command:

**CREATE DATABASE employees LOCATION '/employees'**

A junior data engineer who recently joined the firm executes the following command to create a database **new\_employees** for testing:

**CREATE DATABASE new\_employees LOCATION '/employees'**

After a few days of testing, the junior data engineer uses the **DROP** command to drop the **new\_employees** database. Which of the following statements is true about the data in both databases?

* **The new\_employees database cannot be created using the same location.**
* **Databases that share the same location cannot be dropped using the DROP command.**
* **Once the DROP command is successful, the data belonging to the new\_employees database will be deleted whereas data in the employees database remains intact.**
* **All the data belonging to both databases will be dropped.**
* **No data will be dropped from any database but the DROP command will be successful.**

**Explanation**

As, **both** the databases share the same **location** in DBFS, dropping **one** of the databases will drop all the data from **both** databases.

Also, note that the databases containing data i.e **non-empty** databases can only be **deleted** using the **CASCADE** command in conjunction with the **DROP** command:

**DROP DATABASE new\_employees CASCADE;**

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2033.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Sharing location across multiple databases](https://docs.databricks.com/lakehouse/data-objects.html#:~:text=To%20avoid%20accidentally,multiple%20database%20definitions)

## Question 34:

Which of the following commands can be used to list the partitions on a Delta table named **courses**?

* **SHOW PARTITIONS ON courses;**
* **SELECT partitions() from courses;**
* **SELECT get\_partitions from courses;**
* **SHOW PARTITIONS courses;**
* **SELECT partitions from unity\_catalog.courses;**

**Explanation**

The **syntax** to get the list of **partitions** on a Delta table is **SHOW PARTITIONS {table\_name}**

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2034.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [List partitions of a Delta table](https://docs.databricks.com/sql/language-manual/sql-ref-syntax-aux-show-partitions.html#:~:text=SHOW%20PARTITIONS%20table_name)

## Question 35:

A data engineer needs to create a function named **print\_details** that prints the name of the current catalog and database separated by a space. Which of the following statements can be used to create the function?

* **CREATE FUNCTION print\_details() RETURNS concat(current\_catalog(),' ',current\_database());**
* **CREATE FUNCTION print\_details(STRING) RETURNS concat(current\_catalog(),' ',current\_database());**
* **CREATE FUNCTION print\_details RETURNS STRING RETURN concat(current\_catalog(),' ',current\_database());**
* **CREATE FUNCTION print\_details(STRING) RETURN concat(current\_catalog(),' ',current\_database());**
* **CREATE FUNCTION print\_details() RETURNS STRING RETURN concat(current\_catalog(),' ',current\_database());**

**Explanation**

This is a **tricky** question as **RETURN** and **RETURNS** keywords create confusion. The **RETURNS** keyword is used to specify the **type of return** value like **string** and **int** whereas the **RETURN** keyword is used to print the **actual** return value. Also, note that the **name** of the **function** should be **appended** with the **empty brackets** if the function does **not** accept any **arguments** i.e **print\_details()**

After knowing the above stuff, you can easily answer these types of questions in the actual exam.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2035.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Crete a user-defined SQL function](https://docs.databricks.com/sql/language-manual/sql-ref-syntax-ddl-create-sql-function.html) | [current\_catalog() function](https://docs.databricks.com/sql/language-manual/functions/current_catalog.html) | [currrent\_database() function](https://docs.databricks.com/sql/language-manual/functions/current_database.html)

## Question 36:

Two data engineers tries to print the **defects\_df** DataFrame using **show()** and **display()** methods respectively. Which of the following statements describes the difference between the type of output format displayed for both the data engineers.

* **The show() method displays the DataFrame in a tabular format but the display() method prints only the column names without any data.**
* **The display() method can be used to visualize the DataFrame in the form of charts, graphs etc.**
* **The show() method can only be used in Databricks.**
* **Running the display() method converts a DataFrame to an RDD.**
* **Both B and D are true.**

**Explanation**

Both **show()** and **display()** methods can be used to **print** a **DataFrame** in **Databricks**. While working in Databricks, the **display()** method enables **visualization** over a DataFrame whereas the **show()** method does not.

Also note, the **show()** method supports arguments like **truncate** and **vertical** to view the data in different forms.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2036.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [display() method in Databricks](https://www.databricks.com/spark/getting-started-with-apache-spark/dataframes#:~:text=An%20additional%20benefit%20of%20using%20the%20Databricks%C2%A0display()%C2%A0command%20is%20that%20you%20can%20quickly%20view%20this%20data%20with%20a%20number%20of%20embedded%20visualizations) | [show() method in Spark](https://spark.apache.org/docs/3.2.0/api/python/reference/api/pyspark.sql.DataFrame.show.html)

## Question 37:

A data engineer executes the following command to add a **NOT NULL** constraint on one of the columns of a Delta table that already had null values.

**ALTER TABLE universities ALTER COLUMN location SET NOT NULL;**

Which of the following would be the outcome if after executing the above command, the data engineer tries to add another null value in the **location** column of the Delta table?

* **The ALTER TABLE command will fail but new NULLS cannot be added to the location column.**
* **Since the ALTER TABLE command will return an error, new NULLS can be added to the location column.**
* **The ALTER TABLE command will be successfully executed but new NULLS can still be added to the location column.**
* **The ALTER TABLE command will be successful and no new NULLS will be accepted in the location column.**
* **ALTER TABLE command will be successful and all the previous and new NULL values will be dropped.**

**Explanation**

As the **location** column already has **NULL** values, the **ALTER TABLE** command will return an **error** and the **NOT NULL** constraint will **not** be added to the column. It means that you can still add **NULL** values in the column.

Also note, every time a constraint is added to an **existing** table, the already existing values should contain **valid** values. If the already existing data is **not** **valid**, the constraint **cannot be added** to the table.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2037.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Adding a NOT NULL constraint](https://docs.databricks.com/tables/constraints.html#:~:text=Before%20adding%20a%20NOT%20NULL%20constraint%20to%20a%20table%2C%20Databricks%20verifies%20that%20all%20existing%20rows%20satisfy%20the%20constraint)

## Question 38:

Which of the following correctly explains the outcome if the below set of statements are executed in a Databricks notebook, assuming **viewer** is a pre-defined delta table with columns like **name** and **age**?

CREATE VIEW viewing AS SELECT age FROM viewer;

CREATE TEMP VIEW viewing AS SELECT name FROM viewer;

CREATE VIEW viewing AS SELECT age FROM viewer;

SELECT \* FROM viewing;

* **All the commands will be executed and the age column will be shown as output.**
* **Only the third command will fail and the name column will be shown as the output.**
* **All the commands will be executed and the name column will be shown as the output.**
* **The second and the third commands will fail and the age column will be shown as the output.**
* **All the commands will be executed but nothing will be shown as the output.**

**Explanation**

This is a tricky but interesting question to prepare you for the exam. Let us see what would happen after each query is executed.

