Contents

[Question 1: 4](#_Toc156661635)

[Question 2: 5](#_Toc156661636)

[Question 3: 6](#_Toc156661637)

[Question 4: 7](#_Toc156661638)

[Question 5: 9](#_Toc156661639)

[Question 6: 11](#_Toc156661640)

[Question 7: 12](#_Toc156661641)

[Question 8: 13](#_Toc156661642)

[Question 9: 14](#_Toc156661643)

[Question 10: 15](#_Toc156661644)

[Question 11: 16](#_Toc156661645)

[Question 12: 17](#_Toc156661646)

[Question 13: 18](#_Toc156661647)

[Question 14: 20](#_Toc156661648)

[Question 15: 21](#_Toc156661649)

[Question 16: 23](#_Toc156661650)

[Question 17: 24](#_Toc156661651)

[Question 18: 26](#_Toc156661652)

[Question 19: 27](#_Toc156661653)

[Question 20: 28](#_Toc156661654)

[Question 21: 31](#_Toc156661655)

[Question 22: 32](#_Toc156661656)

[Question 23: 33](#_Toc156661657)

[Question 24: 34](#_Toc156661658)

[Question 25: 35](#_Toc156661659)

[Question 26: 36](#_Toc156661660)

[Question 27: 37](#_Toc156661661)

[Question 28: 38](#_Toc156661662)

[Question 29: 40](#_Toc156661663)

[Question 30: 41](#_Toc156661664)

[Question 31: 42](#_Toc156661665)

[Question 32: 43](#_Toc156661666)

[Question 33: 44](#_Toc156661667)

[Question 34: 45](#_Toc156661668)

[Question 35: 46](#_Toc156661669)

[Question 36: 47](#_Toc156661670)

[Question 37: 48](#_Toc156661671)

[Question 38: 49](#_Toc156661672)

[Question 39: 50](#_Toc156661673)

[Question 40: 51](#_Toc156661674)

[Question 41: 53](#_Toc156661675)

[Question 42: 54](#_Toc156661676)

[Question 43: 56](#_Toc156661677)

[Question 44: 57](#_Toc156661678)

[Question 45: 58](#_Toc156661679)

[Question 46: 59](#_Toc156661680)

[Question 47: 60](#_Toc156661681)

[Question 48: 62](#_Toc156661682)

[Question 49: 63](#_Toc156661683)

[Question 50: 64](#_Toc156661684)

[Question 51: 65](#_Toc156661685)

[Question 52: 66](#_Toc156661686)

[Question 53: 67](#_Toc156661687)

[Question 54: 69](#_Toc156661688)

[Question 55: 70](#_Toc156661689)

[Question 56: 71](#_Toc156661690)

[Question 57: 73](#_Toc156661691)

[Question 58: 74](#_Toc156661692)

[Question 59: 75](#_Toc156661693)

[Question 60: 76](#_Toc156661694)

## Question 1:

A data engineer has scheduled a job that uses the legacy code involving a shell script. The job is scheduled to run every day at 2 PM. The shell script tries to copy a file from **/tmp**directory on the local file system to the **/FileStore** directory using the following command:

1. %sh
2. cp /tmp/raw\_data.csv /FileStore/data.csv

After a few weeks, the data processing team informs the engineering team about the non-availability of the **/tmp** directory from the past 1 week. The data engineer later found that even after the source directory was unavailable, the job did not fail.

What could be the possible reason for this outcome?

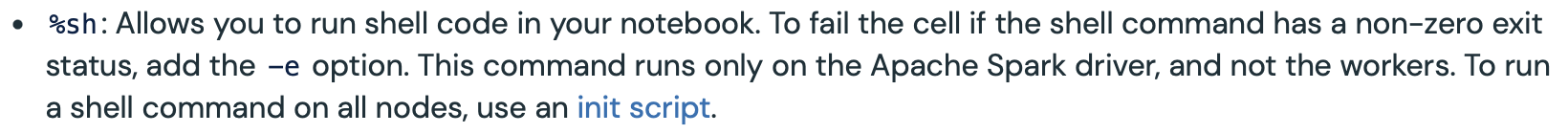
* **-e is missing from the cp command, it should be cp /tmp/raw\_data.csv /FileStore/data.csv -e**
* **-error is missing from the cp command, it should be cp /tmp/raw\_data.csv /FileStore/data.csv -error**
* **-e is missing from the %sh magic command, it should be %sh -e**
* **-error is missing from the %sh magic command, it should be %sh -error**
* **-error is missing from the %sh magic command and -e is missing from the cp command**

**Explanation**

When you work with **shell commands** inside a notebook’s cell using the **%sh** magic command, the cell will **not** fail even if the command results in an error. To **force the cell to fail**, in case of any error, you should use **-e** with the **%sh** command i.e.

1. %sh -e
2. cp tmp/raw\_data.csv FileStore/data.csv

Since the **/tmp** directory is **unavailable**, using **-e** with **%sh** command will **fail the cell** which will result in the **job failure.**



[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%203/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: [Failing the shell command on error](https://docs.databricks.com/en/notebooks/notebooks-code.html#mix-languages:~:text=%25sh%3A%20Allows%20you%20to%20run%20shell%20code%20in%20your%20notebook.%20To%20fail%20the%20cell%20if%20the%20shell%20command%20has%20a%20non%2Dzero%20exit%20status%2C%20add%20the%20%2De%20option.)

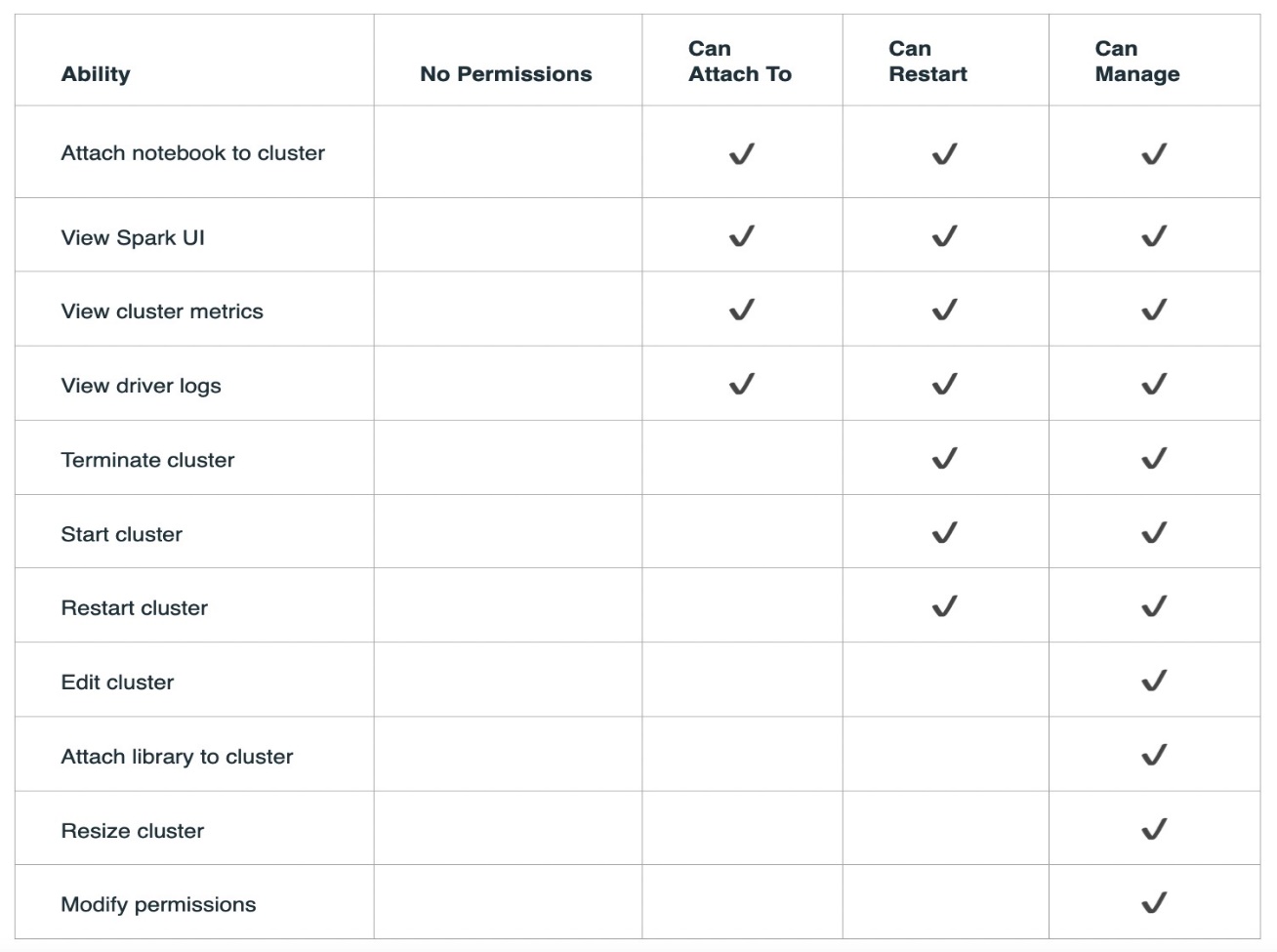
## Question 2:

A Databricks workspace admin has enabled access control on the clusters. A new team member has been added to the team and wants to run a notebook to ingest the data from an S3 bucket. To run the notebook, the new team member tries to start an already existing interactive cluster named **cluster\_to\_ingest\_s3\_data** but doesn’t have permission. Which of the following permissions can be granted to the member to enable them to start the cluster?

* **Can Attach To permission**
* **Can Restart permission**
* **Can Manage permission**
* **Either A or B**
* **Either B or C**

**Explanation**

Cluster-level permission is usually set to **limit the accessibility** to a cluster. There are **4 different permission levels** that can be set to a cluster. Each permission level enables a different set of cluster actions. The following table lists the **permissions**granted at each level:



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

According to the table, anyone having **Can Restart** or **Can Manage** permission will be able to **start** the cluster.

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

## Question 3:

A new data engineer tries to create a job using Databricks REST API by posting the following JSON to the **2.0/jobs/create** endpoint.

{

"name": "new\_job"

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

"spark\_python\_task": {

"python\_file": "dbfs:/first\_method.py",

}

}

Another data engineer from the same team tries to create a job using the following JSON

{

"name": "new\_job"

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

"spark\_python\_task": {

"python\_file": "dbfs:/example.py",

}

}

Assuming that the first job is created successfully. Which of the following statements explains the outcome when the second data engineer tries to create the job?

* **The job will be created successfully with both the jobs named new\_job**
* **The job will be created successfully with the second job named new\_job\_1**
* **The job will not be created as a job with the same name already exists.**
* **The job will be created successfully by overwriting the previous job as two jobs cannot share a name.**
* **The task in the second job will be appended to the existing job.**

**Explanation**

*This is a tricky question.*

While creating a **new** job using **Databricks REST API**, the name of the job in the JSON payload is ***optional***. The **default** name is **Untitled**. It means that you can post **multiple JSONs** to the **2.0/jobs/create** endpoint **without the name parameter**. It does **not** return any error and **creates multiple jobs** with the same name - **Untitled**. In this question also, the name of the two jobs is the same. So, the jobs will be created **successfully** with both the jobs named **new\_job**

**Exam Tip** **-** If you post the **same JSON more than once**, exact copies of the job will be created. All the jobs will have the **same name** but **different IDs.** The **ID** of a job acts as a **unique** identifier to differentiate between jobs in Databricks.

More Info: [Creating a new job through Databricks REST API](https://docs.databricks.com/dev-tools/api/2.0/jobs.html#create)

## Question 4:

Which of the following defines the difference between **cache()** and **persist()** in Spark?

* **cache() persists the DataFrame in memory whereas persist() saves it in memory as well as the disk.**
* **persist() accepts storage level as an argument while cache() does not.**
* **The default storage level for cache() and persist() is MEMORY\_ONLY**
* **Unpersisting the DataFrames persisted through cache() is not possible while the DataFrames persisted through persist() can be unpersisted using unpersist()**
* **cache() should be used to persist a DataFrame for a small period of time. To persist a DataFrame for a longer period, persist() should be used.**

**Explanation**

**cache()** and **persist()** are two of the operations used to **persist** the DataFrames. Persisting the DataFrames can **increase** **the** **performance** of your jobs. Let us look at each option one by one:

***cache()****persists the DataFrame in memory whereas****persist()****saves it in memory as well as the disk.*

**INCORRECT!** The default storage level of both operations is **MEMORY\_AND\_DISK**

***persist()****accepts storage level as an argument while****cache()****does not.*

**CORRECT!** Although, **both** can be used to **persist** a DataFrame, only **persist()** accepts different **storage levels** as an argument.

*The default storage level for****cache()****and****persist()****is****MEMORY\_ONLY***

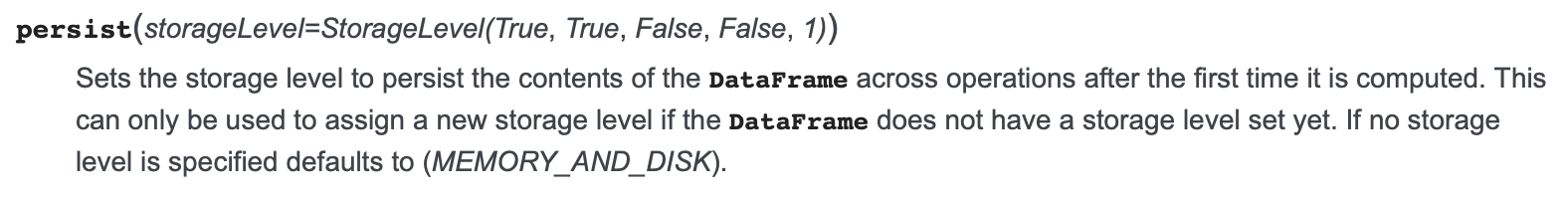
**INCORRECT!** The **default** storage level of both operations is **MEMORY\_AND\_DISK**

*Unpersisting the DataFrames persisted through****cache()****is not possible while the DataFrames persisted through****persist()****can be unpersisted using****unpersist()***

**INCORRECT!** **unpersist()** can be used to **unpersist** a DataFrame persisted through **both** **cache()** and **persist()**

***cache()****should be used to persist a DataFrame for a small period of time. To persist a DataFrame for a longer period,****persist()****should be used.*

**INCORRECT!** The only difference between **cache()** and **persist()** is that **persist()** accepts various **storage levels** as an argument whereas **cache()** does **not**. The **default** storage level of both operations is **MEMORY\_AND\_DISK**



[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%203/Question%204.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: [cache operation in Spark](https://spark.apache.org/docs/3.0.0-preview/api/python/pyspark.sql.html?highlight=cache#pyspark.sql.DataFrame.cache:~:text=Persists%20the%20DataFrame%20with%20the%20default%20storage%20level%20(MEMORY_AND_DISK).) | [persist operation 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.%20This%20can%20only%20be%20used%20to%20assign%20a%20new%20storage%20level%20if%20the%20DataFrame%20does%20not%20have%20a%20storage%20level%20set%20yet.%20If%20no%20storage%20level%20is%20specified%20defaults%20to%20(MEMORY_AND_DISK).)

## Question 5:

When mounting an external data source in Databricks, which of the following options provides the best approach for managing secrets, such as access keys or credentials, required to access the external storage?

* **Storing secrets directly in the Databricks notebook code.**
* **Embedding secrets in environment variables within the Databricks cluster configuration.**
* **Including secrets in plain text files within the mount options when defining the mount point.**
* **Storing secrets in Databricks Secrets.**
* **Using a third-party password manager to securely store and retrieve the secrets.**

**Explanation**

When mounting an **external data source** in Databricks, it is crucial to handle **secrets**, such as **access** **keys** or **credentials**, in a **secure** manner. Let us look at each option one by one to find the most secure option.

*Storing secrets directly in the Databricks notebook code.*

Storing secrets directly in the Databricks notebook code can introduce **security risks**, as the **secrets** may be **exposed** in **logs** or **inadvertently shared.**

*Embedding secrets in environment variables within the Databricks cluster configuration.*

Similarly to option A, Embedding secrets in **environment variables** within the Databricks **cluster configuration** can also introduce **security risks**, as the **secrets** may be **exposed** in **logs** or **inadvertently shared.**

*Including secrets in plain text files within the mount options when defining the mount point.*

It is **not recommended** by Databricks, as it can expose sensitive information.

*Storing secrets in Databricks Secrets.*

Databricks Secrets provides a secure key-value store specifically designed for storing and managing secrets. It **encrypts** the stored values at rest and in transit, providing an **added layer of security.** Secrets can be securely accessed within notebooks or jobs using the **Databricks Secrets API.**

*Using a third-party password manager to securely store and retrieve the secrets.*

It can be a viable option, but **Databricks Secrets** is a **native** feature of Databricks that offers a seamless and **integrated solution** for managing secrets within the platform.

More Info: [Secret Management in Databricks](https://docs.databricks.com/security/secrets/index.html)

## Question 6:

The data engineering team is facing issues with cloud storage costs. The cloud team informed the engineering team about the historic data stored for **transactions** Delta table. The data engineering team wants to get rid of the data files older than 60 hours. One of the members suggested using **VACUUM** command to remove the data completely but the team wants to check the list of files that will be deleted after running the **VACUUM** command. Which of the following commands can be used for that?

* **VACUUM transactions 2.5 DRY RUN**
* **VACUUM transactions 60 DRY\_RUN**
* **VACUUM transactions 2.5 DRY\_RUN**
* **VACUUM transactions 60 DRY RUN**
* **VACUUM transactions 2.5 DAYS DO\_NOT\_DELETE**

**Explanation**

The **VACUUM** command is used to **get rid of older data**from Delta tables. Using **DRY RUN** at the end of the **VACUUM** command enables the user to have a look at the files **which will be deleted** using the command. The **first 1000 files** will be displayed if the number of files to be deleted is greater than 1000.

Hence, the correct code block would be **VACUUM transactions 60 DRY RUN** where 60 refers to the number of **retention hours**. Also note, running this command will **not** delete any file associated with the **transactions** table. If you need to delete the files **DRY RUN** should be **omitted** from the command.

More Info: [Using DRY RUN with VACUUM command](https://docs.databricks.com/sql/language-manual/delta-vacuum.html#:~:text=The%20retention%20threshold.-,DRY%20RUN,Return%20a%20list%20of%20up%20to%201000%20files%20to%20be%20deleted,-.)

## Question 7:

Which of the following statements is not true about clusters in Databricks?

* **An all-purpose cluster can be created using the UI, CLI or REST API.**
* **A job will always use a new Job cluster.**
* **A job cluster cannot be restarted.**
* **All-purpose clusters should be used to develop code in interactive notebooks.**
* **A job cluster is less costly than an all-purpose cluster when used to run the same notebook.**

**Explanation**

All the options are correct **except** the **second** one. A job can use the following type of clusters:

1. *A new job cluster*
2. *A shared job cluster*
3. *An existing all-purpose cluster.*

More Info: [Types of Clusters in Databricks](https://docs.databricks.com/clusters/index.html#clusters:~:text=You%20can%20create,a%20job%20cluster.)