***CREATE VIEW viewing AS SELECT age FROM viewer;***

The command will be executed **without** any errors and a view named **viewing** will be registered.

***CREATE TEMP VIEW viewing AS SELECT name FROM viewer;***

This command will also be executed **without** any errors and a **temporary** view named **viewing** will be registered.

***CREATE VIEW viewing AS SELECT age FROM viewer;***

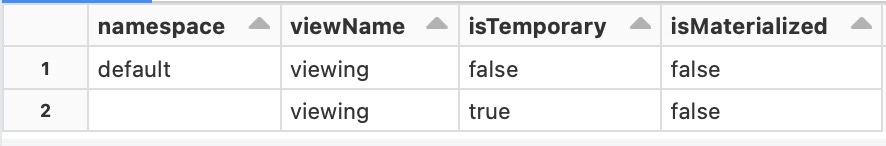
This command will **fail** as a view named **viewing** has **already** been registered after the execution of the **first** command.

***SELECT \* FROM viewing;***

This command will output the **name** column and **not** the **age** column. To get the **age** column as output, you will need to append the **database** **name** to the view i.e. **SELECT \* FROM db\_name.viewing;**

Also note, to get the names of **all** the views, you can use **SHOW VIEWS;** It prints **all** the views currently **active** including the **name of the database** to which the view is currently **tied.**

In the above question, if **SHOW VIEWS** is run, the output will be:



[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%238.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Create View in Databricks](https://docs.databricks.com/sql/language-manual/sql-ref-syntax-ddl-create-view.html)

## Question 39:

An organization stores the salary of its employees in the **emp\_sal** delta table. The requirement is to store the last salary as well as the current salary of all the employees in the table without increasing the number of rows of the table. Which of the following type of table can be selected to incorporate the requirement?

* **Type 0 SCD**
* **Type 1 SCD**
* **Type 2 SCD**
* **Type 3 SCD**
* **Type 4 SCD**

**Explanation**

Three of the most commonly used types of **Slowly Changing Dimension(SCD)** are:

1. ***Type 1 SCD*** - The **old** record is **overwritten** by the **new** record.
2. ***Type 2 SCD*** - The **new** record is **appended** to the table whereas the **old** record is set to **not-active** either by using the **end\_date** or **active** column.
3. ***Type 3 SCD*** - Changes are tracked using **column** **addition**. A **new** column is **added** for the **current** value while the **original** value is **retained**, as well.

In this question, as the number of rows should **not** increase and the **previous** record should also be **retained**, **Type 3 SCD** can be used to fulfill the requirement which **adds** another column **current\_salary** which can be used for storing the **current salary** whereas the **original** column stores the **previous salary.**

More Info: [Type 3 SCD](https://docs.oracle.com/cd/E41507_01/epm91pbr3/eng/epm/phcw/concept_UnderstandingSlowlyChangingDimensions-405719.html#:~:text=that%20are%20tracked.-,Type%203%20Slowly%20Changing%20Dimensions,must%20be%20retained%20as%20part%20of%20the%20record%2C%20usually%20for%20reporting,-.)

## Question 40

As part of a proof of concept, a data engineer is working with the Olympics dataset. The data is first filtered by limiting the records to only those participants who won the gold medals at least once in the games. The size of the delta table has now decreased to 50 MB. To enable faster reads from the table, the data engineer has partitioned the table on the **year** column making each partition approximately 2 MB in size. Which of the following will enable even faster reads from the table?

* **The partitioning column should be changed to athlete\_name column to decrease the size of each partition significantly.**
* **As the size of the table is just 50 MB, the partitioning should be removed.**
* **The data should be stored in CSV to enable faster reads.**
* **The table should be queried by the admin to increase the speed of the queries.**
* **The table should be partitioned on 2 columns i.e. year and athlete\_name**

**Explanation**

Adding **partitions** to a **small** delta table like the one given in the question will **decrease** the table’s **performance.** As the **size** of the table is just **50 MB,** the partitions should be **removed** from the table. This will enable even **faster** reads, as asked in the question.

More Info: [Adding partitions to a Delta table](https://docs.databricks.com/tables/partitions.html#:~:text=Because%20of%20built%2Din%20features%20and%20optimizations%2C%20most%20tables%20with%20less%20than%201%20TB%20of%20data%20do%20not%20require%20partitions)

## Question 41

In which of the following hops of the medallion or multi-hop architecture, aggregation is the most common transformation?

* **Raw-Bronze**
* **Bronze-Silver**
* **Silver-Gold**
* **Gold-Bronze**
* **Raw-Silver**

**Explanation**

This is an easy picker as you might know that the **Gold** layer is the one that has Delta tables that are **aggregated.** So, the **Silver to Gold layer hop** would mostly include **aggregations.**

More Info: [Medallion Architecture - Gold Layer](https://docs.databricks.com/lakehouse/medallion.html#:~:text=This%20gold%20data%20is%20often%20highly%20refined%20and%20aggregated%2C%20containing%20data%20that%20powers%20analytics%2C%20machine%20learning%2C%20and%20production%20applications)

## Question 42:

A Delta table needs to be joined with a lookup table to add a column. Which of the following techniques can be used by the data engineer to fasten the process of column addition, knowing that the size of the lookup table is very small as compared to the other table?

* **The union method should be used instead of join to add the column.**
* **A UDF should be created to add the column.**
* **A full outer join should be used to speed up the process as the outer join is always optimized.**
* **The code should be converted to Python to increase the speed of the join.**
* **The lookup table should be broadcasted using the broadcast method.**

**Explanation**

In Spark, if you need to join a **smaller** table/DataFrame with a relatively **bigger** table/DataFrame, **broadcast** join should be used to **increase** the **performance** of the join.

By **broadcasting** the **smaller** table/DataFrame, a **copy** of the table/DataFrame is **broadcasted** to all the **executors** within the cluster. This helps in **decreasing** the **shuffles** and thus **increasing** the **speed** of the joins.

More Info: [Broadcast joins in Spark](https://docs.databricks.com/sql/language-manual/sql-ref-syntax-qry-select-hints.html#:~:text=Use%20broadcast%20join.%20The%20join%20side%20with%20the%20hint%20is%20broadcast%20regardless%20of%20autoBroadcastJoinThreshold.%20If%20both%20sides%20of%20the%20join%20have%20the%20broadcast%20hints%2C%20the%20one%20with%20the%20smaller%20size%20(based%20on%20stats)%20is%20broadcast)

Bottom of Form

Top of Form

Question 43:

Which of the following libraries can be used to encrypt PII data in PySpark?

* **PyProtect**
* **Fernet**
* **PyLint**
* **PyNet**
* **FerProtect**

**Explanation**

The **Fernet** library can be used to **encrypt** columns from a DataFrame. The columns can then be **decrypted** using the **encryption keys.** The encryption keys are shared with only those users who are authorized to view the **PII**data.

More Info: [Securing PII data through Fernet Library](https://www.databricks.com/blog/2020/11/20/enforcing-column-level-encryption-and-avoiding-data-duplication-with-pii.html)

## Question 44:

A new member has recently been added to a team of developers. The new member wants to run an existing notebook using **%run** magic command in their newly created notebook. What is the minimum notebook-level permission that can be granted to the new member allowing them to run the existing notebook?

* **No permissions are required**
* **Can Read permission**
* **Can Run permission**
* **Can Edit permission**
* **Can Mange permission**

**Explanation**

There are **5** levels of permissions for a **notebook** in Databricks:

1. No permissions
2. **Can Read** - Can **view the cells** of the notebook, **add comments** and **run the notebook** using the **%run** magic command or the notebook workflows.
3. **Can Run** - All the permissions stated in **Can Read** plus can **attach** or **detach** notebooks from a cluster and **run** **the** **cells** of the notebook.
4. **Can Edit** - All the permissions in **Can Run** and can also **edit the cells.**
5. **Can Manage** - All the permissions stated in **Can Edit** and also the ability to **change the permissions** of the notebook.

So, the data engineer with **Can Manage** permission can **change the permissions** on the notebook while the other one **cannot** do the same.

More Info: [Notebook-level permissions in Databricks](https://docs.databricks.com/security/access-control/workspace-acl.html#:~:text=You%20can%20assign%20five%20permission%20levels%20to%20notebooks%3A%20No%20Permissions%2C%20Can%20Read%2C%20Can%20Run%2C%20Can%20Edit%2C%20and%20Can%20Manage.%20The%20table%20lists%20the%20abilities%20for%20each%20permission)

## Question 45:

To comply with CCPA and GDPR, the company needs to delete the PII data. Each of the Delta tables used for the data storage contains both PII and non-PII data. Which of the following techniques can be used by the company to get rid of the PII data without losing the ability to perform statistical analysis on the historical data?

* **Perform DELETE operation followed by VACUUM operation on all the Delta tables.**
* **Drop the PII columns from the Delta tables and add their data in a shared location for the compliance team.**
* **Use ACLs to set permissions on different columns.**
* **Anonymize data in PII columns.**
* **Do not delete the PII data from the tables, as it can result in the non-performance of SQL queries over the Delta tables.**

**Explanation**

According to the question, the ability to perform the **statistical** **analysis** on the **historical** data should **not** be lost. So, the best option for the company is to **anonymize** the data in the PII columns. In anonymization, the data is **transformed** in a way that it **cannot** be recreated again.

More Info: [Anonymizing PII Data](https://www.databricks.com/blog/2022/03/23/implementing-the-gdpr-right-to-be-forgotten-in-delta-lake.html#:~:text=Approach%202%20%2D%20Anonymization)

## Question 46:

Which of the following permission levels can be set on Secrets in Databricks?

* **READ, WRITE and MANAGE**
* **WRITE and MANAGE**
* **READ, WRITE and EXECUTE**
* **READ, CREATE, WRITE and MANAGE**
* **CREATE, WRITE and MANAGE**

**Explanation**

The following set of permissions can be set on Secrets in Databricks:

1. READ - **Read** and **List** secrets in the scope.
2. WRITE - All READ permissions including **writing** secrets to the scope.
3. MANAGE - All WRITE permissions and the ability to **change** permissions.

More Info: [Access Control on Secrets in Databricks](https://docs.databricks.com/security/access-control/secret-acl.html#:~:text=The%20secret%20access,secrets%20are%20available)

## Question 47:

The following code intends to create a view that has 2 columns - **price(int)** and **quantity(int)**, where the **price** column should be visible to all the users who are members of the **auditor** group but not the **compliance** group. Which of the following options will complete the code?

CREATE VIEW shares\_view AS

SELECT

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

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

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

quantity

FROM shares;

* CASE WHEN

isMember(‘auditor’) and isNotMember(‘compliance’) THEN price

ELSE 0

END AS price,

* CASE WHEN

is\_member(‘auditor’) and is\_not\_member(‘compliance’) THEN price

ELSE 0

END AS price,

* CASE WHEN

isMember(‘auditor’) and NOT isMember(‘compliance’) THEN price

ELSE 0

END AS price,

* CASE WHEN

is\_member(‘auditor’) and NOT is\_member(‘compliance’) THEN price

ELSE 0

END AS price,

* CASE WHEN

member(‘auditor’) and not\_member(‘compliance’) THEN price

ELSE 0

END AS price,

**Explanation**

This question checks your knowledge of **column-level permissions** on the **dynamic views.** Databricks supports two dynamic views functions:

1. **current\_user()**
2. **is\_member()**
3. **is\_account\_group\_member()**

**current\_user()** gives you the name of the **current user** whereas **is\_member(group)** checks if the **current user**is a member of the **account-level group**. It returns **TRUE** if the **current user** is a member of the **account-level group**.

**member()**, **isMember()**, **isNotMember()** and **is\_not\_member()** are some of the **invalid** methods used in the options to create **confusion** and prepare you for the **actual** **exam.**

So, the correct code block should be:

CREATE VIEW shares\_view AS

SELECT

CASE WHEN

is\_member('auditor') and NOT is\_member('compliance') THEN price

ELSE 0

END AS price,

quantity

FROM shares;

**IMPORTANT: is\_member() has now been converted to legacy, going forward is\_account\_group\_member() function will be used to check if the current user is a member of the account-level group or not. To read more about these recent changes**[**click here**](https://docs.databricks.com/data-governance/unity-catalog/create-views.html#:~:text=current_user()%3A%20Returns,level%20group%20membership.)**. You might still see is\_member()** **in the actual exam as this is one of the recent changes done by Databricks.**

More Info: [Column-level permissions on Dynamic Views](https://docs.databricks.com/data-governance/unity-catalog/create-views.html#:~:text=CREATE%20VIEW%20sales_redacted%20AS%0ASELECT%0A%20%20user_id%2C%0A%20%20CASE%20WHEN%0A%20%20%20%20is_account_group_member(%27auditors%27)%20THEN%20email%0A%20%20%20%20ELSE%20%27REDACTED%27%0A%20%20END%20AS%20email%2C%0A%20%20country%2C%0A%20%20product%2C%0A%20%20total%0AFROM%20sales_raw)