## Question 8:

A Databricks user needs to cancel the run of a job but does not have the access to the REST API or the UI. The only access provided to the user is Databricks CLI. Which of the following commands can be used by the user to cancel the run of a job with the following details:

**job-id** - **2795**

**run-id**-**96746**

**job-name** - **fetch\_details**

* **databricks run cancel --run-id 2795**
* **databricks runs cancel --job-id 2795 --run-id 96746**
* **databricks run cancel --job-name fetch\_details --run-id 96746**
* **databricks runs cancel --run-id 96746**
* **databricks run cancel --job-id 2795 --run-id 96746**

**Explanation**

In simple terms, **every job** in Databricks has a **unique** **job-id** Once the job runs, it has a **unique** **run-id** If you wish to **cancel a job run**, you would use the **run-id** of the job. So, the correct answer would be the one using only the **run-id** of the job i.e **databricks runs cancel --run-id 96746**

Moreover, you **cannot** use a **job-name** as a uniqueidentifier for a job as **more than one job** can share a **similar** name.

More Info: [Cancel a Job Run using Databricks CLI](https://docs.databricks.com/dev-tools/cli/runs-cli.html#:~:text=databricks%20runs%20cancel%20%2D%2Drun%2Did%20119)

## Question 9:

A data engineer posts the following JSON to the **2.0/jobs/create** endpoint of Databricks REST API.

{

"name": "test\_api",

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

"notebook\_task": {

"notebook\_path": "/example.py",

},

"schedule": {

"quartz\_cron\_expression": "0 0 8 \* \* ?",

"timezone\_id": "Australia/Brisbane"

}

}

Assuming cluster **1198-132537-dht25rtr** exists, what will be the expected outcome?

* **The job named test\_api will be created and the job will be triggered daily when the clock strikes 8:00 AM in Brisbane, Australia.**
* **The job will not be created as the schedule for the job cannot be created using 2.0/jobs/create endpoint.**
* **The job named test\_api will be created and the job will be triggered on the 8th day of each month.**
* **The job will be created but will not be triggered as the Cron expression is not valid.**
* **The job named test\_api will be created but can only be triggered by clicking Run Now in Jobs UI.**

**Explanation**

The **2.0/jobs/create** endpoint of the **Databricks REST API** can be used to **create** and **schedule** a job. Once the schedule has been set for a job, it will **automatically** trigger at the mentioned time.

In this question, a similar schedule has been set for the job which can trigger the job daily at **8:00 AM** according to the Brisbane, Australia time. Also note, the Cron expression uses **Quartz syntax**, which enables the usage of **seconds** while scheduling the job.

More Info: [Scheduling a Job using Databricks REST API](https://docs.databricks.com/dev-tools/api/2.0/jobs.html#cronschedule)

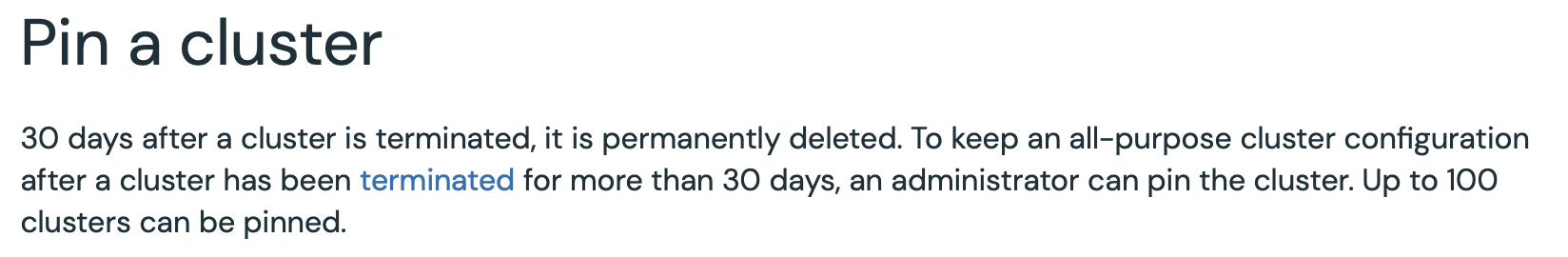
## Question 10:

How can a Workspace Admin ensure that an all-purpose cluster retains its configurations even if it remains terminated for over 30 days?

* **Set automatic termination to 30 days**
* **Use Databricks UI to create the cluster**
* **Use Databricks REST API to create the cluster**
* **Pin the cluster**
* **Use Databricks CLI to create the cluster**

**Explanation**

As you might know, if a cluster remains terminated for more than **30 days**, the cluster is **permanently deleted.** To **retain** its **configurations** even after 30 days, you can **pin the cluster.**



More Info: [Retaining the configuration of a cluster by pinning](https://docs.databricks.com/en/clusters/clusters-manage.html#pin-a-cluster:~:text=To%20keep%20an%20all%2Dpurpose%20cluster%20configuration%20after%20a%20cluster%20has%20been%20terminated%20for%20more%20than%2030%20days%2C%20an%20administrator%20can%20pin%20the%20cluster)

## Question 11:

A data engineer is assigned the task of creating a table using a CSV file stored in local storage. The data engineer executes the following SQL statement and the table is created successfully.

1. CREATE TABLE venues
2. (name STRING, area INT)
3. USING CSV
4. LOCATION 'dbfs:/FileStore/data/venues.csv'

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**

If you try to insert data in the **venues** table, the following error will be displayed.

**AnalysisException: Cannot insert into dbfs:/FileStore/data/venues.csv, as it is a file instead of a directory.**

As the **venues** table is created using **dbfs:/FileStore/data/venues.csv** as **LOCATION**, new records **cannot** be **inserted**. If 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 in **dbfs:/FileStore/data/** directory.

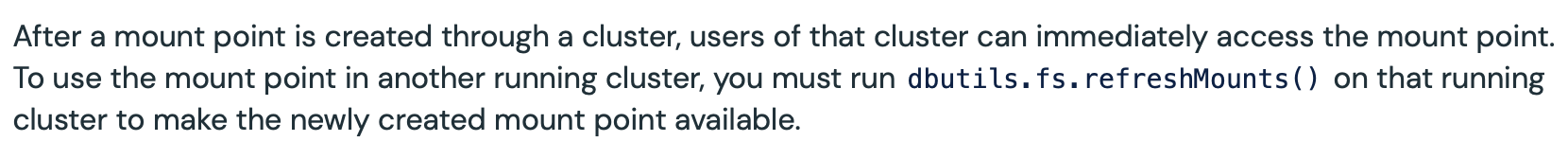
## Question 12:

A data engineer created a new cluster named **s3mount** and attached a notebook to it, which mounts an S3 bucket to DBFS. The other data engineers are using an existing cluster named **prod-support** to run their notebooks. One of the data engineers tries to access the data from the mount point using their notebook. Which of the following statements is true about data accessibility?

* **The data engineer will be able to access the data without any errors.**
* **The mount point will only be accessible through the notebooks attached to the s3mount cluster.**
* **To access the data on the prod-support cluster, dbutils.refreshMounts() needs to run.**
* **The only way to access the data on the s3mount cluster is to restart the cluster.**
* **The mount point will be accessible on the prod-support cluster after the data engineer runs dbutils.fs.refreshMounts()**

**Explanation**

The data engineer is trying to access the **mount point** which was created using **another cluster**. To access the mount point on the **already running** prod-support cluster, the data engineer needs to run dbutils.fs.refreshMounts()



More Info: [Refreshing a Mount point in Databricks](https://docs.databricks.com/en/dbfs/mounts.html#mount-an-s3-bucket:~:text=After%20a%20mount%20point%20is%20created%20through%20a%20cluster%2C%20users%20of%20that%20cluster%20can%20immediately%20access%20the%20mount%20point.%20To%20use%20the%20mount%20point%20in%20another%20running%20cluster%2C%20you%20must%20run%20dbutils.fs.refreshMounts()%20on%20that%20running%20cluster%20to%20make%20the%20newly%20created%20mount%20point%20available.)

## Question 13:

A data engineer recently learned about the schema evolution modes in Databricks Auto Loader. Which of the following modes could be selected by them to ensure that the stream does not fail if a new column is encountered?

* **dropNewColumns**
* **none**
* **rescue**
* **Any one from rescue or none can be used**
* **Any one from rescue or dropNewColumns can be used**

**Explanation**

Let us look at all the options one by one:

***dropNewColumns***

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

***none***

**INCORRECT!** 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**. As the question only specifies that the stream should **not** fail when a **new** column is encountered, either **rescue** or **none** can be used.

***rescue***

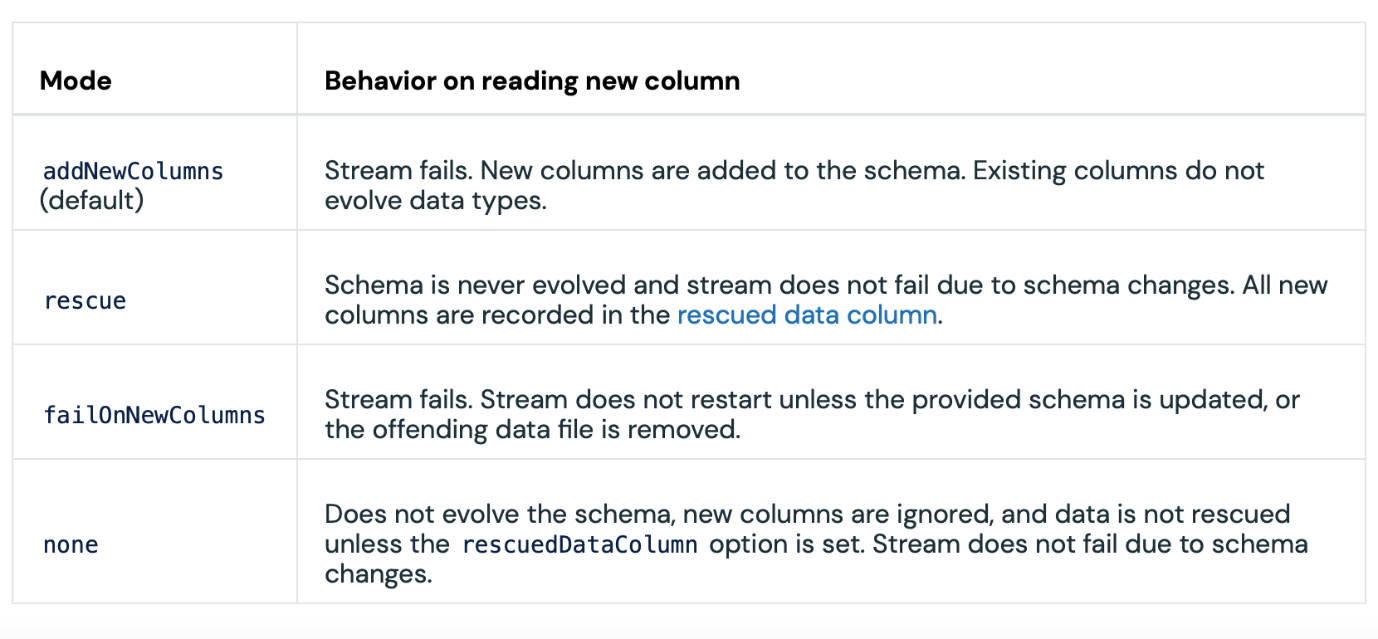
**INCORRECT!** If this mode is selected, the stream will **not** fail even if a **new** column is detected. The column will be added to the **rescued data column** while the schema will **not** be **evolved**. As the question only specifies that the stream should **not** fail when a new column is encountered, either **rescue** or **none** can be used.

*Any one from****rescue****or****none****can be used*

**CORRECT!** As the question only specifies that the stream should **not** fail when a **new** column is encountered, either **rescue** or **none** can be used.

*Any one from****rescue****or****dropNewColumns****can be used*

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

More Info: [Schema evolution modes in Auto Loader](https://docs.databricks.com/en/ingestion/auto-loader/schema.html#how-does-auto-loader-schema-evolution-work:~:text=addNewColumns%20(default),set.%20Stream%20does%20not%20fail%20due%20to%20schema%20changes)

## Question 14:

A data engineer executes the following query to optimize the join operation between df1 and df2:

**joined\_df = df1.join(broadcast(df2), “id”, “inner”)**

Which of the following correctly explains the working of the join?

* **A copy of df2 will be broadcasted to all the worker nodes.**
* **The join query will fail as inner should be replaced with broadcast.**
* **Only the first 10 MBs of data from the DataFrame df2 will be used for the join.**
* **The join query will fail as the broadcast\_df should be used instead of broadcast.**
* **The joined\_df will be broadcasted to all the worker nodes as the broadcast function is used on one of the DataFrames involved in the join.**

**Explanation**

According to the spark documentation, the broadcast function

**"Marks a DataFrame as small enough for use in broadcast joins."**

But, once the query is executed the copy of broadcasted DataFrame is distributed to all the worker nodes.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%203/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: [Broadcast function in PySpark](https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.functions.broadcast.html#:~:text=Marks%20a%20DataFrame%20as%20small%20enough%20for%20use%20in%20broadcast%20joins)

## Question 15:

A data engineer runs the below query:

spark.readStream.format('delta') \

.table('stock\_prices') \

.join(table('company\_info'), how = 'left', on='id') \

.writeStream \

.option('checkpointLocation', '/tmp/share\_details') \

.format('delta') \

.table('shares')

Which of the following statements is true about the result of the query if one of the data analysts changes the schema for the **company\_info** table.

* **The streaming query will fail as soon as the schema is changed for the company\_info table.**
* **The streaming query will fail as soon as the next batch of data is received in the stock\_prices table only if the id column is dropped as part of the schema change.**
* **The streaming query will fail as soon as the next batch of data is received in the stock\_prices table.**
* **The streaming query will fail as soon as the next batch of data is received in the stock\_prices tables only if the id column is dropped or renamed as part of the schema change.**
* **The schema for the company\_info table cannot be changed while the streaming query is running.**

**Explanation**

Let us look at all the options:

*The streaming query will fail as soon as the schema is changed for the****company\_info****table.*

**INCORRECT!** The effect of schema change in the **static table** is witnessed only when the **next** **batch** of data arrives in the **streaming table.**

*The streaming query will fail as soon as the next batch of data is received in the****stock\_prices****tables only if the****id****column is dropped as part of the schema change.*

**INCORRECT!** This statement is **partially** correct. The query will **fail** if the **id** column is dropped from the **company\_info** table but it will **also** fail if the **id** column is **renamed.**

*The streaming query will fail as soon as the next batch of data is received in the****stock\_prices****table.*

**INCORRECT!** In this streaming query, the column used from the **static** **table** is **id**. As long as the **id** column is **not** altered as part of the schema change, the query will **continue** to run.

*The streaming query will fail as soon as the next batch of data is received in the****stock\_prices****tables only if the****id****column is dropped or renamed as part of the schema change.*

**CORRECT!** Once the **id** column is **altered**(*dropped or renamed*), the query will **not** **fail** **immediately**. The query will fail **only** when the **new batch** of data arrives in the streaming table.

*The schema for the****company\_info****table cannot be changed while the streaming query is running.*

**INCORRECT!** The schema **can** **be** **changed** while the streaming query is **in progress.**

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%203/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: [Stream-static join in Spark](https://docs.databricks.com/structured-streaming/delta-lake.html#:~:text=A%20stream%2Dstatic,current%20micro%2Dbatch)

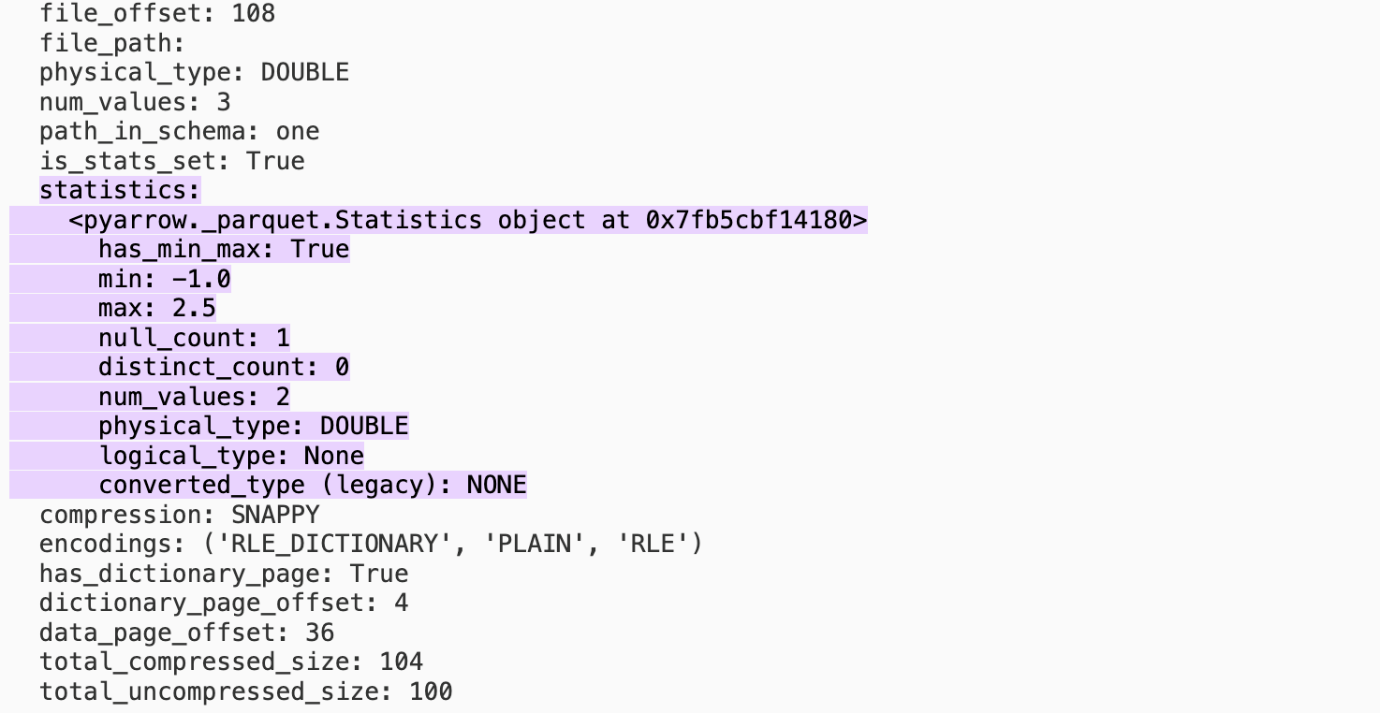
## Question 16:

Which of the following column statistics is not a part of the Parquet file footers?

* **Min value**
* **Max value**
* **NULL counts**
* **Mean value**
* **The data type of columns**

**Explanation**

Parquet file column statistics look like this:



Clearly mean value is **not** a part of the**column-level statistics**in the **Parquet footers**.