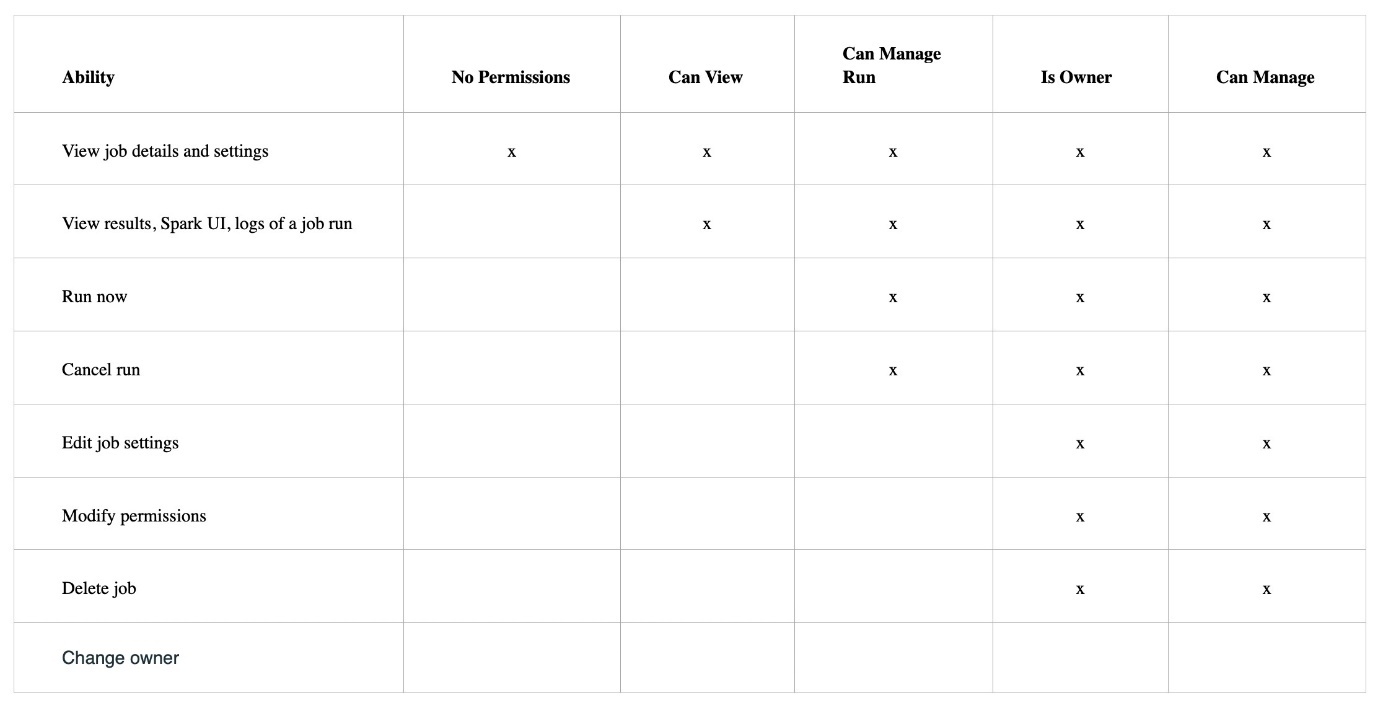
## Question 48:

A data engineering team wants to give the run permission on a job to a junior data engineer who can run the particular job using the Jobs UI but also want to prevent accidental deletion of the job by the junior data engineer. Which of the following permissions can be granted to the junior data engineer to enable them to run the job?

* **No permissions are required, anyone can run the job using Jobs UI.**
* **The junior data engineer should have Can Manage permission to run the job.**
* **Can View permission can be granted to the user to enable them to run the job.**
* **The junior data engineer should be given the Can Manage Run permission.**
* **The ownership of the job should be transferred to the junior data engineer.**

**Explanation**

There are **5** sets of **job permissions** that can be granted to a user:

Source: [Databricks](https://docs.databricks.com/security/access-control/jobs-acl.html#job-permissions)

Coming to the answer to this question, the user with **Can Manage Run** permission can **run the job** without having the privilege to **delete the job.**

More Info: [Job permissions in Databricks](https://docs.databricks.com/security/access-control/jobs-acl.html#:~:text=Run%20now,x)

## Question 49:

Which of the following statements about the Physical and Logical plans is true?

* **Physical plans can be viewed by using explain method whereas Logical plans can only be viewed in Spark UI.**
* **Logical plans and Physical plans can be viewed only in the Spark UI.**
* **Physical plans and Logical plans are only visible if the History server is enabled for Spark UI.**
* **DataFilters, PushedFilters and the PartitionFilters are part of the Physical plan.**
* **After the submission of the spark application, the Physical plans are laid out followed by the generation of Logical Plans.**

**Explanation**

**Logical plans:**When you submit a spark application, a series of **logical plans** are generated by **Spark** in the below order:

*Parsed or Unresolved Logical plan:* As the name suggests, the **database**, **table** and **column** **names** are not resolved in this plan.

*Analyzed or Resolved Logical plan:* After the generation of the **Unresolved Logical plan**, the **Analyzed or Resolved Logical plan** is generated after resolving the **column**, **table** and **database** **names.**

*Optimized Logical Plan:* Lastly, the **Optimized Logical Plan** is created which reorders the **execution of commands.** It also combines the **same transformation** used multiple times on a DataFrame, if possible. For example, adding **two consecutive filter operations** on a DataFrame column will result in **combining** of the **two** **filter conditions**, resulting in **faster** execution.

**Physical Plan:** A **physical plan** acts as a **link** between **Logical Plans** and the **RDDs.** After the **Optimized Logical Plan**, multiple **Physical Plans** are generated by Spark. The plans are sent to the **Cost Model**, which selects the best plan. The **Physical Plan** is then converted to **RDDs.**

Also, you would notice different types of filters in the **Physical Plan** like **DataFilters**, **PushedFilters** and **PartitionFilters**. Remember, to avoid full table scans every time, Spark encourages making use of the **push-down predicate.** By using push-down predicates, you will avoid **unnecessary reads** from the source. The **PushedFilters** visible in the **Physical Plan** are nothing but **push-down predicates.**

If you want to read more about the **Physical plans** in Spark, you should visit this link.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2049.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

[Physical Plans in Spark](https://www.databricks.com/session_eu19/physical-plans-in-spark-sql)

More Info: [explain method in Spark](https://spark.apache.org/docs/3.1.2/api/python/reference/api/pyspark.sql.DataFrame.explain.html)

## Question 50:

You have received the following error while running your code:

****

What can be the possible reason behind this error message?

**right\_outer join is not supported in Spark**

**You cannot use multiple select operations in a single line of code**

**All the columns from both the DataFrames cannot be selected**

**medal column exists in both the DataFrames**

**medal column does not exist in any of the DataFrames being joined**

**Explanation**

It is an **easy** picker for the exam. As the **last line** of the error states that the **reference to the column medal is ambiguous**, it means that the column is present in **both** the DataFrames - **medalDf** as well as **df**

Once the DataFrames are **joined** and**all**the columns are selected from both the DataFrames, **medal** column is added **twice**. Now, when you try to **select** the **medal** column, it results in **ambiguity.**

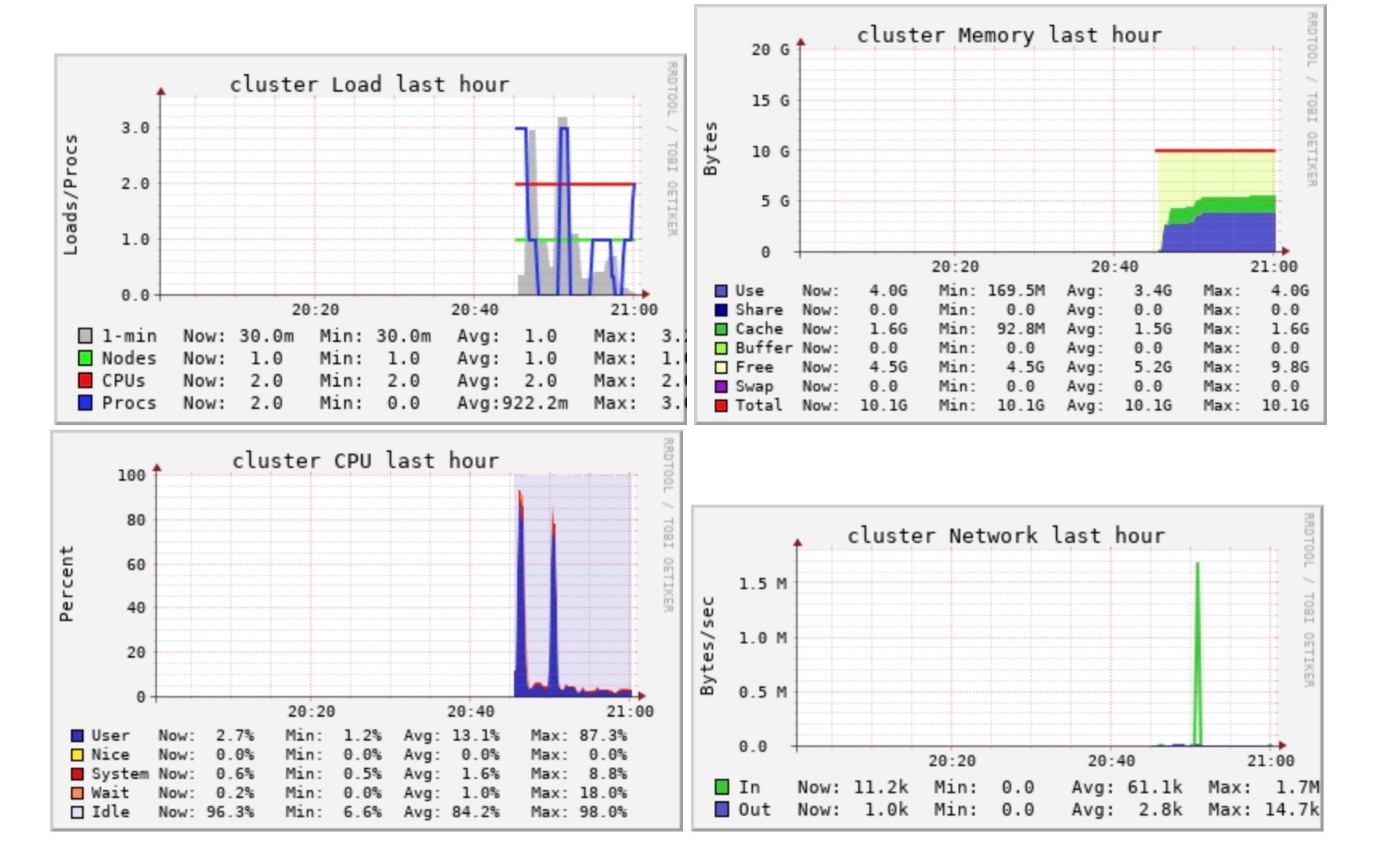
[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2050.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

More Info: [Ambiguous Reference error in Spark](https://docs.databricks.com/error-messages/index.html#:~:text=AMBIGUOUS_REFERENCE,could%20be%3A%20%3CreferenceNames%3E)

## Question 51:

A data engineering team was trying to access the information on cluster load, during the last hour, as the number of users increased drastically. They wanted to know more about the cluster performance in this peak hour. One of the team members quickly turns up with this screenshot. How can the other team members access this dashboard?

****

* **The cluster’s performance information can only be accessed by Databricks Admin.**
* **This dashboard can be seen inside the notebook by running the cluster.detail() command.**
* **This dashboard is a part of the Spark UI’s Executor’s Tab.**
* **Event logs in the Cluster information page display this type of dashboard.**
* **The dashboard can be seen in Ganglia UI.**

**Explanation**

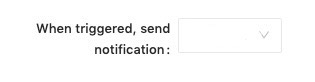
All the cluster-related information including **hourly load**, **cluster memory used**, **cluster CPU usage** and **cluster Network** can be easily accessed from **Ganglia UI.**

To access **Ganglia UI** for a cluster, select the cluster name on the **Compute page** and click on **Metrics**. In the **Metrics tab**, you can find a **link** to **Ganglia UI.**

More Info: [Accessing Ganglia Metrics for a Cluster](https://docs.databricks.com/clusters/clusters-manage.html#ganglia-metrics:~:text=your%20Datadog%20account.-,Ganglia%20metrics,-To%20access%20the)

## Question 52:

The speed of a car is recorded every 5 seconds to trigger notifications to the car owner if the average speed over the past minute is over 100. The data engineer starts to create the alert but is not sure what to enter in the below field:

****

What should be the value in this field, according to you, if the alert has to be triggered every minute if the average speed in the past 1 minute exceeds 100?

* **Always**
* **Just Once**
* **Each time alert is evaluated**
* **At most every 10 minutes**
* **At most every 100 minutes**

**Explanation**

Let us look at each option one by one:

*Always*

**INCORRECT!** There is **no** **Always** option available.

*Just Once*

**INCORRECT!** If you select **Just Once**, the alert will **not** be triggered if the state remains **TRIGGERED** for **two** consecutive minutes but the question demands that the alert should be triggered **every minute** if the **average** **speed** is greater than **100**. So, **Just Once** cannot be used.

*Each time alert is evaluated*

**CORRECT!** If you choose this option, the alert will be triggered **every** **minute** if the average speed is **greater** than **100** in the **last minute**, even if the alert was triggered one minute **prior**.

*At most every 10 minutes*and*At most every 100 minutes*

**INCORRECT!** These options will force the alert to be triggered only **once** in **10** and **100** **minutes** respectively, which is **not** required.

More Info: [Setting up Alerts in Databricks SQL](https://docs.databricks.com/sql/user/alerts/index.html#:~:text=In%20the%20When,that%20trigger%20often)

## Question 53:

A data engineer tries to create a Delta table using **CREATE** command but got the following error:

The associated location (dbfs:/user/hive/warehouse/table1) is not empty but it's not a Delta table

Which of the following should be done before executing the **CREATE** statement again?