***In the case of the delta table, these statistics are stored in the Delta log.***

More Info: [Delta Log statistics](https://docs.databricks.com/en/delta/data-skipping.html#:~:text=Delta%20Lake%20on%20Databricks%20takes%20advantage%20of%20this%20information%20(minimum%20and%20maximum%20values)%20at%20query%20time%20to%20provide%20faster%20queries.)

## Question 17:

To store the latest bills of all the customers, a broadband company uses a delta table named **bills** with Change Data Feed enabled. Whenever a bill is generated for a customer for the first time, the bill details are added to the table. Afterward, each time a new bill is generated for a customer, only the **bill\_date** and **amount** columns are updated. If a customer parts ways with the company, the bill details are deleted from the **bills** table. Which of the following queries should be used to get the list of all the customers with details of their first-ever bill?

* SELECT customer\_id, bill\_date, amount

FROM table\_changes('bills', 0)

WHERE \_change\_type = 'INSERT';

* SELECT customer\_id, bill\_date, amount

FROM table\_changes('bills', 0)

WHERE \_change\_type in ('insert', 'update\_preimage');

* SELECT customer\_id, bill\_date, amount

FROM table\_changes('bills', 0)

WHERE \_change\_type in ('insert', 'delete');

* **SELECT customer\_id, bill\_date, amount**

**FROM table\_changes('bills', 0)**

**WHERE \_change\_type = 'insert';**

* SELECT customer\_id, bill\_date, amount

FROM bills

WHERE \_change\_type = 'INSERT';

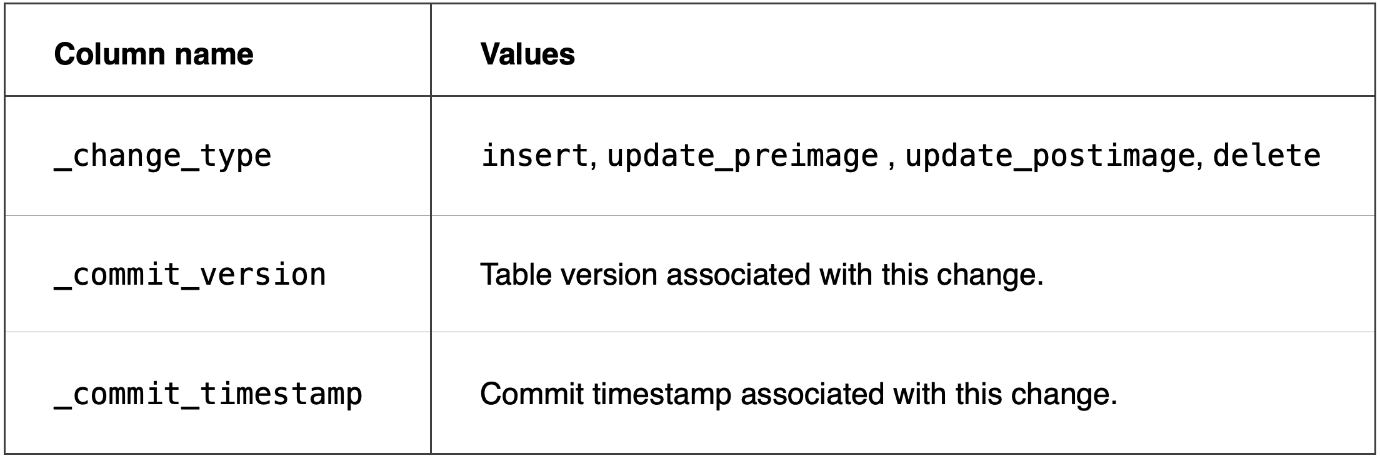
**Explanation**

Once **Change Data Feed** is enabled for a delta table, the change data logs can be accessed using **table\_changes()** function. To output the **bills** table with all the **historical data**, the following query can be used:

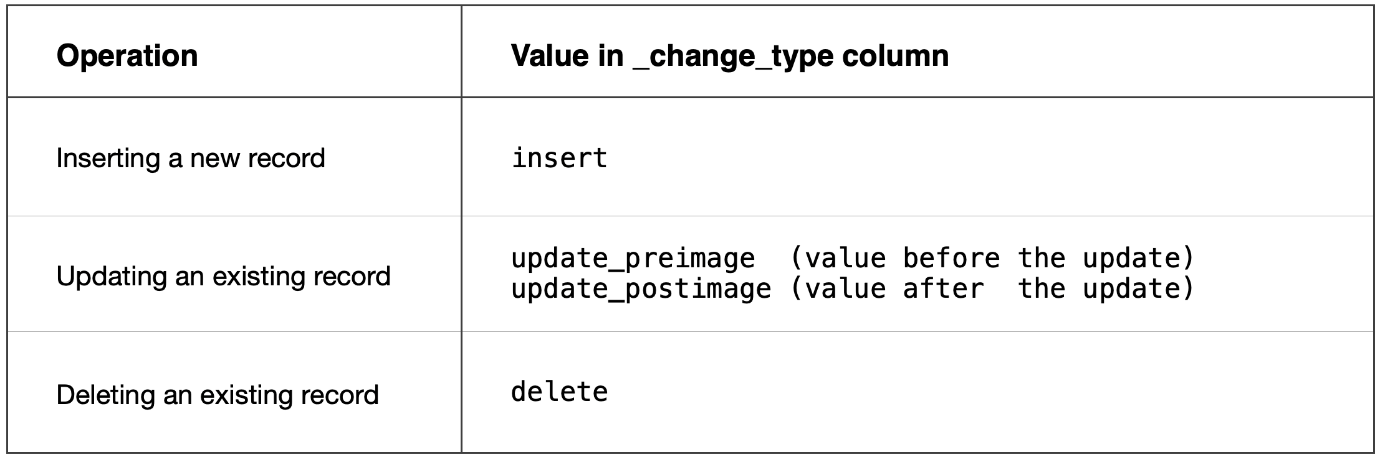
**SELECT \* FROM table\_changes('bills', 0);**

This will print out all the **insertions, deletions, and updations** on the **bills** table starting from **version 0**. Version 0 refers to the **table creation version.**

The **table\_changes()** will print **all** the columns from the **original table(bills)** with the following **three** more columns:



The following are the **values** inserted in the **\_change\_type** column according to the **operation** performed on the table:



So, the correct option will be **option D** i.e.

1. SELECT customer\_id, bill\_date, amount
2. FROM table\_changes('bills', 0)
3. WHERE \_change\_type = 'insert';

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%203/Question%2017.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: [Change Data Feed schema for a Delta Table](https://docs.databricks.com/en/delta/delta-change-data-feed.html#what-is-the-schema-for-the-change-data-feed)

## Question 18:

Which of the following is the default trigger interval for a streaming job in Databricks?

* **5 ms**
* **1000 ms**
* **10 ms**
* **100 ms**
* **500 ms**

**Explanation**

If the trigger option is missing from the streaming query, the data is read every **500ms**, by default. To change the trigger interval you can add **trigger()** in the query.

For example **trigger(processingTime='10 seconds')** or **trigger(processingTime='5 minutes')**



More Info: [Default trigger interval in a streaming query](https://docs.databricks.com/en/structured-streaming/triggers.html#what-is-the-default-trigger-interval:~:text=Structured%20Streaming%20defaults%20to%20fixed%20interval%20micro%2Dbatches%20of%20500ms)

## Question 19:

How does reducing the number of nodes or VMs while keeping the total storage as it is in a cluster affect data shuffling?

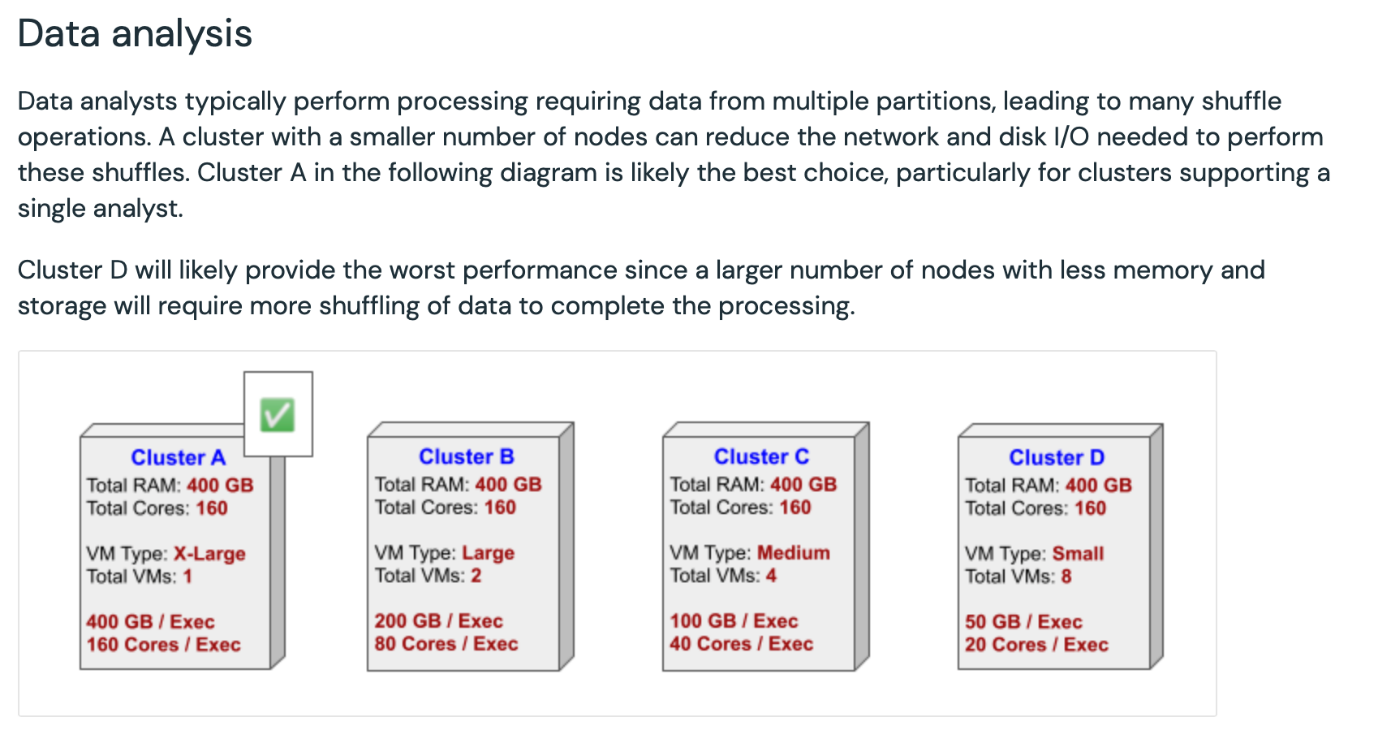
* **Reducing the number of nodes or VMs while keeping the total storage as it is will decrease the overall cluster performance.**
* **Reducing the number of nodes or VMs while keeping the total storage as it is will increase the network and disk I/O for shuffle operations.**
* **Reducing the number of nodes or VMs while keeping the total storage as it is will have no effect on shuffle operations.**
* **Reducing the number of nodes or VMs while keeping the total storage as it is can improve shuffle performance by reducing network and disk I/O.**
* **Reducing the number of nodes or VMs while keeping the total storage as it is will only affect CPU utilization during shuffle operations.**

**Explanation**

**Shuffling** is a process where data is **reorganized**, **grouped**, or **redistributed** across different nodes in a cluster. It is a **resource-intensive** operation that involves significant network and Disk I/O activities.

When we have a **smaller** number of **nodes** or **VMs** in a cluster, the amount of data that needs to be transferred over the network during shuffle operations is **reduced.** Additionally, since there are fewer nodes, the disk I/O required for reading and writing shuffled data is also **decreased.** This can lead to **faster** and more **efficient** shuffle operations, as there is **less** contention for **network bandwidth.**

***Hence, if a lot of shuffles involving transformations are needed to be performed, less number of nodes is recommended by Databricks.***



More Info: [Best practices for Cluster configurations](https://docs.databricks.com/en/clusters/cluster-config-best-practices.html#data-analysis:~:text=Data%20analysts%20typically,a%20single%20analyst.)

## Question 20:

An e-commerce company maintains its product information in a delta table named **inventory** with the following columns - **product\_id**, **product\_name**, **quantity**, **price** and **is\_active**. At the end of each month, they receive a new delta table named **updated\_inventory** with the same schema as the inventory table. Which of the following **MERGE INTO** statements should be used to update the **inventory** table based on the end-of-month updates given that if the **product\_id** is missing from the **updated\_inventory** table, the **is\_active** column in the **inventory** table should be turned to false?

* MERGE INTO inventory USING updated\_inventory

ON inventory.product\_id = updated\_inventory.product\_id

WHEN MATCHED THEN

UPDATE SET

inventory.quantity = updated\_inventory.quantity,

inventory.price = updated\_inventory.price

WHEN NOT MATCHED BY SOURCE THEN

INSERT (product\_id, product\_name, quantity, price, is\_active)

VALUES (updated\_inventory.product\_id, updated\_inventory.product\_name, updated\_inventory.quantity, updated\_inventory.price, true)

WHEN NOT MATCHED BY TARGET THEN

UPDATE SET is\_active = false;

* MERGE INTO inventory USING supplier\_updates

ON inventory.product\_id = updated\_inventory.product\_id

WHEN MATCHED THEN

UPDATE SET

inventory.quantity = updated\_inventory.quantity,

inventory.price = updated\_inventory.price

WHEN NOT MATCHED BY SOURCE THEN

INSERT (product\_id, product\_name, quantity, price)

VALUES (updated\_inventory.product\_id, updated\_inventory.product\_name, updated\_inventory.quantity, updated\_inventory.price)

WHEN NOT MATCHED THEN

UPDATE SET is\_active = false;

* MERGE INTO inventory USING updated\_inventory

ON inventory.product\_id = updated\_inventory.product\_id

WHEN MATCHED THEN

UPDATE SET

updated\_inventory.quantity = inventory.quantity,

updated\_inventory.price = inventory.price

WHEN NOT MATCHED THEN

INSERT (product\_id, product\_name, quantity, price, is\_active)

VALUES (updated\_inventory.product\_id, updated\_inventory.product\_name, updated\_inventory.quantity, updated\_inventory.price, false)

WHEN NOT MATCHED BY SOURCE THEN

UPDATE SET is\_active = true;

* MERGE INTO updated\_inventory USING inventory

ON updated\_inventory.product\_id = inventory.product\_id

WHEN MATCHED THEN

UPDATE SET

updated\_inventory.quantity = inventory.quantity,

updated\_inventory.price = inventory.price

WHEN NOT MATCHED THEN

INSERT (product\_id, product\_name, quantity, price, is\_active)

VALUES (inventory.product\_id, inventory.product\_name, inventory.quantity, inventory.price, true)

WHEN NOT MATCHED BY SOURCE THEN

UPDATE SET is\_active = false;

* **MERGE INTO inventory USING updated\_inventory**

**ON inventory.product\_id = updated\_inventory.product\_id**

**WHEN MATCHED THEN**

**UPDATE SET**

**inventory.quantity = updated\_inventory.quantity,**

**inventory.price = updated\_inventory.price**

**WHEN NOT MATCHED THEN**

**INSERT (product\_id, product\_name, quantity, price, is\_active)**

**VALUES (updated\_inventory.product\_id, updated\_inventory.product\_name, updated\_inventory.quantity, updated\_inventory.price, true)**

**WHEN NOT MATCHED BY SOURCE THEN**

**UPDATE SET is\_active = false;**

**(Correct)**

**Explanation**

In **MERGE INTO** statement,

The **WHEN MATCHED** clause is used to handle rows that **match** between the **target** and **source** tables. In **Option E**, it correctly **updates** the **quantity** and **price** columns in the **inventory** table with the values from the **updated\_inventory** table.

The **WHEN NOT MATCHED** clause is used to handle rows that **do not match** between the **target** and **source** tables. In **Option E**, it **inserts**new rows into the **inventory** table with the corresponding information from the **updated\_inventory** table. The **is\_active** column is set to **true** to indicate that the product is **active.**

The **WHEN NOT MATCHED BY SOURCE** clause handles rows that **exist** in the **target** table but **not** in the **source** table. In **Option E**, it updates the **is\_active** column of those rows to **false**, marking them as **inactive.**

Thus, the correct answer is **Option E**, i.e

1. MERGE INTO inventory USING updated\_inventory
2. ON inventory.product\_id = updated\_inventory.product\_id
3. WHEN MATCHED THEN
4. UPDATE SET
5. inventory.quantity = updated\_inventory.quantity,
6. inventory.price = updated\_inventory.price
7. WHEN NOT MATCHED THEN
8. INSERT (product\_id, product\_name, quantity, price, is\_active)
9. VALUES (updated\_inventory.product\_id, updated\_inventory.product\_name, updated\_inventory.quantity, updated\_inventory.price, true)
10. WHEN NOT MATCHED BY SOURCE THEN
11. UPDATE SET is\_active = false;

***EXAM TIP - Whenever the source and target schemas are similar, the*MERGE INTO*command should be used to update the target table.***

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%203/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: [Using MERGE INTO in Databricks](https://docs.databricks.com/en/sql/language-manual/delta-merge-into.html#:~:text=MERGE%20INTO%20target_table_name,%5B...%5D)

## Question 21:

Which property should be changed in an Auto Loader pipeline to enable the case-insensitive behavior of column names while iteratively reading Parquet files from an S3 location if the schema evolution mode is selected as **rescue** ?

* **No changes are required as the column names in Auto Loader are always inferred case-insensitively in rescue mode.**
* **The readerCaseSensitive option should be set to false**
* **The readerCaseSensitive option should be set to true**
* **The readerCaseInsensitive option should be set to false**
* **The readerCaseInsensitive option should be set to true**

**Explanation**

Let us look at each option one by one:

*No changes are required as the column names in Auto Loader are always inferred case-insensitively in****rescue****mode.*

**INCORRECT!** By default, the readerCaseSensitive option is set to **true** when the **rescue** mode is selected, meaning the inference for the column names is case-sensitive.

*The****readerCaseSensitive****option should be set to****false***

**CORRECT!** The readerCaseSensitive option controls the case-sensitive behavior of the column names in an Auto Loader pipeline. If the option is false, the column names will be inferred case-insensitively.

*The****readerCaseSensitive****option should be set to****true***

**INCORRECT!** The readerCaseSensitive option is, by default, set to true when the **rescue** mode is selected.

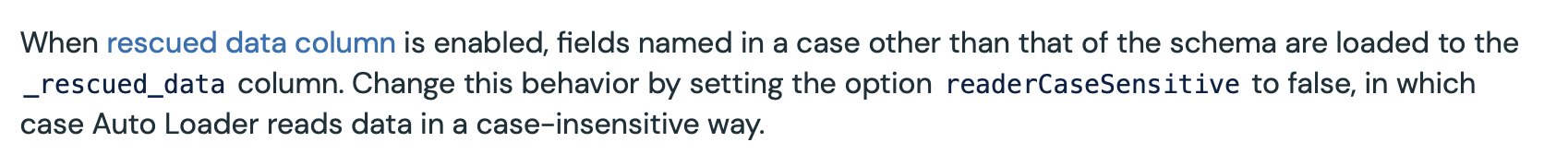
*The****readerCaseInsensitive****option should be set to****false***

**INCORRECT!** readerCaseInsensitive is not a valid option.

*The****readerCaseInsensitive****option should be set to****true***

**INCORRECT!** readerCaseInsensitive is an invalid option.

Also, note this option can only be used with AVRO, Parquet and CSV files.

More Info: [Case-sensitive behavior in Auto Loader](https://docs.databricks.com/en/ingestion/auto-loader/schema.html#change-case-sensitive-behavior:~:text=Change%20this%20behavior%20by%20setting%20the%20option%20readerCaseSensitive%20to%20false%2C%20in%20which%20case%20Auto%20Loader%20reads%20data%20in%20a%20case%2Dinsensitive%20way.)

## Question 22:

Which of the following is incorrect about the difference between using an existing all-purpose cluster and a new job cluster while creating a job in Databricks?

* **A new job cluster should be used if a dashboard needs to be updated after every 30 minutes.**
* **If a terminated all-purpose cluster is selected, the job owner should have at least Can Restart permission to run the job successfully.**
* **If you choose a new job cluster while creating a job, the cluster is created when the first task in the job starts.**
* **If a job cluster fails while the job is running, a new job cluster is created.**
* **If you choose a new job cluster while creating a job, the cluster is terminated when the last task using the cluster is completed.**

**Explanation**

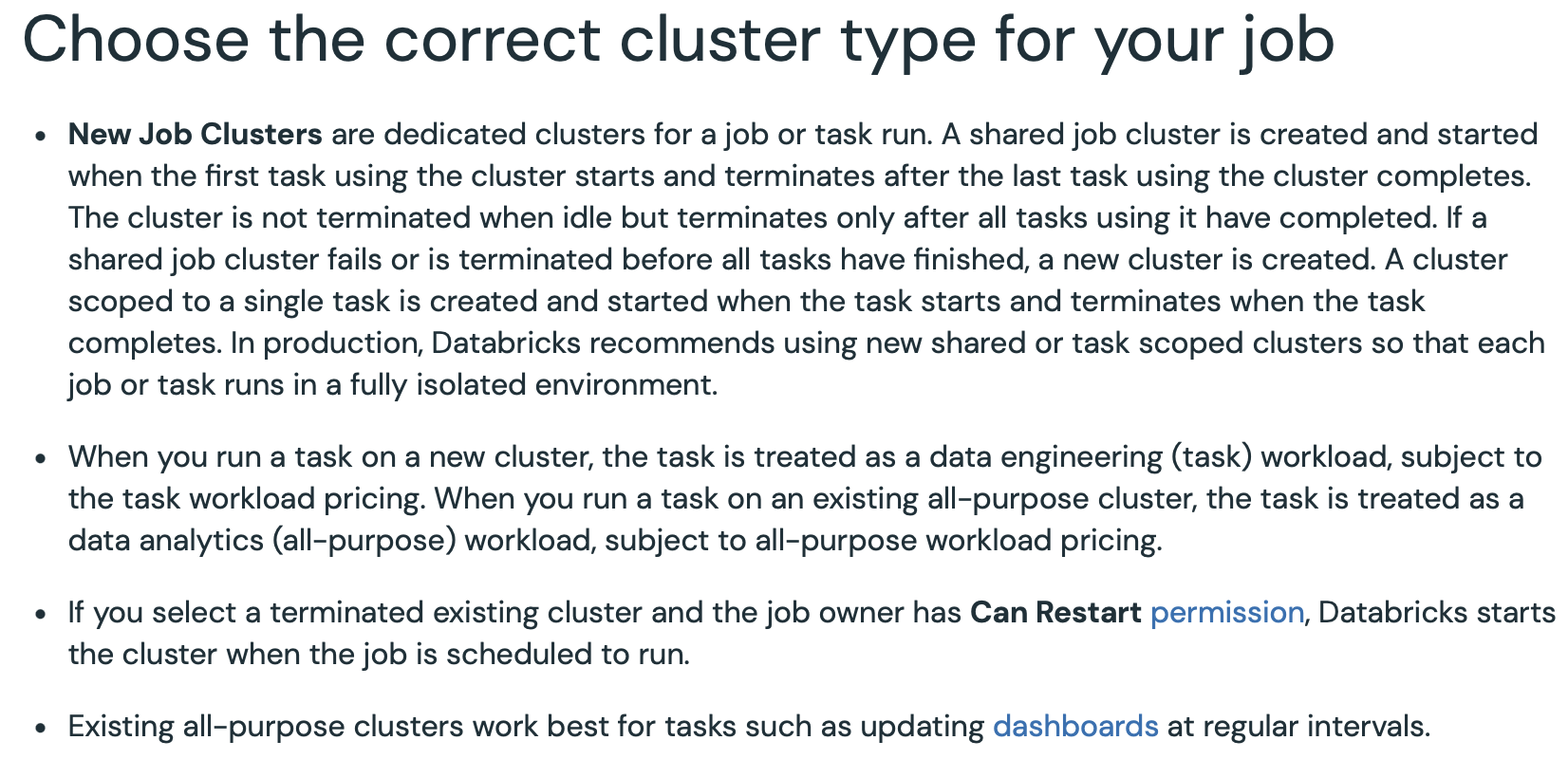
All of the above statements are true **except** the **first** one.

If you wish to perform tasks like **dashboard updates**, an **existing all-purpose cluster** is recommended for the following reasons:

**Resource Utilization**: All-purpose clusters are designed to handle a variety of tasks with varying resource requirements. Running dashboard updates typically doesn't require constant high resource usage, making it ideal to use existing clusters that are shared among multiple jobs.

**Efficiency**: Instead of creating new clusters for each dashboard update job, using an existing cluster avoids the overhead of cluster creation, resulting in more efficient utilization of computational resources.

**Cost Savings**: Since you're utilizing already available resources, it helps in optimizing costs by reducing the number of clusters needed.



More Info: [Choosing the cluster type for the job](https://docs.databricks.com/en/workflows/jobs/use-compute.html#choose-the-correct-cluster-type-for-your-job:~:text=New%20Job%20Clusters%20are,dashboards%20at%20regular%20intervals.)

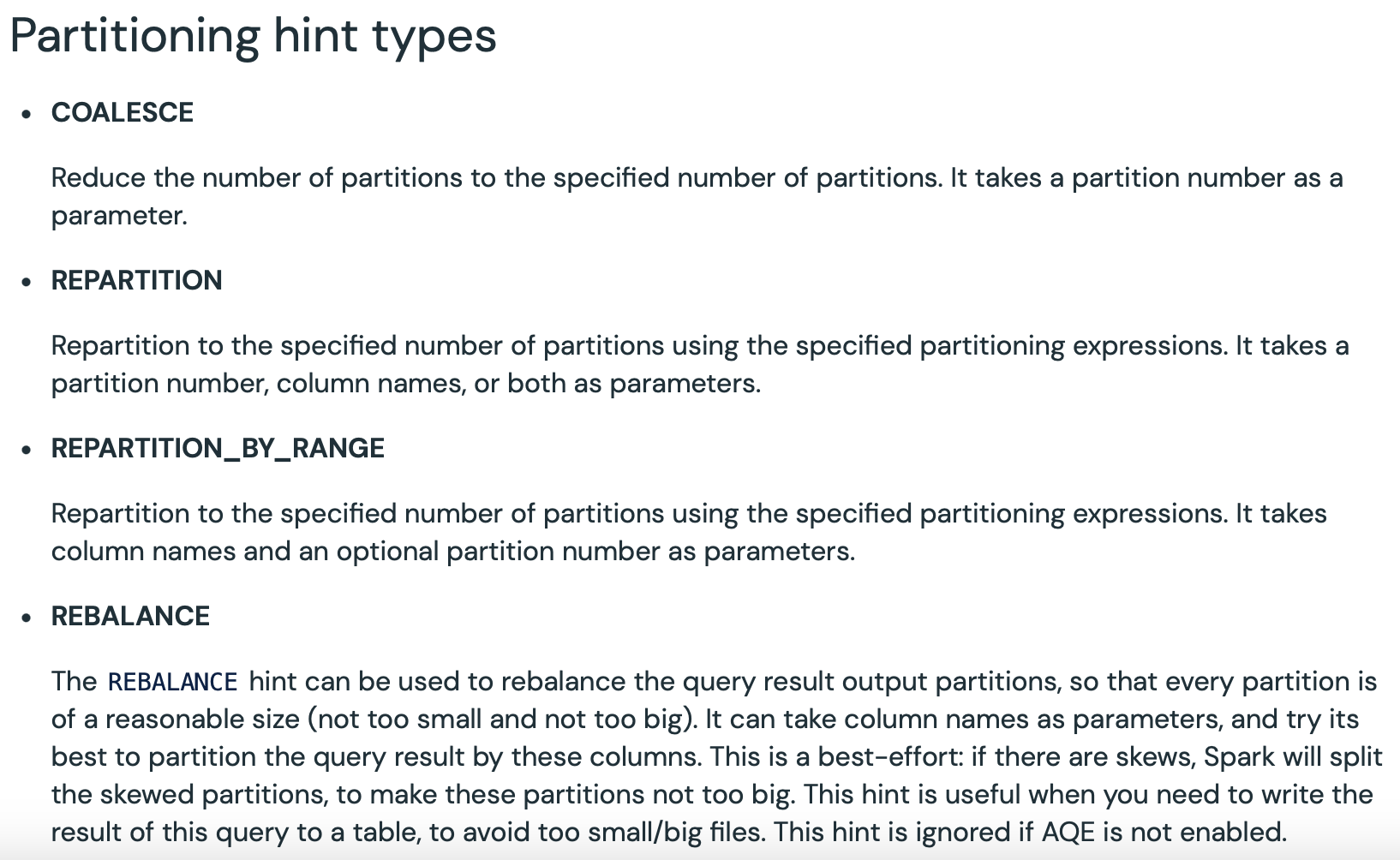
## Question 23:

Which of the following is not one of the partitioning hints in Spark?

* **COALESCE**
* **SHUFFLE\_HASH**
* **REPARTITION**
* **REPARTITION\_BY\_RANGE**
* **REBALANCE**

**Explanation**

**COALESCE**, **REPARTITION**, **REPARTITION\_BY\_RANGE** and **REBALANCE** are the types of **partitioning hints** in Spark whereas **SHUFFLE\_HASH** is a type of **Join hint.**



More Info: [Partitioning hints in Spark](https://docs.databricks.com/en/sql/language-manual/sql-ref-syntax-qry-select-hints.html#partitioning-hint-types)

## Question 24:

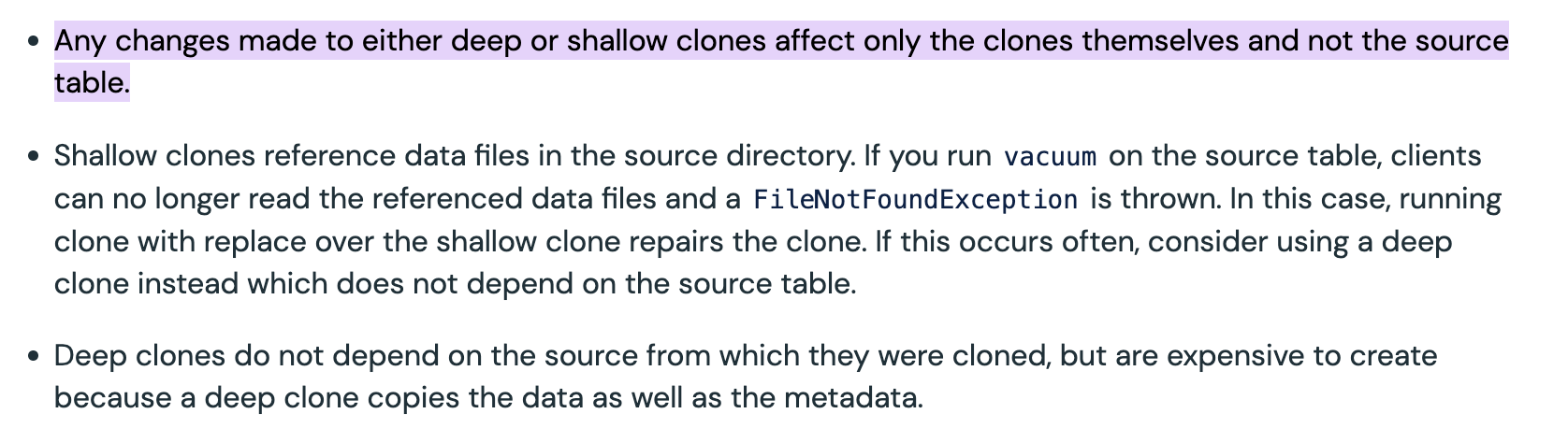
A deep clone of a delta table is created for testing the data. The original table is named **tickets** whereas the deep-cloned table is named **deep\_tickets** Another data engineer creates a delta table named **shallow\_tickets** by shallow cloning the newly created **deep\_tickets** table and writes some records in it. Which of the following is true about the data in the **tickets** table and the deep cloned **deep\_tickets** table.

* **The records added to the shallow\_tickets table will not be added to the tickets table but will be added to the deep\_tickets table.**
* **The records added to the shallow\_tickets table will be added to the tickets table but will not be added to the deep\_tickets table.**
* **The records added to the shallow\_tickets table will be added to the tickets table as well as the deep\_tickets table.**
* **The records added to the shallow\_tickets table will not be added to the tickets table or the deep\_tickets table.**
* **The records cannot be added to the shallow\_tickets table as it is a shallow clone of a deep-cloned 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 **fourth** option which states that the records added to the **shallow\_tickets** table will **not** be added to the **tickets** table or the **deep\_tickets** table.

More Info: [Adding data to the cloned tables](https://docs.databricks.com/en/delta/clone.html#what-are-the-semantics-of-delta-clone-operations:~:text=Any%20changes%20made%20to%20either%20deep%20or%20shallow%20clones%20affect%20only%20the%20clones%20themselves%20and%20not%20the%20source%20table)

## Question 25:

A data engineer executes the following SQL query to create a Delta table:

**CREATE TABLE data\_changes (\_change\_type STRING, \_changed\_by STRING)**

**TBLPROPERTIES (delta.enableChangeDataFeed = true);**

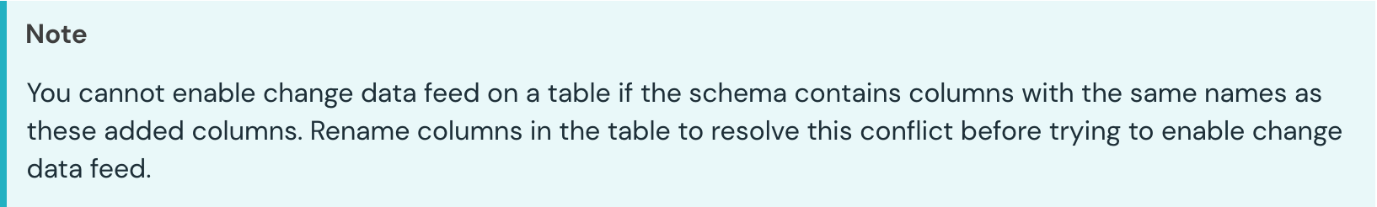
Which of the following describes the output of the above query?

* **The Delta table will be created successfully but Change Data Feed will not be enabled.**
* **The SQL statement will result in an error and the Delta table will not be created.**
* **The Delta table will be created successfully with the Change Data Feed enabled.**
* **The Delta table will be created successfully but the \_change\_type column will be removed.**
* **The SQL statement will not return any error but the Delta table will not be created.**

**Explanation**

As the **\_change\_type** column is reserved for **Change Data Capture**, the above query will result in the following error:

1. Error in SQL statement: DeltaIllegalStateException: Unable to enable Change Data Capture on
2. the table. The table already contains reserved columns [\_change\_type] that will be used
3. internally as metadata for the table's Change Data Feed. To enable Change Data Feed on the
4. table rename/drop these columns.



[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%203/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: [Schema for capturing Change Data Feed](https://docs.databricks.com/en/delta/delta-change-data-feed.html#what-is-the-schema-for-the-change-data-feed:~:text=You%20cannot%20enable%20change%20data%20feed%20on%20a%20table%20if%20the%20schema%20contains%20columns%20with%20the%20same%20names%20as%20these%20added%20columns.%20Rename%20columns%20in%20the%20table%20to%20resolve%20this%20conflict%20before%20trying%20to%20enable%20change%20data%20feed)

Bottom of Form

Top of Form

## Question 26:

The name and age of each person entering a museum are added manually in a CSV file. The following code block intends to read that CSV file and convert it into a DataFrame:

schema = StructType([StructField("name", StringType()), StructField("age", IntegerType())])

df = spark.read.format(“csv”)

.schema(schema)

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

.load(“/tmp/logs.csv”)

What should be replaced with the blank to ensure that the records having **NA** in the **age** column are not added to the DataFrame?

* **.option('mode', 'DROPMALFORMED')**
* **.option('mode', 'DROPNA')**
* **.option('drop', 'NA')**
* **.option('mode', 'PERMISSIVE')**
* **.option('mode', 'FAILFAST')**

**Explanation**

The following **three** modes can be selected while **reading** the data from the CSV files:

1. **PERMISSIVE** - If the data **cannot** be parsed properly, **nulls** replace the data. This is the **default** mode.
2. **DROPMALFORMED** - The rows having improper data are **dropped.**
3. **FAILFAST** - The command **fails** in case the data cannot be **parsed** properly.

In this question, the **age** column is of **IntegerType** whereas the **value** stored is **NA**, which is a **string**. So, **DROPMALFORMED** should be used to ensure that the **malformed** records are **dropped.**

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%203/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: [Handling malformed CSV records](https://docs.databricks.com/en/external-data/csv.html#work-with-malformed-csv-records:~:text=DROPMALFORMED%3A%20drops%20lines%20that%20contain%20fields%20that%20could%20not%20be%20parsed)

Bottom of Form

Top of Form

## Question 27:

Which of the following depicts the primary difference between the **withWatermark** and **window** functions in a streaming query?

* **The withWatermark function defines a time threshold for data to be considered late and discarded, while the window function defines a fixed-size time interval for grouping data.**
* **The withWatermark function is used for creating sliding time-based windows, while the window function is used for specifying watermark thresholds to manage late data.**
* **The withWatermark function and the window function serve the same purpose and can be used interchangeably in streaming queries.**
* **The window function defines a time threshold for data to be considered late and discarded, while the withWatermark function defines a fixed-size time interval for grouping data.**
* **The withWatermark function is used to define schema evolution for incoming data, while the window function is used for data aggregation within specified time intervals.**

**Explanation**

**withWatermark** Function:

The **withWatermark** function is used in streaming queries to handle **event-time-based** data. It defines a **watermark**, which is essentially a time threshold indicating up to what point in the event time the system should consider data as **"on time"**. Data with event times **earlier** than the watermark is considered **valid**, and data with event times **later** than the watermark is considered **"late"** and might be discarded. **This function helps manage late data in streaming processing.**

**window** Function:

The **window** function, on the other hand, is used for **creating windows** or **time intervals** within the streaming data. It allows you to **group** data based on time intervals, such as **fixed-size windows** or **sliding windows**. These windows help in performing **aggregate operations** on the **grouped data within each window interval**, allowing you to analyze the data within specific time segments.