* **Restart the cluster**
* **Delete the data location specified in the error i.e dbfs:/user/hive/warehouse/table1**
* **Detach the notebook from the cluster and attach it again.**
* **Change the cluster.**
* **This may be a temporary error, re-run the CREATE command after some time.**

**Explanation**

Once a Delta table is created, a **location** is also created to save its data. If you do not drop the table using the **DROP** statement and the cluster is **terminated**, the new cluster will **not** have the Delta table but will **preserve** the **data**. This will result in the **error** mentioned in the question. To resolve this error, you can **delete** the path specified in the error.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2053.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

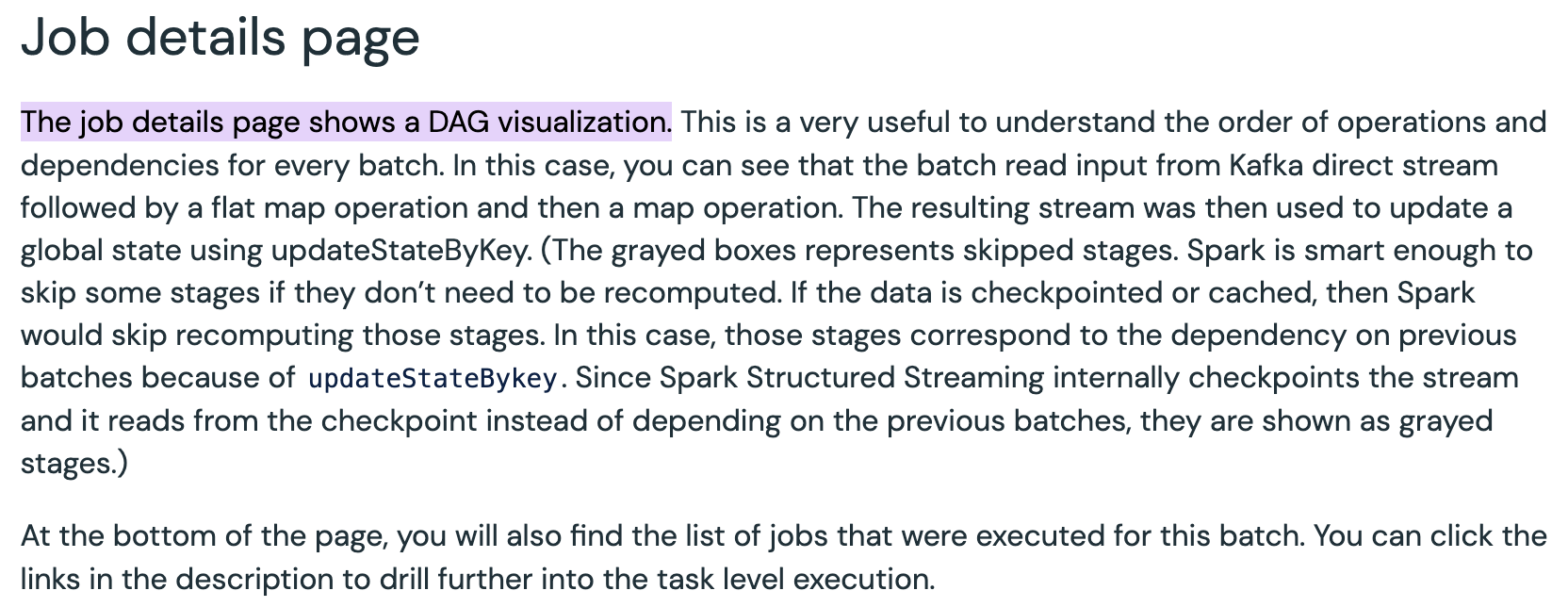
## Question 54:

A data engineer wants to see the visualization of the DAG created for a streaming job. Which of the following Tabs should the data engineer check in Spark UI to view the DAG with all the operations applied for the current batch of data?

* **Jobs**
* **Stages**
* **Structured Streaming**
* **JDBC/ODBC Server**
* **Storage**

**Explanation**

The **DAG** visualization for each batch of data can be viewed in the **Jobs** Tab in **Spark UI.**



More Info: [Jobs Tab of Spark UI](https://docs.databricks.com/en/clusters/debugging-spark-ui.html#job-details-page:~:text=The%20job%20details%20page%20shows%20a%20DAG%20visualization)

## Question 55:

A team member has written a series of unit test cases using **assert** method to test the python function named **return\_sample\_dataframe**. Which of the following libraries must be installed by them to run the unit test cases?

* **pythontesting**
* **pyunittest**
* **pytest**
* **doctest**
* **testdatabricks**

**Explanation**

The **functions**(also known as **units**) are tested by the **developer** by running a single or a series of **test cases.** Typically, the **unit** test cases **send a value to a function** and expect a **value** to be returned. Based on the return value obtained, the test case is said to be **passed** or **failed**.

To run a **python** test case **pytest** library should be installed.

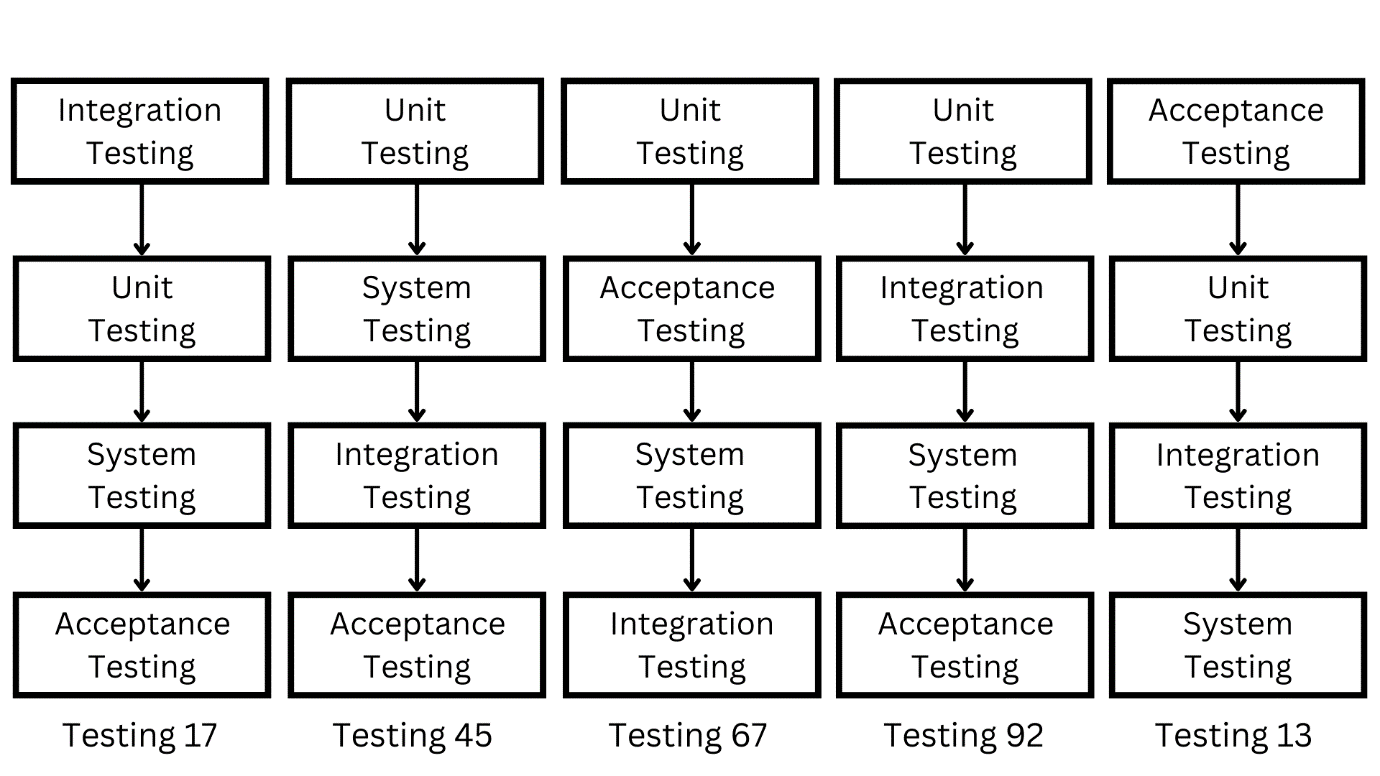
In most languages including **Python** and **Scala**, a series of **assert** methods are used for writing the **unit test cases.**