More Info: [withWatermark function](https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.sql.DataFrame.withWatermark.html) | [window function](https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.sql.functions.window.html)

Bottom of Form

Top of Form

## Question 28:

A Kafka stream that acts as an upstream system in an ETL framework tends to produce duplicate values within a batch. The streaming query reads the data from the source and writes to the downstream delta table using the default trigger interval. If the upstream system emits the data every 20 minutes, which of the following strategies can be used to remove the duplicates before saving the data to the downstream delta table while keeping the costs low?

* **Use dropDuplicates method after every 20 minutes on the target table.**
* **Change the downstream table to a temporary table in the streaming query, drop the duplicates from the temporary table every 20 minutes, and load the data to the original downstream table.**
* **Update the processing time to 20 minutes and add dropDuplicates() in the streaming query.**
* **Adding dropDuplicates() to the streaming query will remove duplicate values from all previous batches of data.**
* **Add withWatermark method in the streaming query with 20 minutes as the argument.**

**Explanation**

Let us look at each option one by one:

***Use dropDuplicates method after every 20 minutes on the target table.***

**INCORRECT!** This can be **one of the solutions** but using **dropDuplicates()** every 20 minutes on the downstream table is **not a cost-effective way** of dropping the duplicates in a streaming query.

***Change the downstream table to a temporary table in the streaming query, drop the duplicates from the temporary table every 20 minutes, and load the data to the original downstream table.***

**INCORRECT!** This can also be**one of the solutions**but using a temporary table will **increase** the storage cost.

***Update the processing time to 20 minutes and add dropDuplicates() in the streaming query.***

**CORRECT!** Updating the **processing time** from the default(**500 ms**) to**20 minutes** assures that all the duplicates that are part of **one batch** can be easily **dropped** using the **dropDuplicates()** method.

***Adding dropDuplicates() to the streaming query will remove duplicate values from all previous batches of data.***

**INCORRECT!** The **dropDuplicates()** method alone will **not** remove the duplicates from the previous batches.

***Add withWatermark method in the streaming query with 20 minutes as the argument.***

**INCORRECT!** Firstly, adding **withWatermark()** requires a **time** **column** in the streaming source. Also, **dropDuplicates** should be added to drop the duplicate records within the **20-minute** timeframe.

More Info: [Using processing time triggers in a streaming query](https://docs.databricks.com/en/structured-streaming/triggers.html)

## Question 29:

Which of the following queries can be used by a data engineer to count all the records that were updated in the source table?

* spark.read.format("delta") \

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

.option("startingVersion", 0) \

.table("source") \

.where(col('\_change\_type') == 'update\_postimage') \

.count()

* spark.read.format("delta") \

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

.option("startingVersion", 0) \

.table("source") \

.where(col('\_change\_type') == 'update\_preimage') \

.count()

* spark.read.format("delta") \

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

.option("startingVersion", 0) \

.table("source") \

.where(col('\_change\_type') == 'update') \

.count()

* **All of the above**
* **Both A and B**

**Explanation**

Once a record is updated in a table with **Change Data Feed enabled**, it yields **two** records.

1. **update\_preimage** - value before the update
2. **update\_postimage** - value after the update

So, **any**of these two values can be used to **count** the number of **updates** made in the **source** table.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%203/Question%2029.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: [update\_preimage and update\_postimage in CDC](https://docs.databricks.com/en/delta/delta-change-data-feed.html" \l "what-is-the-schema-for-the-change-data-feed:~:text=preimage%20is%20the%20value%20before%20the%20update%2C%20postimage%20is%20the%20value%20after%20the%20update)

## Question 30:

The source path for an Auto Loader has been set as **s3://records/year=2023/month=11/season1.parquet** Which of the following partitioning details can be inferred by the above path?

* **The year column will be inferred as the partition column as Auto Loader supports partitioning only on one column.**
* **The month column will be inferred as the partition column as Auto Loader supports partitioning only on one column.**
* **As Auto Loader does not support partitioning inference, no column will be selected as partition column.**
* **Both year and month column will be inferred as the partition columns.**
* **A new column named year\_month will be created for partitioning.**

**Explanation**

As the above path contains **two**columnsfor **partitioning**, **Auto Loader** will infer both **year** and **month** columns as **partition** columns.

More Info: [Partition inference in Auto Loader](https://docs.databricks.com/ingestion/auto-loader/schema.html#:~:text=For%20example%2C%20the%20file%20path%20base_path/event%3Dclick/date%3D2021%2D04%2D01/f0.json%20results%20in%20the%20inference%20of%20date%20and%20event%20as%20partition%20columns)

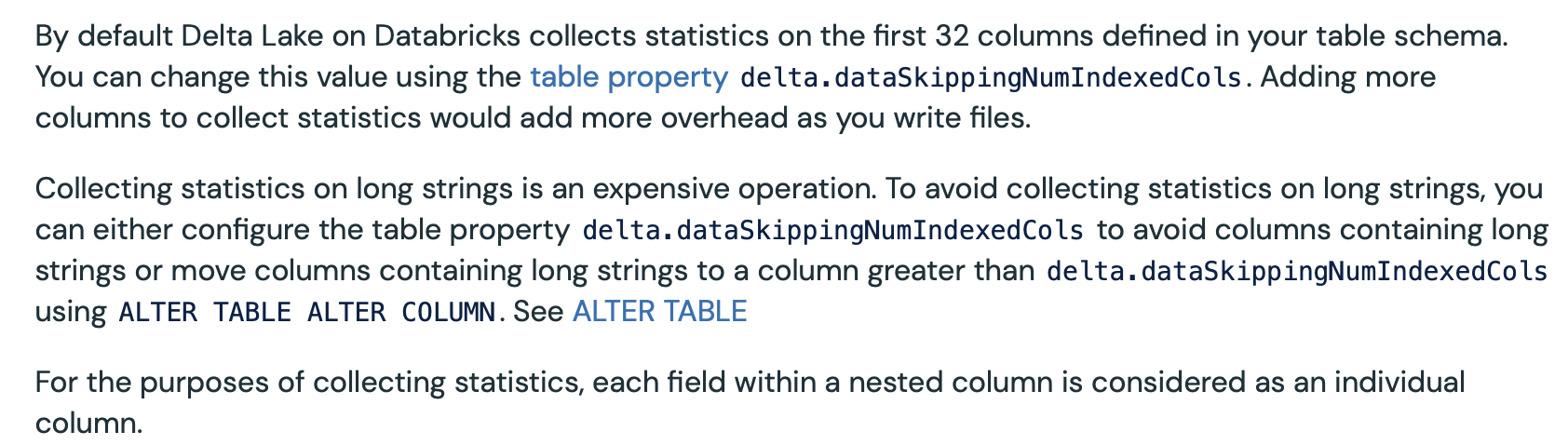
## Question 31:

A filter operation is the most common operation on a delta table named **voters** where columns number 58 and 59 are to be used for highly selective filters. Which of the following strategies can be used by the data engineer while creating the table to increase the query performance? The data engineer needs to keep the overhead lower as new records are added to the table.

* **Update the value of delta.dataSkippingNumIndexedCols to 59**
* **Statistics are collected for all the columns in a delta table, by default. Thus, nothing needs to be done.**
* **Re-order the columns by bringing columns 58 and 59 in the range of the first 16 columns as statistics are collected automatically for the first 16 columns only.**
* **Update the value of delta.dataSkippingNumIndexedCols to 2**
* **Re-order the columns by bringing columns 58 and 59 in the range of the first 32 columns as statistics are collected automatically for the first 32 columns only.**

**Explanation**

The **easiest** solution is to update the value of **delta.dataSkippingNumIndexedCols** to **59** to enable statistics collection for the **first 59 columns** but adding more columns will add more **overhead** as you add records to the delta table. Thus, the **best** way to collect statistics on columns 58 and 59 is to put the columns in the range of the **first 32 columns** as by default, the statistics are collected for the first 32 columns.



More Info: [Data Skipping in Databricks](https://docs.databricks.com/en/delta/data-skipping.html#:~:text=By%20default%20Delta,an%20individual%20column.)

## Question 32:

Which of the following constraints can be enforced on a delta table to prevent the addition of bad data?

* **CHECK**
* **NOT NULL**
* **DEFAULT**
* **A and B**
* **A, B and C**

**Explanation**

The delta table supports **CHECK** and **NOT NULL** constraints to **prevent** the addition of **bad data.** **ALTER** command can be used to add the **CHECK** and **NOT NULL** constraints.

For eg: **ALTER TABLE routes ADD CONSTRAINT negaitive\_distance CHECK (distance >= 0);**

Remember, if the **routes** table already contains some records where distance is **negative**, the above command will **fail.** Once you add a constraint, the **existing** records in the table are also **verified** for the condition. If the condition is **not**satisfied on the existing records, the **ADD CONSTRAINT** command will **fail.**



More Info: [Constraints on a Delta table](https://docs.databricks.com/en/tables/constraints.html#enforced-constraints-on-databricks:~:text=When%20a%20constraint,each%20input%20row.)

## Question 33:

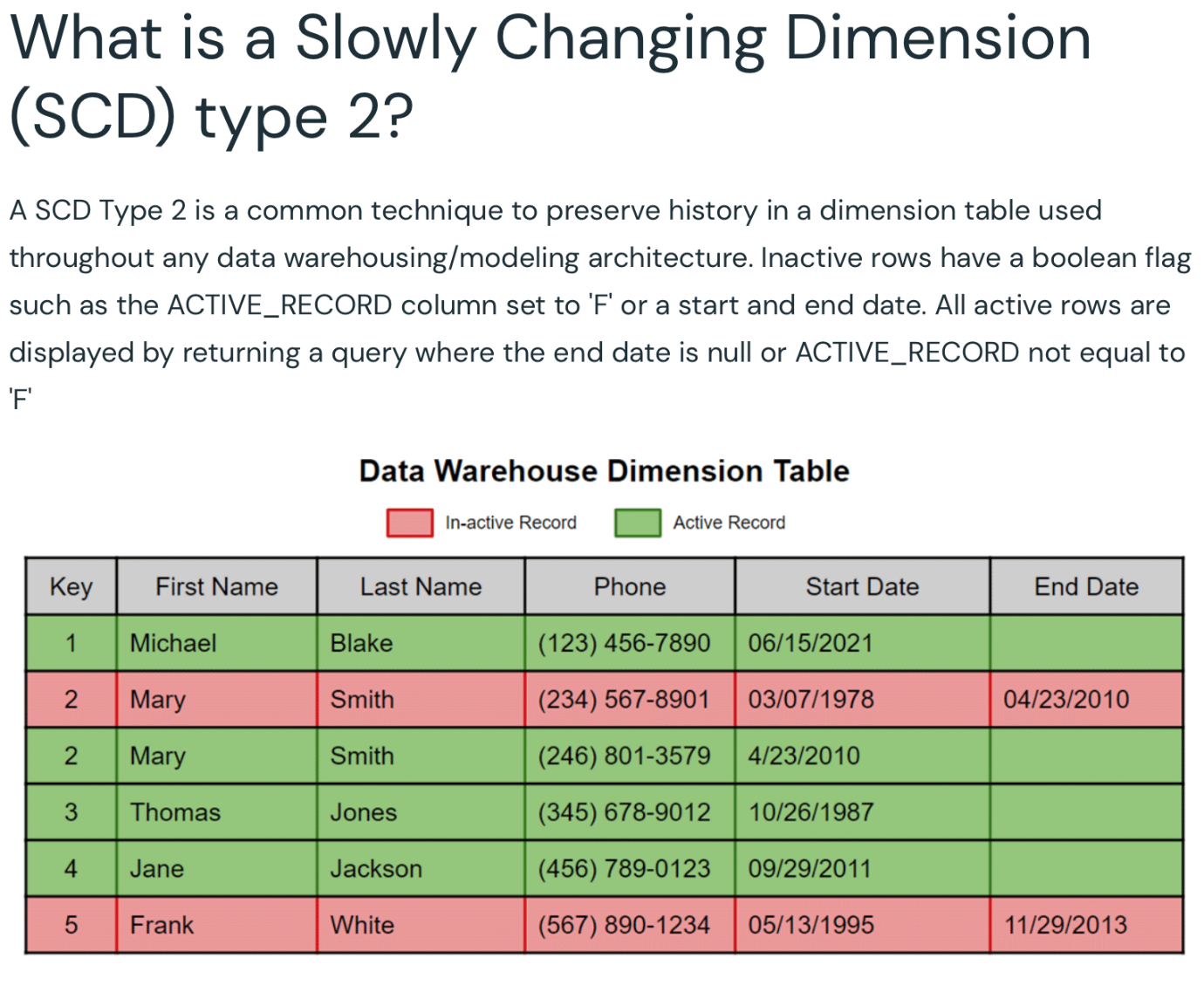
Which of the following SCD types is most likely to have **is\_active** column with values as **True** and **False**?

* **Type 0 SCD**
* **Type 1 SCD**
* **Type 2 SCD**
* **Type 3 SCD**
* **None of the above**

**Explanation**

Three of the most commonly used types of slowly changing dimensions 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 columns like **end\_date** or **is\_active**
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.

More Info: [Type 2 SCD](https://www.databricks.com/blog/2023/01/25/loading-data-warehouse-slowly-changing-dimension-type-2-using-matillion.html#:~:text=What%20is%20a,equal%20to%20%27F%27)

## Question 34:

A delta table named **currency\_conversion** is updated daily at midnight by the central banking system. The table contains the conversion rates between all the currencies available in the banking system. The banks want to access the latest conversion rate whenever the users execute a transaction. Which of the following is the most efficient method that can be implemented by the central banking system for sharing the conversion rates to all the banks without giving access to the original table?

* **Create a DEEP CLONE of the original table**
* **Manually send the data to all the banks at midnight**
* **Create a VIEW from the original table**
* **Create a SHALLOW CLONE of the original table**
* **Create a new table using the CTAS command**

**Explanation**

Creating a **new table** using the original table will involve **truncating** the previous data and adding the latest data on a daily basis. This will **increase** the manual effort and hence, will **not** be an efficient way to make the data available to the banks. So, a **view** can be created using the **original table**. Once the view is **registered**, the banks can query the view to get the **latest** conversion rates.

More Info: [Views in Databricks](https://docs.databricks.com/data-governance/unity-catalog/create-views.html#:~:text=A%20view%20is%20a%20read%2Donly%20object%20composed%20from%20one%20or%20more%20tables%20and%20views%20in%20a%20metastore.)

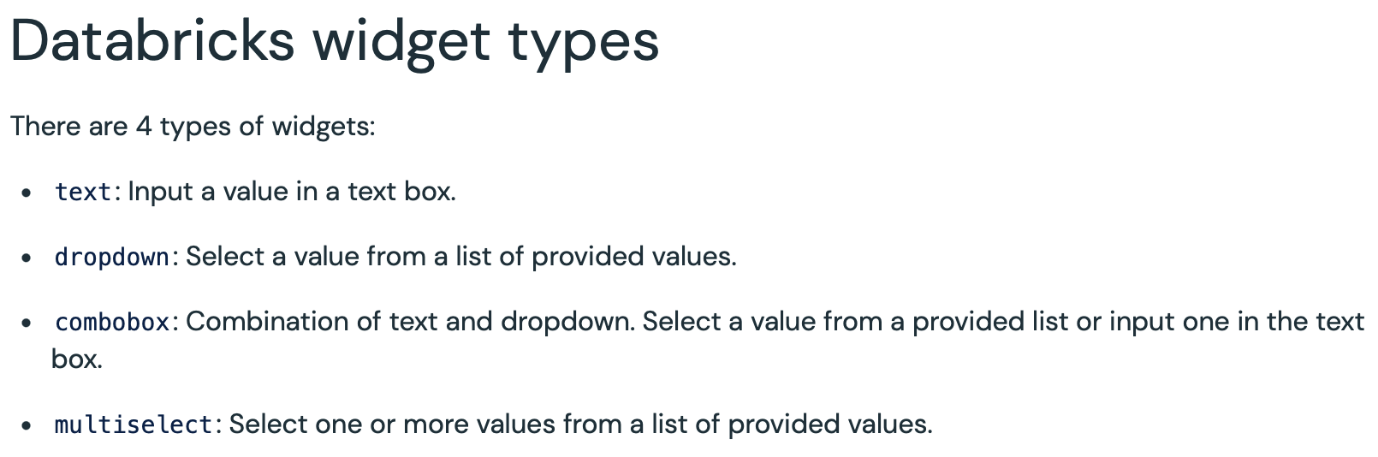
## Question 35:

Which of the following is not a type of widget supported in Databricks?

* **text**
* **dropdown**
* **combobox**
* **list**
* **multiselect**

**Explanation**

4 types of **widgets** available in Databricks are - **text**, **dropdown**, **combobox** and **multiselect**



The following commands can be used to create each of the above-mentioned widgets:

**dbutils.widgets.text("environment", "dev")**

**dbutils.widgets.dropdown("language", "Python", ["Python", "Scala", "SQL", “R"])**

**dbutils.widgets.combobox("cloud", "AWS", ["GCP", "Azure", “AWS"])**

**dbutils.widgets.multiselect("database", "SQL Server", ["SQL Server","PostgreSQL", "MySQL", "MongoDB"])**



[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%203/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: [Widgets in Databricks Notebooks](https://docs.databricks.com/en/notebooks/widgets.html#databricks-widget-types:~:text=There%20are%204%20types%20of%20widgets%3A,values%20from%20a%20list%20of%20provided%20values)

Bottom of Form

Top of Form

## Question 36:

Which of the following visualization types is not supported in Databricks?

* **Bar Chart**
* **Pie Chart**
* **Word Cloud**
* **Heat Map**
* **All of the above visualizations are supported in Databricks**

**Explanation**

**All** of the above-mentioned visualizations are **supported** in Databricks.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%203/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: [Bar Chart](https://docs.databricks.com/en/visualizations/visualization-types.html#bar-chart) | [Pie Chart](https://docs.databricks.com/en/visualizations/visualization-types.html#pie-charts) | [Word Cloud](https://docs.databricks.com/en/visualizations/visualization-types.html#word-cloud) | [Heat Map](https://docs.databricks.com/en/visualizations/visualization-types.html#heatmap-chart)

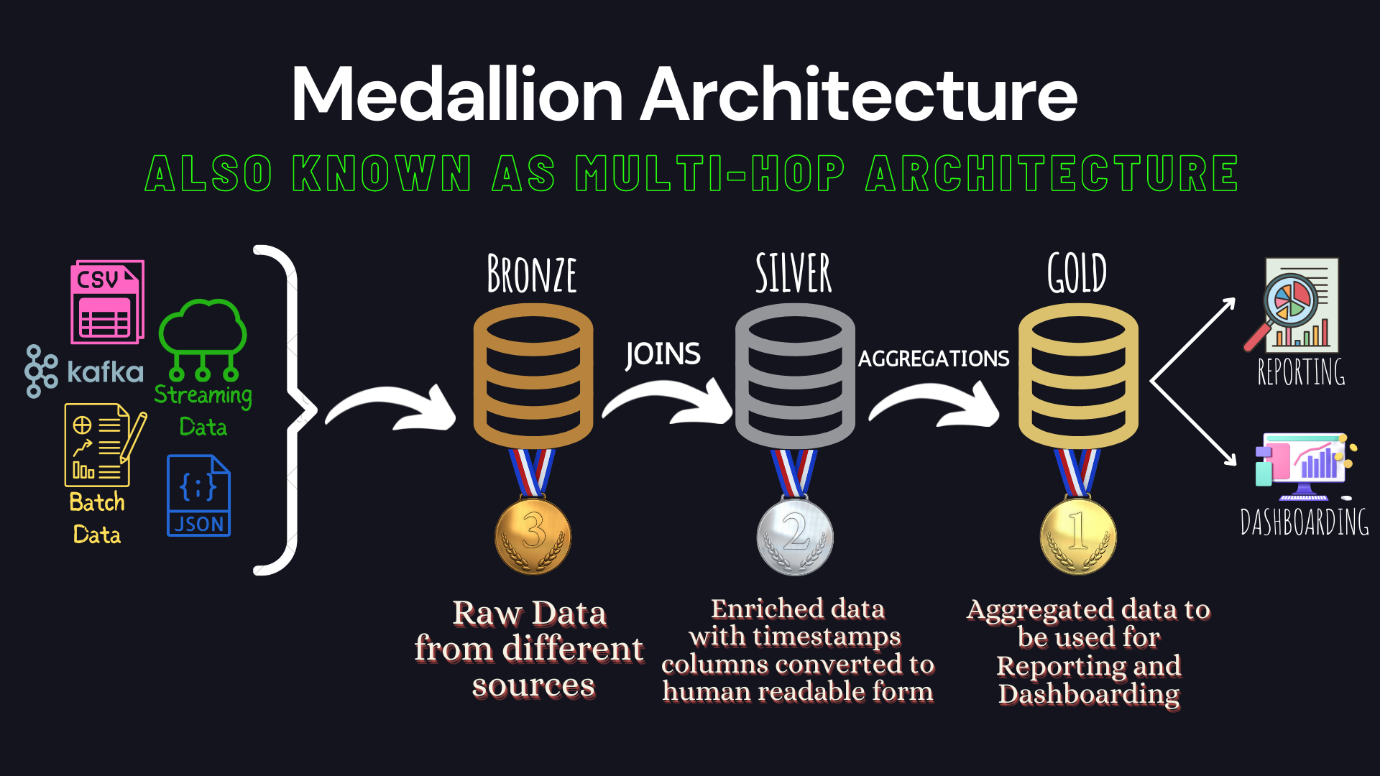
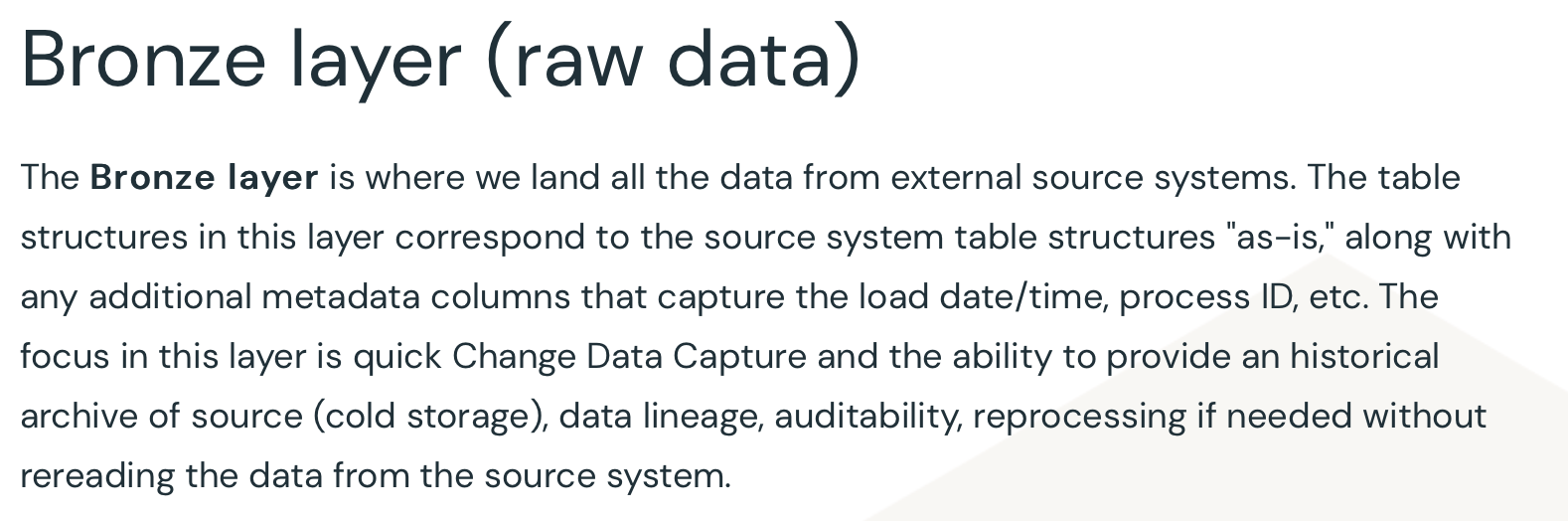
## Question 37:

What is the primary benefit of using the Bronze layer in a multi-hop or medallion architecture in Databricks?

* **It accelerates real-time data processing.**
* **It stores only processed and aggregated data.**
* **It maintains data lineage and historical records.**
* **It enforces data access control and security.**
* **It provides a user-friendly interface for data analysis.**

**Explanation**

The primary benefit of using the **Bronze layer** in a multi-hop or medallion architecture in Databricks is to maintain **data lineage** and **historical records.** This layer stores raw, unprocessed data as it is ingested, which allows **tracking the history** of the data and **understanding** **its** **transformations** over time. Data lineage is crucial for auditing, troubleshooting, and ensuring data quality in the analytics pipeline. It also helps in creating a **replayable** history of data.



More Info: [Using the Bronze layer to create a replayable history of data](https://www.databricks.com/glossary/medallion-architecture#:~:text=The%20Bronze%20layer,the%20source%20system.)

## Question 38:

The following command intends to add a check constraint on the **routes** table but fails to execute:

**ALTER TABLE routes ADD CHECK (distance > 0)**

As a data engineer, which of the following changes will you suggest to make the above command work?

* **CHECK keyword should be replaced by CONSTRAINT**
* **The TABLE keyword should be removed.**
* **ADD CHECK should be replaced by ADD CONSTRAINT CHECK**
* **(distance > 0) should be replaced by distance > 0, i.e. the brackets around the condition should be removed.**
* **ADD CHECK should be replaced by ADD CONSTRAINT constraint\_name CHECK**

**Explanation**

While adding a **CHECK** constraint on the delta table, the **constraint name** should be provided, which means **ADD CHECK** should be replaced by **ADD CONSTRAINT constraint\_name CHECK** to make the command work i.e.

**ALTER TABLE routes ADD CONSTRAINT negative\_distance CHECK (distance > 0);**

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%203/Question%2038.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 CHECK constraint](https://docs.databricks.com/en/sql/language-manual/sql-ref-syntax-ddl-alter-table-add-constraint.html#examples:~:text=ALTER%20TABLE%20pets%20ADD%20CONSTRAINT%20pets_name_not_cute_chk%20CHECK%20(length(name)%20%3C%2020)%3B)

## Question 39:

When selecting a partition column for optimizing data storage and query performance in Databricks, which of the following considerations should be taken into account?

* **Choose a column with a low cardinality that evenly distributes data.**
* **Select a column with high cardinality to reduce the number of partitions.**
* **Opt for a column with frequent updates to improve partition pruning.**
* **Prioritize a column with complex data types for better organization.**
* **Use a column unrelated to the data to enhance partitioning efficiency.**

**Explanation**

Let us look at each option one by one:

***Choose a column with a low cardinality that evenly distributes data.***

**CORRECT!** Low cardinality means the column has a limited number of distinct values. When data is evenly distributed across partitions, it helps in efficient storage and query performance as partition pruning can eliminate unnecessary partitions during queries. For example, In a dataset of online orders, the **order\_status** column with values like **Pending**,**Shipped** and **Delivered** has low cardinality and evenly distributes data across partitions. This choice can lead to effective partition pruning and improved performance.

***Select a column with high cardinality to reduce the number of partitions.***

**INCORRECT!** High cardinality columns have a large number of distinct values. They can lead to uneven distribution and inefficient storage, as well as increased overhead in managing a large number of partitions.

***Opt for a column with frequent updates to improve partition pruning.***

**INCORRECT!** Frequent updates to a partition can actually make partition pruning less effective. Partition pruning works by skipping unnecessary partitions during queries. If a partition is frequently updated, it might require more effort to track changes and maintain accurate pruning metadata.

***Select a column with complex data types for better organization.***

**INCORRECT!** Choosing a partition column with complex data types (like arrays or maps) might complicate the partitioning process and hinder query performance. Using simple data types for partitioning is generally recommended, as they are easier to manage and query.

***Choose a column unrelated to the data to enhance partitioning efficiency.***

**INCORRECT!** Partition columns should be relevant to the data and how it is queried. Using an unrelated column for partitioning won't improve efficiency and might lead to confusion and incorrect results.

More Info: [Choosing the correct partition column](https://docs.databricks.com/en/tables/partitions.html)

## Question 40:

Which of the following is the correct usage of row-level security in Databricks using dynamic views?

* CREATE VIEW restricted\_address AS

SELECT cost

CASE WHEN is\_member('analyst') THEN address

ELSE 'RESTRICTED'

END AS address

FROM employees;

* + CREATE VIEW restricted\_address AS

SELECT cost

CASE WHEN is\_account\_group\_member('analyst') THEN address

ELSE 'RESTRICTED'

END AS address

FROM employees;

* CREATE VIEW restricted\_address AS

SELECT cost, address

FROM employees

WHERE is\_member('analyst') or address like %AUS%;

* CREATE VIEW restricted\_address AS

SELECT cost, address

FROM employees

WHERE cost>=25000;

**All of the above options signify row-level security in Databricks.**

**Explanation**

Before moving to the answer you will need to understand **column-level**security and **row-level**security using **dynamic views** in Databricks.

Databricks supports the following three functions related to dynamic views:

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**.

**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/en/data-governance/unity-catalog/create-views.html#create-a-dynamic-view:~:text=is_account_group_member()%3A%20Returns%20TRUE%20if%20the%20current%20user%20is%20a%20member%20of%20a%20specific%20account%2Dlevel%20group.%20Recommended%20for%20use%20in%20dynamic%20views%20against%20Unity%20Catalog%20data.)**. You might still see is\_member()** **in the actual exam as this is one of the recent changes done by Databricks.**

Coming back to the question, let's understand each option in detail:

CREATE VIEW restricted\_address AS

SELECT cost

CASE WHEN is\_member('analyst') THEN address

ELSE 'RESTRICTED'

END AS address

FROM employees;

**INCORRECT!** This is **column-level** security. This query **restricts** the users from groups other than **analyst** from viewing the **addresses** of the **employees**. If a **non-member** of the **analyst** group executes the above query, they will see the values in the **address** column as **'RESTRICTED'**.

CREATE VIEW restricted\_address AS

SELECT cost

CASE WHEN is\_account\_group\_member('analyst') THEN address

ELSE 'RESTRICTED'

END AS address

FROM employees;

**INCORRECT!** This query also emulates **column-level** security.

CREATE VIEW restricted\_address AS

SELECT cost, address

FROM employees

WHERE is\_member('analyst') or address like %AUS%;

**CORRECT!** This query is the correct usage of **row-level** security. If a member of the **analyst** group executes the above query, **all the rows** will be returned from the view whereas if the query is executed by a **non-member**, only the rows having **'AUS'** in the **address** column will be returned.

CREATE VIEW restricted\_address AS

SELECT cost, address

FROM employees

WHERE cost>=25000;

**INCORRECT!** This is **neither** row-level security **nor** column-level security. In this query, only a **filter** condition is applied to the **cost** column. To add the security, you would need to add either **is\_member(group)** or **is\_account\_group\_member(group)** in the query.

**EXAM TIP:** In **row-level** security, the functions like **is\_member()** are used in the **WHERE** clause which helps in filtering the rows. Whereas in**column-level**security, the functions are used while **defining** the **column.**

More Info: [Row-level security in Databricks](https://docs.databricks.com/data-governance/unity-catalog/create-views.html#:~:text=CREATE%20VIEW%20sales_redacted%20AS%0A%20SELECT%0A%20%20%20user_id%2C%0A%20%20%20country%2C%0A%20%20%20product%2C%0A%20%20%20total%0A%20FROM%20sales_raw%0A%20WHERE%0A%20%20%20CASE%0A%20%20%20%20%20WHEN%20is_account_group_member(%27managers%27)%20THEN%20TRUE%0A%20%20%20%20%20ELSE%20total%20%3C%3D%201000000%0A%20%20%20END%3B)

## Question 41:

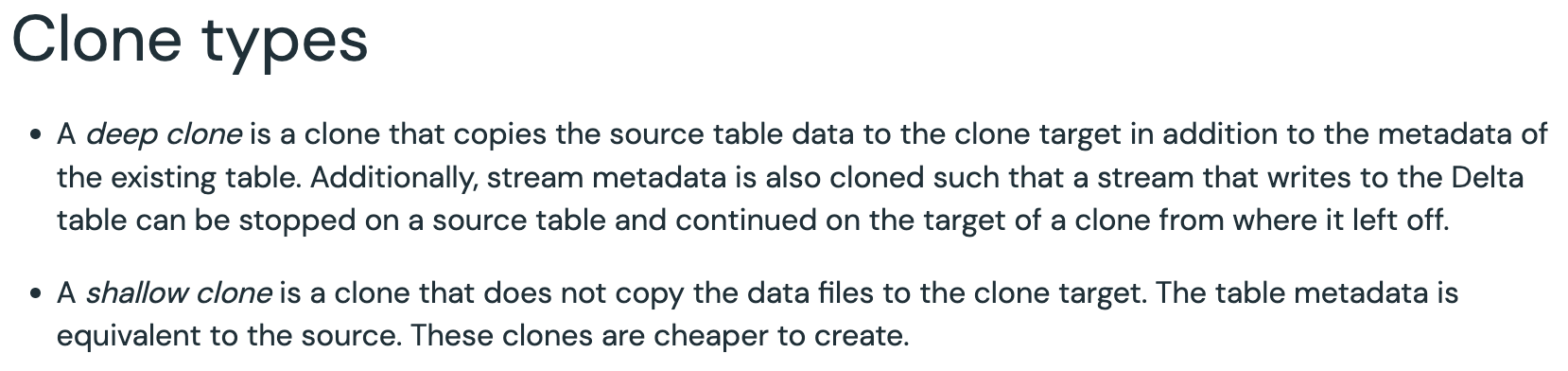
Which of the following explains the difference in querying the deep and shallow clones of a delta table once the original table is dropped?

* + **The original table can only be dropped after dropping the shallow and deep clones.**
  + **Querying both the shallow and deep clones will result in an error.**
  + **The shallow clone can be queried but querying the deep clone will result in an error.**
  + **Both the clones can be queried without facing any errors.**
  + **The deep clone can be queried but querying the shallow clone will result in an error.**

**Explanation**

When a **deep clone** is created over a table **all the files are copied to the new location** while in the case of a **shallow clone,** the **files from the original table are referenced.** Thus, creating a shallow clone is **faster** and **cost-effective.**

But if the **original table** is **dropped,** the shallow clone **loses** the files it refers to. So, querying the **shallow clone** of a managed delta table, after the original table is dropped will result in **FileNotFoundException**



[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%203/Question%2041.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: [Shallow Clone vs. Deep Clone in Databricks](https://docs.databricks.com/en/delta/clone.html#clone-types:~:text=A%20shallow%20clone%20is%20a%20clone%20that%20does%20not%20copy%20the%20data%20files%20to%20the%20clone%20target)

## Question 42:

Which of the following explains the difference between using the **LOCATION** keyword while creating a database and a table?