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%202/Question%2055.html)

*To view all the notebooks in this course, download [Notebooks.dbc](https://certification-champs.github.io/professional-data-engineer-notebooks/Notebooks.dbc) and import to your Databricks Account.*

## Question 56:

**Which of the following is the correct order of testing a software project?**

****

* **Testing 67**
* **Testing 13**
* **Testing 17**
* **Testing 92**
* **Testing 45**

**Explanation**

The correct order of testing is **Unit testing** -> **Integration** **testing** -> **System** **testing** -> **System** **testing.**

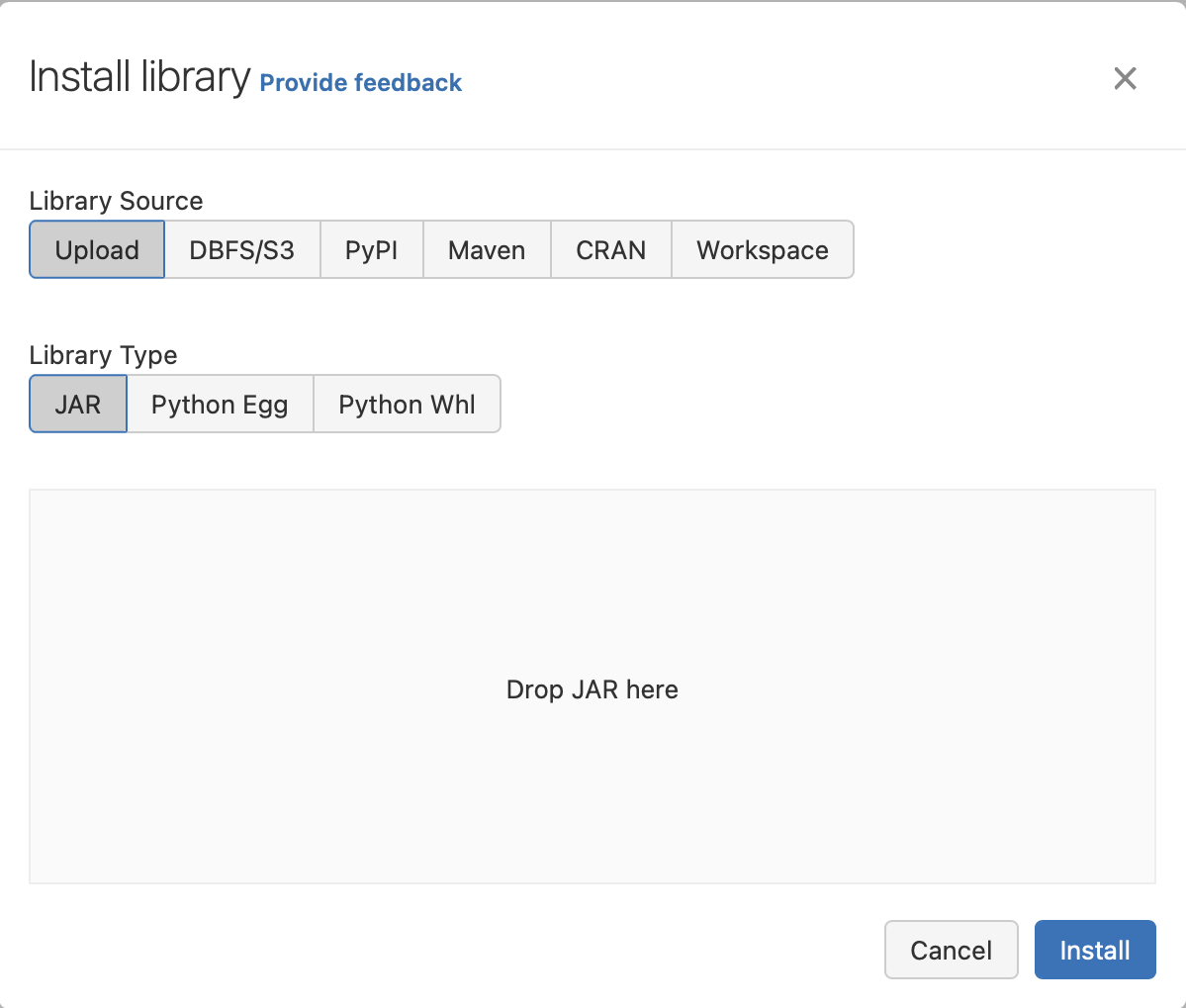
## Question 57:

A workspace administrator is trying to upload a library to a cluster. Which of the following types of files can be added to the cluster as a library, after the cluster has been created and is running?

* **Once the cluster is created and running, only Python wheel files can be added to it.**
* **The files can be added only from PyPI.**
* **After the creation of the cluster, no library can be added to it.**
* **A JAR, Python egg and Python wheel files can be added to a cluster as libraries.**
* **A Python egg and Python wheel files can be added to the cluster.**

**Explanation**

Once the cluster is **created**, you can **add** **libraries** to it from **different** sources including but not limited to your **Workspace**, **DBFS** and **S3.**



Also note, to print the **list of libraries** already **installed**, you can run the following command:

**import sys**

**print(sys.path)**

## Question 58:

A Databricks job consists of a single notebook task that performs an ETL on two different datasets. The notebook contains Python code that extracts CSV files from an AWS S3 location and loads the transformed data in an Azure blob storage. The data from azure blob storage is then extracted, transformed and loaded into a Google Cloud bucket.

The job is scheduled to run daily but the task failed after writing the data to Azure blob storage. The data engineer wants to repair the job using the **Repair Run** utility for the failed tasks. Which statement explains the effect of using **Repair Run** on the job?