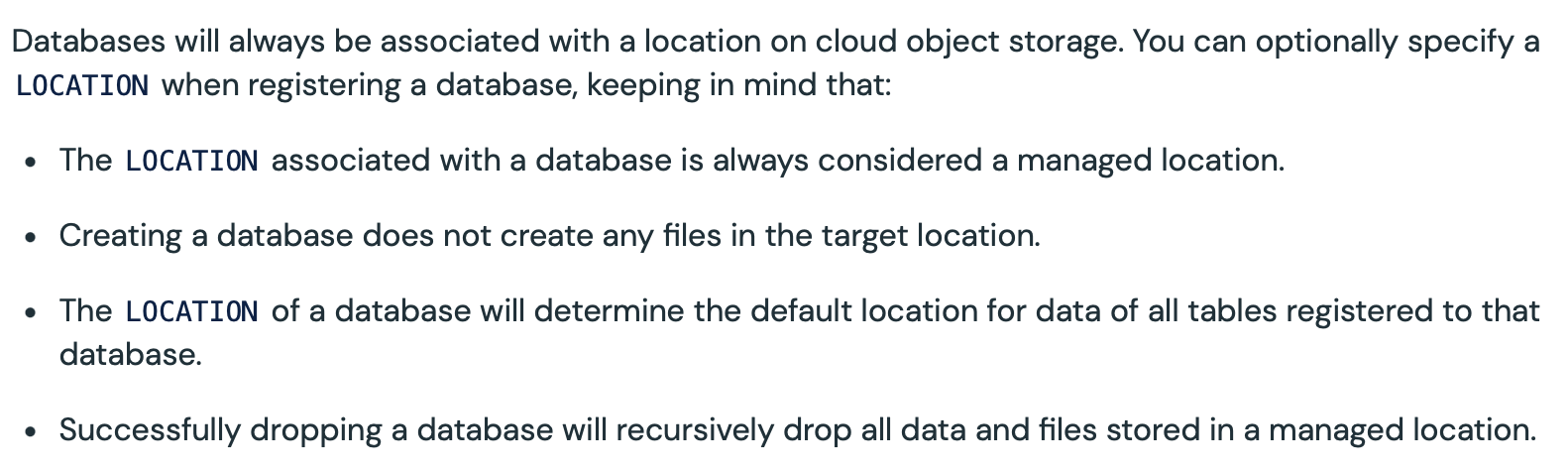
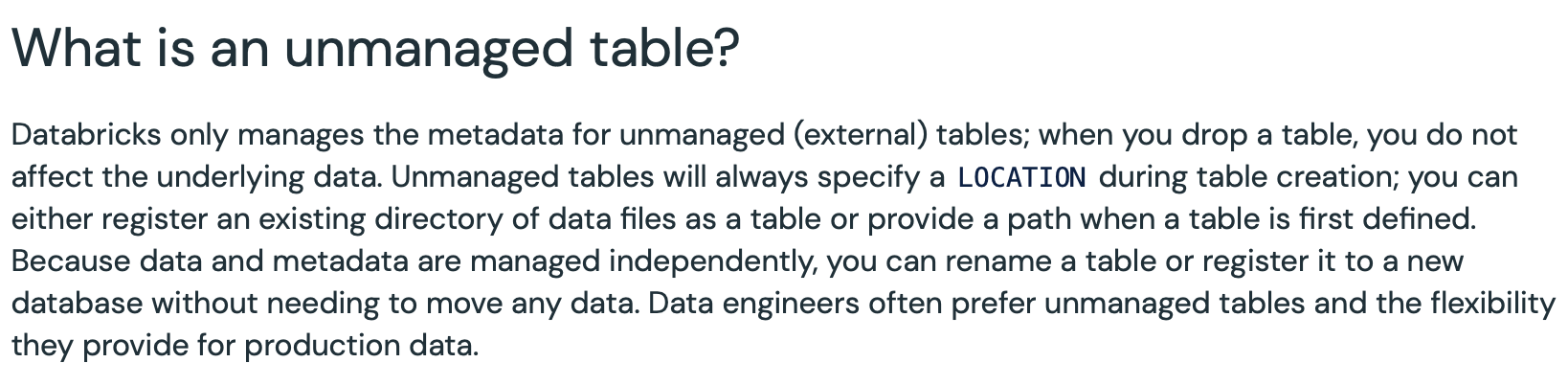
* + **Using the LOCATION keyword while creating a database or a table will mark them as unmanaged or external.**
  + **Using the LOCATION keyword while creating a database will mark it as unmanaged while using it with the CREATE TABLE command will mark the table as managed.**
  + **Using the LOCATION keyword with CREATE TABLE and CREATE DATABASE commands will have no effect on them.**
  + **Using the LOCATION keyword while creating a table will mark the table as unmanaged while using it with the CREATE DATABASE command will add the location where the underlying tables of the database will be stored.**
  + **LOCATION keyword cannot be used with the CREATE DATABASE command whereas it can be used while creating a table.**

**Explanation**

**LOCATION** keyword can be added while creating a database or a table but the **effect varies.**

The **LOCATION** keyword is added to the **CREATE DATABASE** command just to add the **path** where the data objects for the database will be stored.

Whereas the **LOCATION** keyword added to the **CREATE TABLE**command marks the table as **unmanaged** or **external.** Dropping a table created using the **LOCATION** keyword will drop **only the table** whereas the underlying data remains as is.



More Info: [Using LOCATION with CREATE TABLE](https://docs.databricks.com/en/lakehouse/data-objects.html#what-is-an-unmanaged-table:~:text=Databricks%20only%20manages,is%20first%20defined) | [Using LOCATION with CREATE DATABASE](https://docs.databricks.com/en/lakehouse/data-objects.html#what-is-a-database:~:text=Databases%20will%20always%20be,in%20a%20managed%20location)

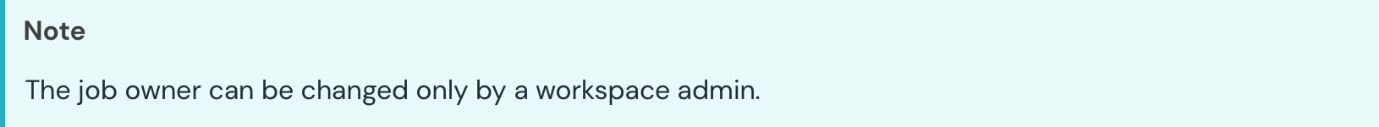
## Question 43:

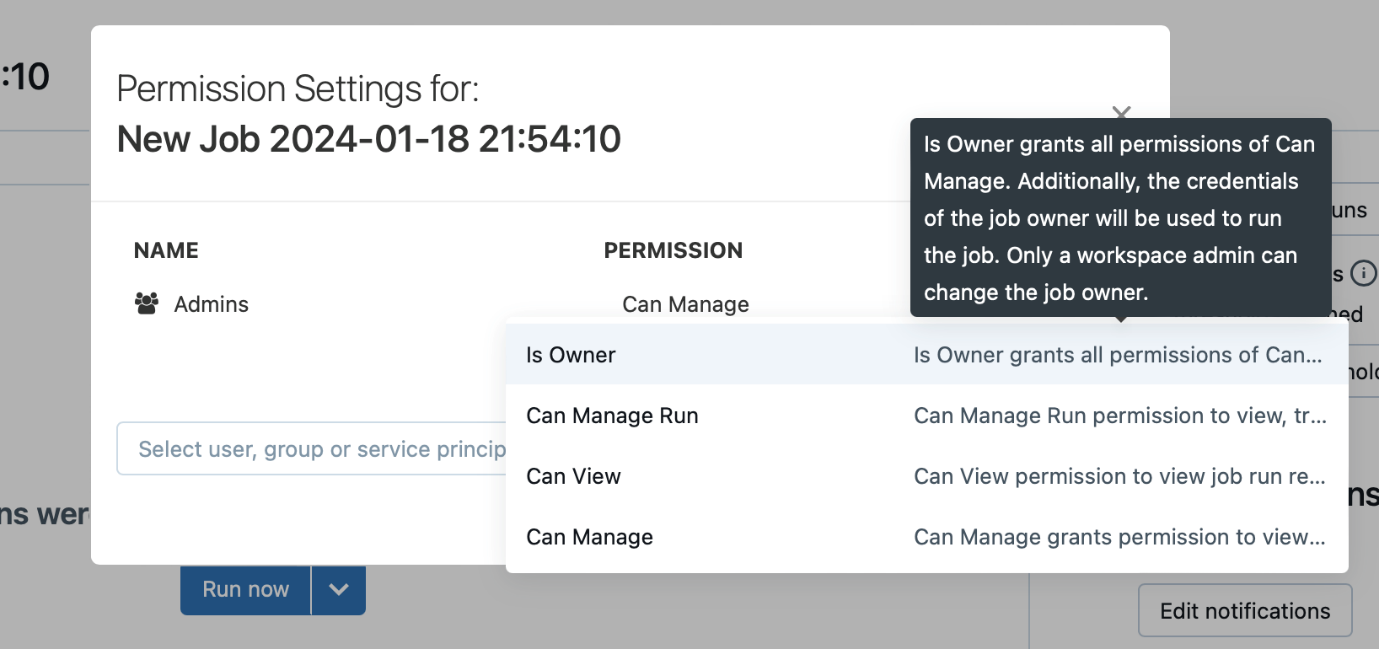
A data engineer needs to transfer the ownership of the job to another user. Which of the following statements is true about the ownership transfer?

* **The ownership of a job can be changed only by the Admin or the creator of the job.**
* **The ownership of a job can be changed only by the creator of the job.**
* **The ownership of a job can be changed by any user having Can Manage permission on the job.**
* **The ownership of a job can be changed only by the Admin.**
* **The ownership of a job can be changed by the Admin or the creator of the job or the user having Can Manage permission on the job.**

**Explanation**

In simple terms, only the **admin** can **transfer** the **ownership** of a job. Also note, that even the **creator** of the job **cannot** change the **ownership** of a job. Moreover, the ownership of a job **cannot** be transferred to a **group**, which means that a **job owner can never be a group**.





More Info: [Changing the owner of a Job](https://docs.databricks.com/en/security/auth-authz/access-control/jobs-acl.html#job-owner:~:text=Only%20workspace%20admins%20can%20change%20the%20job%20owner%20and%20Is%20Owner%20cannot%20be%20granted%20to%20a%20group.)

## Question 44:

A newly appointed data engineer needs **USAGE** privileges on the following tables:

Table 1: **policy** table from the **insurance** database

Table 2: **demographics** table from **customers** database

Which of the following statements is correct about the result of the execution of the following query:

**GRANT USAGE ON TABLES customers.demographics, insurance.policy TO new\_user;**

* **TABLES should be replaced by TABLE to execute the query correctly.**
* **The query will fail as privileges on multiple tables cannot be granted in a single GRANT statement.**
* **The query will be successful, the new\_user will have the requested privileges.**
* **The query will not be successful as the GRANT statement cannot be used to grant permissions on tables.**
* **The query will be executed without errors, but the grants will be given on only the first table.**

**Explanation**

Granting **permissions** on **tables**, **views**, **schemas** etc is common practice once a new member is added to a team. Similarly, **revoking** is also one of the essential activities if the user needs to be **restricted** the access to the objects like **tables** and **databases.**

But, the statements like **GRANT** and **REVOKE** can only be performed on **one object at a time.** To enable multiple **GRANTS** or **REVOKES**, you will need to run multiple statements.

Similarly, the above statement can be executed as:

**GRANT USAGE ON TABLE customers.demographics TO new\_user;**

**GRANT USAGE ON TABLE insurance.policy TO new\_user;**

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%203/Question%2044.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: [Object privileges in Databricks](https://docs.databricks.com/en/data-governance/table-acls/object-privileges.html#how-do-i-grant-a-user-or-group-privileges-on-multiple-tables-at-once:~:text=A%20grant%2C%20deny%2C%20or%20revoke%20statement%20can%20be%20applied%20to%20only%20one%20object%20at%20a%20time.%20The%20recommended%20way%20to%20organize%20and%20grant%20privileges%20on%20multiple%20tables%20to%20a%20principal%20is%20via%20schemas.)

## Question 45:

A databricks engineer needs to create a secret scope to store a connection string to connect the application to a JDBC source. In which of the following ways the data engineer can create the secret scope?

* + **The secret scope can only be created using Databricks CLI.**
  + **Only Databricks REST API can be used to create a secret scope.**
  + **dbutils.secrets() is the only way to create a secret scope in Databricks.**
  + **Databricks CLI, Databricks REST API and dbutils.secrets() can all be used to create a secret scope.**
  + **Only Databricks CLI and Databricks REST API can be used to create secret scope.**

**Explanation**

Let us see each option one by one:

*The secret scope can only be created using Databricks CLI.*

**INCORRECT!** A secret scope can be used by **Databricks CLI**as well as **Databricks REST API.** To create a secret scope named **connection** using a **Databricks CLI** you can use the following command:

**Databricks secrets create-scope —scope connection**

Also, to create a secret scope using the **Databricks REST API** you can use the **2.0/secrets/scopes/create** endpoint.

*Only Databricks REST API can be used to create a secret scope.*

**INCORRECT!** As discussed in the previous option, the secret scope can be created using **both** Databricks REST API and Databricks CLI.

***dbutils.secrets()****is the only way to create a secret scope in Databricks.*

**INCORRECT!** A secret scope **cannot** be created using **dbutils.secrets()** method. It can only be used to **read** an **already stored** secret in Databricks.

*Databricks CLI, Databricks REST API and****dbutils.secrets()****can all be used to create a secret scope.*

**INCORRECT!** As discussed in the previous option, **dbutils.secrets()** **cannot** be used to create a secret scope. It can **only** be used to **read** a secret.

*Only Databricks CLI and Databricks REST API can be used to create secret scope.*

**CORRECT!** The **Databricks CLI** and **Databricks REST API** are the **only** two methods to create a **secret scope** in Databricks.

More Info: [Secret scope creation in Databricks](https://docs.databricks.com/en/security/secrets/secret-scopes.html#overview:~:text=You%20create%20a%20Databricks%2Dbacked%20secret%20scope%20using%20the%20Databricks%20CLI%20tutorial%20(version%200.7.1%20and%20above).%20Alternatively%2C%20you%20can%20use%20the%20Secrets%20API.)

## Question 46:

Which of the following columns from a Delta table cannot be classified as a PII column?

* **passport\_number**
* **credit\_card\_number**
* **biometrics**
* **gender**
* **name**

**Explanation**

**PII**(*Personally Identifiable Information*) is any information that is related to an **identifiable** person. The sensitive **PII** **data** should always have an **additional** layer of **security** when saving the data. Delta table’s **encryption** can be leveraged to secure the **PII** data.

While all the above options can be used to personally **identify** a person, **gender** column will **not** be considered **PII** data.

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

## Question 47:

Two data engineers are having different sets of permissions over a notebook. One of them is having **Can Manage** permission whereas the other one is having **Can Edit** permission. Which of the following statements defines the abilities of each of them?

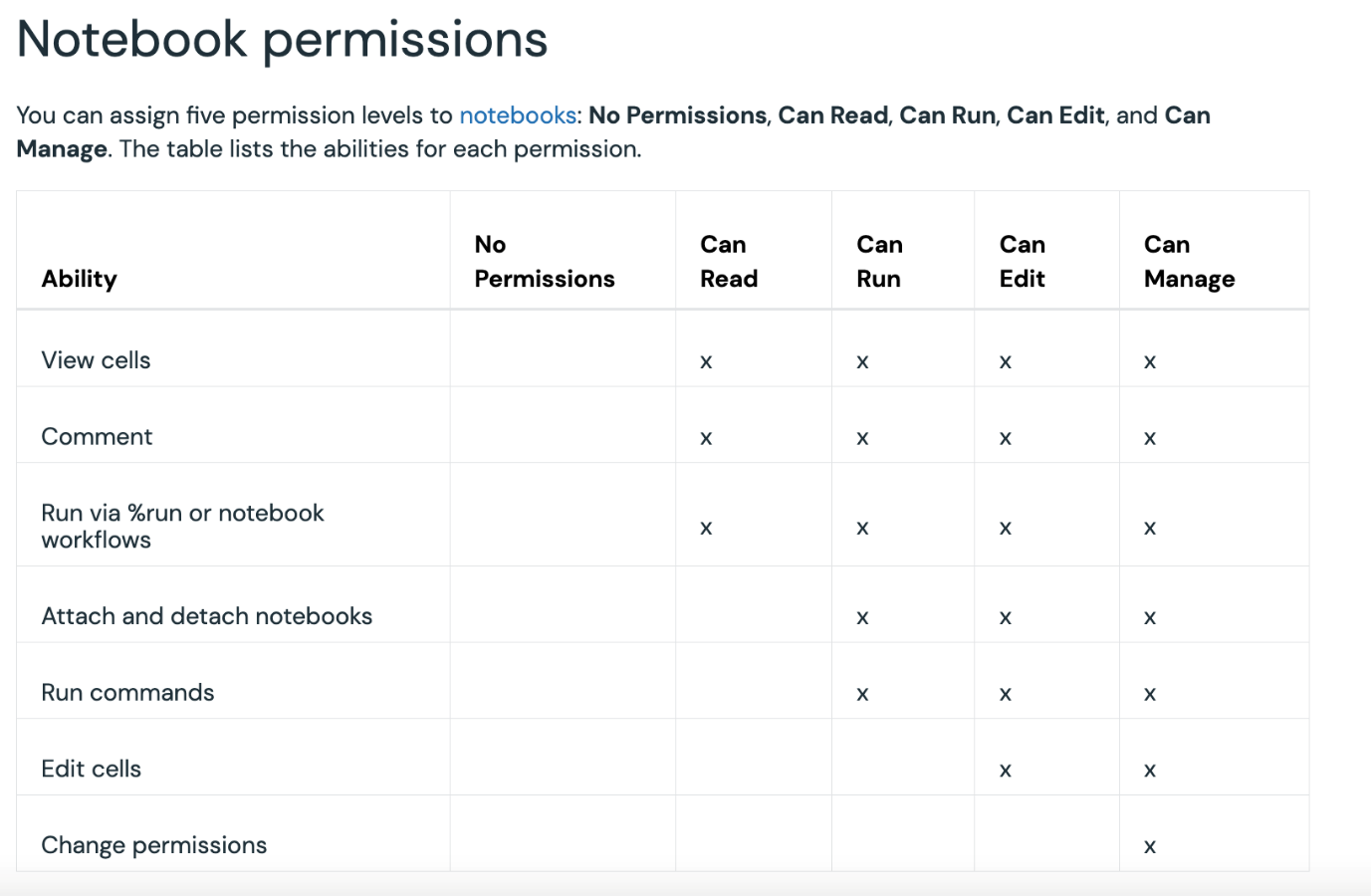
* **Only the data engineer with Can Manage permission can attach the notebook to a cluster.**
* **The data engineer with Can Edit permission can edit the notebook while the other data engineer cannot.**
* **Both the data engineers cannot detach the notebook from a terminated cluster.**
* **The data engineer with Can Manage permission can change the permissions on the notebook whereas the other data engineer cannot.**
* **Both data engineers share the same set of abilities.**

**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** **permission** on 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 permissions in Databricks](https://docs.databricks.com/en/security/auth-authz/access-control/workspace-acl.html#notebook-permissions)

## Question 48:

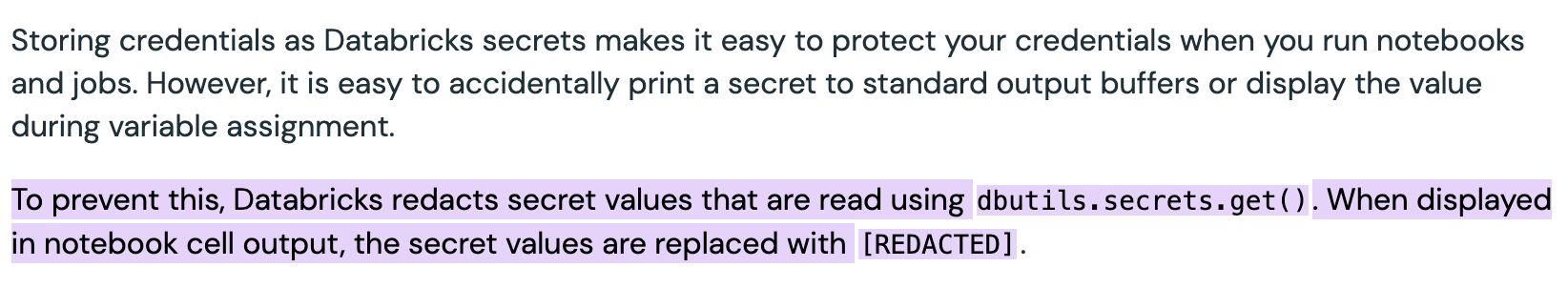
A data engineer creates a secret scope named **prod** and adds a secret with key as **authenticate** and value as **my\_unique\_authentication** What will be the output if a data engineer executes the following command in a Python notebook:

**dbutils.secrets.get('prod','authenticate')**

* **The command will fail as the secrets cannot be read inside a notebook using the get() method.**
* **The command will be successful and [REDACTED] will be printed as the output.**
* **my\_unique\_authentication will be printed as the output.**
* **As soon as the command is executed, a file will be downloaded with the password.**
* **The command will be executed successfully but no output will be printed.**

**Explanation**

A secret can be read using **dbutils.secrets.get()** method but once the command is **executed**in a notebook, it will print **[REDACTED]** This is done to ensure that the secret value is **not** accidentally **printed** in a notebook cell.