* **As Databricks maintains the checkpoint, the task will start from the cell which failed during the execution.**
* **The Repair Run will delete the data from Azure blob storage and execute the task again.**
* **Due to the version control of Databricks notebook, the task will restart from the cell that first failed.**
* **The Repair Run cannot be used for jobs having a single task.**
* **The task will be restarted and all the cells in the notebook will be executed in the order of their existence.**

**Explanation**

Databricks jobs often fail to execute in their **entirety** due to the **cancellation** or **failure** of a task. Sometimes a job with more than **100** tasks fails due to the failure of the **last** task. In this case, the **Repair Run** feature is used to repair the job by running **only** the failed task and its dependent tasks. The **Repair Run** feature is **only** available if the job has **more than one task**.

More Info: [Repair an unsuccessful Job run](https://docs.databricks.com/workflows/jobs/jobs.html#:~:text=You%20can%20repair%20failed%20or%20canceled%20multi%2Dtask%20jobs%20by%20running%20only%20the%20subset%20of%20unsuccessful%20tasks%20and%20any%20dependent%20tasks)

## Question 59:

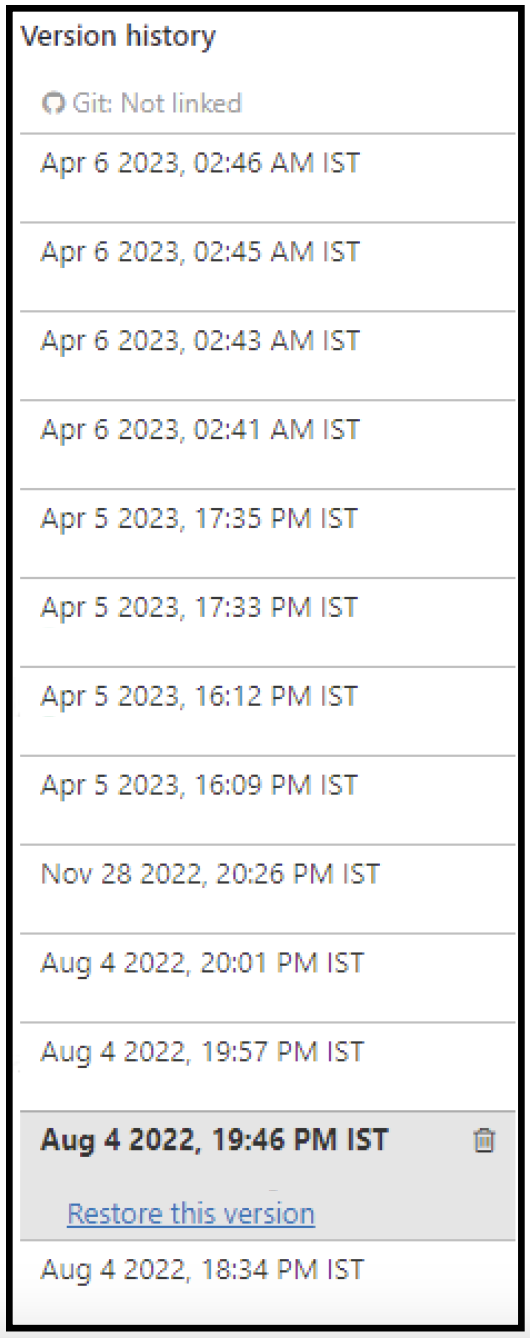
A data engineer wants to see the version history of their notebook which they did not add to Git. How many days of the version history of the notebook will be visible to them, assuming that the history is not cleared?

* **60 days**
* **45 days**
* **No version history is visible if the notebook is not attached to Git.**
* **The version history can be accessed from the time the notebook was created.**
* **30 days**

**Explanation**

Once the first change is done in a **notebook**, the **version history** is created. The version history is preserved **FOREVER**, unless **deleted**. You can perform the following operations on the version history of a notebook:

1. Add **comments** to a version
2. **Restore** a previous version
3. **Delete** a version that is no more required
4. Clear the version history which **deletes** all the previous versions of the notebook.



More Info: [Version history of a Notebook in Databricks](https://docs.databricks.com/notebooks/notebooks-code.html#:~:text=Databricks%20notebooks%20maintain%20a%20history%20of%20notebook%20versions%2C%20allowing%20you%20to%20view%20and%20restore%20previous%20snapshots%20of%20the%20notebook.%20You%20can%20perform%20the%20following%20actions%20on%20versions%3A%20add%20comments%2C%20restore%20and%20delete%20versions%2C%20and%20clear%20version%20history)

## Question 60:

Which of the following pairs of REST API endpoints and their response is incorrect?

* **2.0/jobs/create - {"job\_id": 34567}**
* **2.0/jobs/delete - { }**
* **2.0/jobs/run-now - {“run\_id”: 5876}**
* **2.0/clusters/create - {"cluster\_id": “2954-848031-php83cps"}**
* **2.0/jobs/runs/submit - {“run\_id”: 5876}**

**Explanation**

Let us look at each of the options one by one:

**2.0/jobs/create** - The **create job** endpoint returns the **job\_id** of the newly created job. This **job\_id** is the **unique** identifier of the job.

**2.0/jobs/delete** - The **delete job** endpoint **deletes** the job specified by the **job\_id** given in the request, but does **not** return anything except an **empty** JSON.

**2.0/jobs/run-now** - This endpoint returns the **unique** **run\_id** of the triggered job as well as the **sequence** **number** of the current run. The **run\_id** is a globally **unique** ID given to each run of all the jobs.

The correct response would be **{“run\_id”: 5876, “number\_in\_job”: 4}** which means that the unique **run\_id** is **5876** whereas this is the **4th** time this job is **triggered.**

**2.0/clusters/create** - The **cluster creation** endpoint returns the **cluster\_id** of the newly created cluster.

**2.0/jobs/runs/submit** - Unlike the **run-now** endpoint, the **submit** endpoint just returns the **run\_id** of the triggered job.

More Info: Response structures for

[2.0/jobs/create endpoint](https://docs.databricks.com/dev-tools/api/2.0/jobs.html#:~:text=Copy-,%7B%0A%20%20%22job_id%22%3A%201%0A%7D,-Request%20structure)

[2.0/jobs/run-now endpoint](https://docs.databricks.com/dev-tools/api/2.0/jobs.html#:~:text=newly%20triggered%20run.-,number_in_job,sequence%20number%20of%20this%20run%20among%20all%20runs%20of%20the%20job,-.)

[2.0/clusters/create endpoint](https://docs.databricks.com/dev-tools/api/latest/clusters.html#:~:text=Description-,cluster_id,Canonical%20identifier%20for%20the%20cluster.,-Edit)

[2.0/jobs/runs/submit endpoint](https://docs.databricks.com/dev-tools/api/2.0/jobs.html#:~:text=Copy-,%7B%0A%20%20%22run_id%22%3A%20123%0A%7D,-Request%20structure)

Bottom of Form