More Info: [Secret Redaction in Databricks](https://docs.databricks.com/security/secrets/redaction.html#:~:text=To%20prevent%20this%2C%20Databricks%20redacts%20secret%20values%20that%20are%20read%20using%20dbutils.secrets.get().%20When%20displayed%20in%20notebook%20cell%20output%2C%20the%20secret%20values%20are%20replaced%20with%20%5BREDACTED%5D)

## Question 49:

Which of the following details are available in the Environment Tab of Spark UI?

* **Properties like Runtime information, Spark properties and Hadoop properties.**
* **Name of the environment - DEV, UAT or PROD.**
* **The Cluster Metrics including network utilization, CPU usage and cluster memory usage.**
* **There is no Environment Tab in Databricks Spark UI.**
* **List of environment-specific tasks like env\_task\_1, env\_task\_2 etc.**

**Explanation**

The **Environment Tab** in the **Spark UI** contains properties of the cluster including **Spark**, **Hadoop** and **System** properties.

More Info: [Environment Tab in Spark UI](https://spark.apache.org/docs/latest/web-ui.html#environment-tab)

## Question 50:

A data engineer has set up a Databricks SQL alert on the **speed** column which returns multiple values. Which of the following values would be selected to trigger the alert?

* **The average of all the values.**
* **The row number selected while setting up the alert.**
* **The first value**
* **The last value**
* **If any of the values satisfy the condition, the alert will be triggered.**

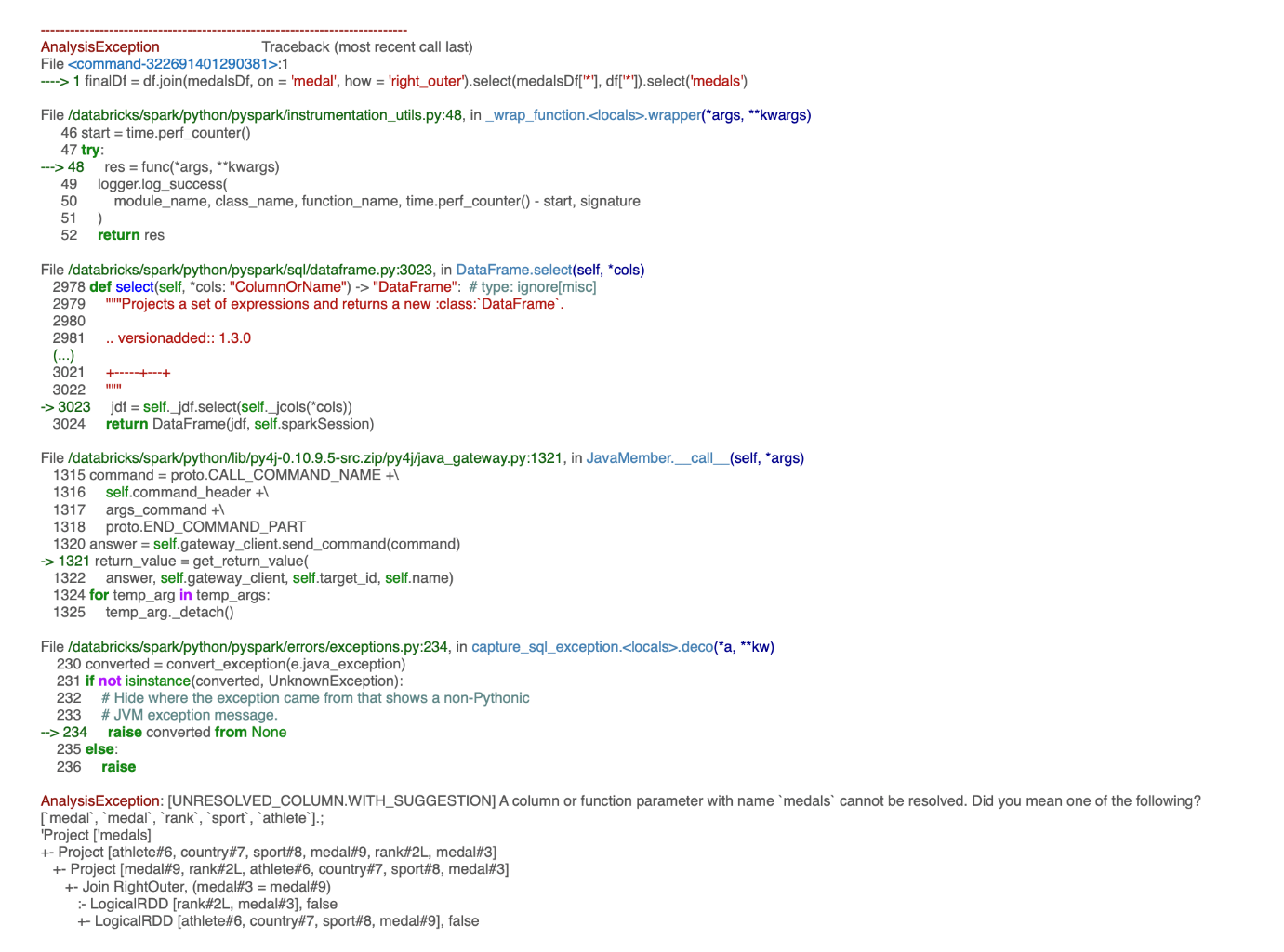
**Explanation**

If your query returns **multiple** records, the condition is checked on the **first one.**

More Info: [Creating an alert in Databricks](https://docs.databricks.com/sql/user/alerts/index.html#:~:text=If%20a%20target%20query%20returns%20multiple%20records%2C%20Databricks%20SQL%20alerts%20act%20on%20the%20first%20one)

## Question 51:

You are given the following error logs:

****

What can be the reason behind this error?

* **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.**
* **medals column exists in both the DataFrames**
* **medals column does not exist in any of the DataFrames being joined**

**Explanation**

Error debugging is an important task for a professional data engineer working on Databricks. In the above error log, **AnalysisException** sums up the whole error in a **developer-friendly** language.

Here, the definition for **AnalysisException** states that the column named **medals** does **not** exist, which is enough to answer these types of questions.

[Static Notebook](https://certification-champs.github.io/professional-data-engineer-notebooks/Practice%20Test%203/Question%2051.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: ['Column not found' error in Databricks](https://docs.databricks.com/en/error-messages/index.html#except_unresolved_column_in_struct_expansion:~:text=The%20column/field%20name%20%3CobjectName%3E%20in%20the%20EXCEPT%20clause%20cannot%20be%20resolved.%20Did%20you%20mean%20one%20of%20the%20following%3A%20%5B%3CobjectList%3E%5D%3F)

## Question 52:

Which of the following fields in Cluster UI can be searched to check the utilization of the cluster CPU?

* **Ganglia UI under the Metrics Tab**
* **Event Log Tab**
* **Storage under the Spark UI Tab**
* **CPU table under the Metrics Tab**
* **CPU table under the Driver logs Tab**

**Explanation**

Ganglia UI is a one-stop destination for all the cluster-related information including **hourly load**, **cluster memory used**, **cluster CPU** **usage** and **cluster Network.**

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 Ganglis Metrics in Databricks](https://docs.databricks.com/clusters/clusters-manage.html#ganglia-metrics:~:text=your%20Datadog%20account.-,Ganglia%20metrics,-To%20access%20the)

## Question 53:

A query is created for sending notifications to the team when the number of pending orders is more than 100 for a manufacturing startup. The query is set to refresh every hour. For the past 10 hours, the **pending\_orders** column returned the following values:

120

115

109

98

67

101

121

98

79

131

How many times the notification would be sent to the team if **Just once** is selected in the **When triggered, send notification** field?

* **3 times**
* **1 time**
* **5 times**
* **No notification will be sent**
* **6 times**

**Explanation**

In the above question, **Just once** is selected in the **When triggered, send notification** field. Let us look at the status after every hour:

**120** -> Changed to **TRIGGERED** (Notification is sent for the **first** time)

**115** -> Remains in the **TRIGGERED** state

**109** -> Remains in the **TRIGGERED** state

**98** -> Changed to **OK** (Notification is sent for the **second** time)

**67** -> Remains in the **OK** state

**101** -> Changed to **TRIGGERED** (Notification is sent for the **third** time)

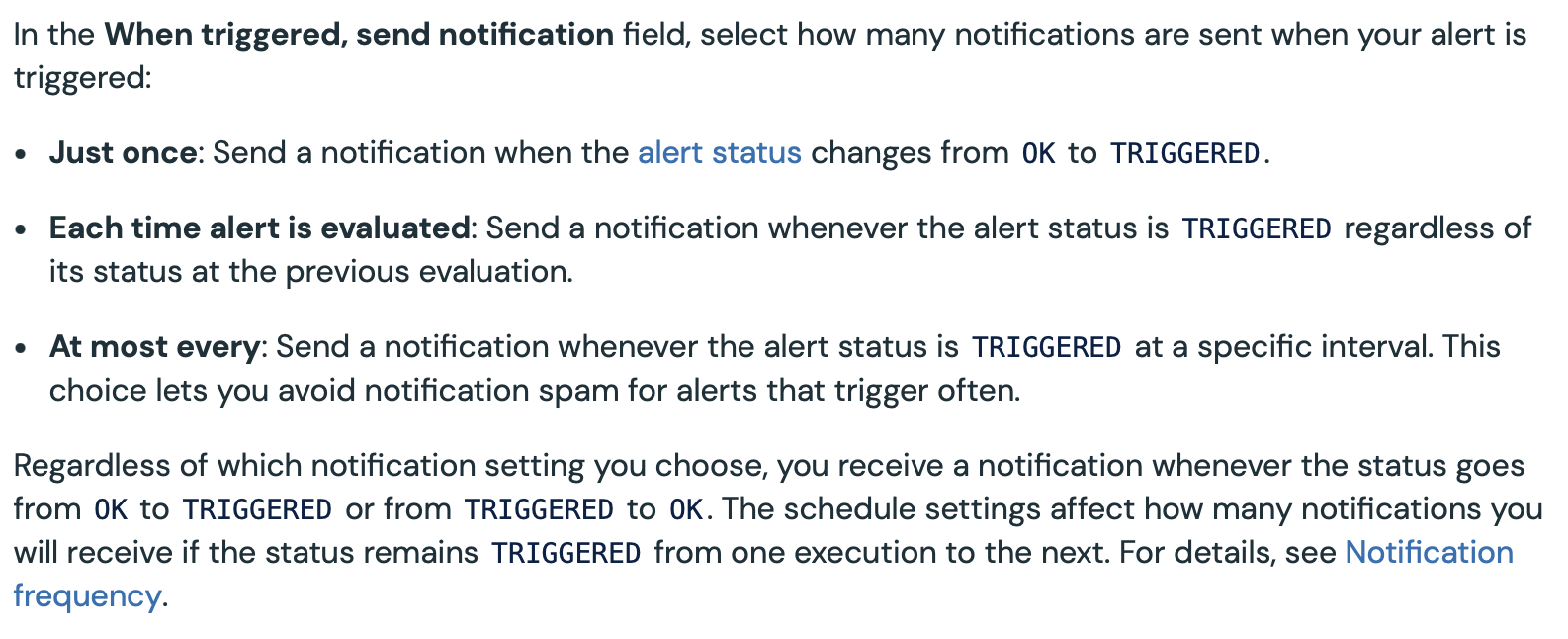
**121** -> Remains in the **TRIGGERED** state

**98** -> Changed to **OK** (Notification is sent for the **fourth** time)

**79** -> Remains in the **OK** state

**131** -> Changed to **TRIGGERED** (Notification is sent for the **fifth** time)

This analysis makes it clear that while using **Just once**, the notification is sent to the team whenever the status is changed from **OK** to **TRIGGERED** or **TRIGGERED** to **OK**. Moreover, if the status remains in the **TRIGGERED** or **OK** state for more than one hour, the notification will be sent only for the **first** hour. For instance, in the first **3** hours, the status remains **TRIGGERED** but the notification is sent only for the **first** hour. Similarly, the status remains **OK** for the next **2** hours but the notification will be sent for the **first** time the state is changed i.e. in the **fourth** hour.

More Info: [Notifications frequency of an Alert in Databricks SQL](https://docs.databricks.com/sql/user/alerts/index.html#:~:text=If%20the%20notification,on%20those%20days)

## Question 54:

Which of the following will you not see on the Spark UI Home page?

* + **Jobs Tab**
  + **Stages Tab**
  + **Tasks Tab**
  + **Environment Tab**
  + **JDBC/ODBC Server Tab**

**Explanation**

The following is the screenshot from the home page of the**Spark UI:**



Clearly, there is **no** **Tasks tab** available in **Spark UI.**

More Info: [Spark UI Tabs](https://spark.apache.org/docs/latest/web-ui.html#:~:text=Table%20of%20Contents,ODBC%20Server%20Tab)

## Question 55:

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

* **Rtest**
* **testthem**
* **testthis**
* **testR**
* **testthat**

**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 test case in **R**, **testthat** library should be installed.

More Info: [Unit test cases in R](https://docs.databricks.com/notebooks/testing.html#:~:text=pytest%20for%20Python%2C-,testthat%20for%20R,-%2C%20and%20ScalaTest%20for)

## Question 56:

A data engineer creates a task named **filter\_countries** with no dependency. After some days, the data engineer edits the job to add another task named **get\_capitals** which is dependent on the **filter\_countries** task. After a week, the data engineer creates another task named **get\_currencies** which is also dependent on the **filter\_countries** task. Which of the following job flows is correct for the above scenario?

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

| get\_capitals | |get\_currencies|

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

\ /

\ /

\ /

\ /

\ /

\ /

\ /

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

|filter\_countries|

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

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

|filter\_countries|

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

|

|

|

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

| get\_capitals |

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

|

|

|

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

| get\_currencies |

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



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

|filter\_countries|

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

/ \

/ \

/ \

/ \

/ \

/ \

/ \

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

| get\_capitals | |get\_currencies|

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

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

|filter\_countries| | get\_capitals | | get\_currencies |

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

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

|filter\_countries|

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

|

|

|

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

| get\_currencies |

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

|

|

|

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

| get\_capitals |

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

**Explanation**

As both tasks are **dependent** on a **single task,** it will create a **hierarchical workflow.**

## Question 57:

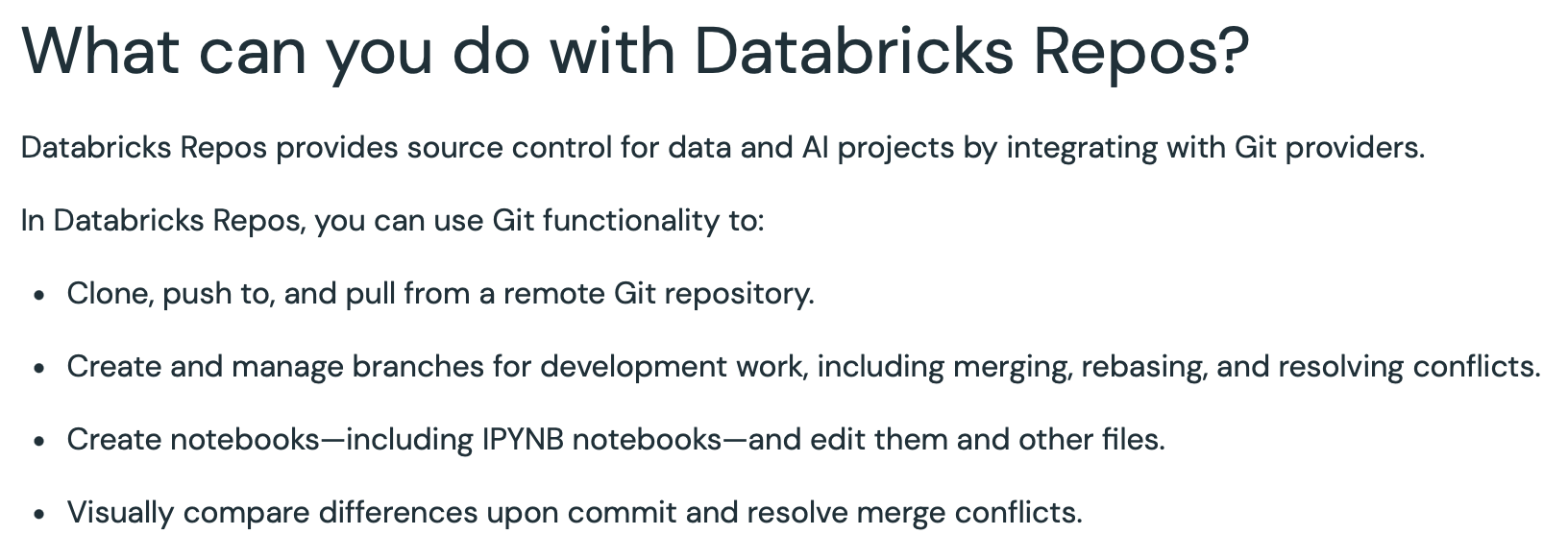
A data engineering team is using Databricks Repos integrated with GitHub for version control. The team got to know that there are some tasks that cannot be performed through Databricks Repos and can only be done using GitHub. Which of the following version control tasks cannot be performed in Databricks Repos?

* + **Create a branch**
  + **Create a notebook**
  + **Clone a Git repository**
  + **Resolve Merge conflicts**
  + **All of the above tasks can be performed using Databricks Repos**

**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 **GitHub** repository, **creating branches** and **notebooks** etc.

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

More Info: [Supported Git operations in Databricks Repos](https://docs.databricks.com/en/repos/index.html#what-can-you-do-with-databricks-repos)

## Question 58:

Which of the following is not true about Unit testing and Integration testing?

* **Integration testing is performed after Unit testing.**
* **The internal functionality of the module should be known while performing Unit testing.**
* **Unit testing is usually performed by the developer.**
* **Integration testing is a type of white-box testing.**
* **Unit testing can be performed simultaneously for different modules.**

**Explanation**

Let us look at all the options one by one:

*Integration testing is performed after Unit testing.*

The **integration testing**(as the name suggests) starts **after** the testing for each module has been finished. Once all the modules are tested **independently**, the interaction between them is tested during integration testing. Hence, the integration testing starts **after** the completion of unit testing.

*The internal functionality of the module should be known while performing Unit testing.*

As **unit testing** involves writing **positive** and **negative** test cases, the functionality of the modules to be tested should be **known** to the unit tester.

*Unit testing is usually performed by the developer.*

The **developer** knows the module **inside out**and thus is **responsible** for performing **unit testing**(usually using **assert** methods).

*Integration testing is a type of white-box testing.*

This is an **incorrect** statement. Integration testing is a type of **black box**testing whereas **unit testing** is a type of **white box** testing.

*Unit testing can be performed simultaneously for different modules.*

As unit testing is performed **independently** for all the modules, it can be done **simultaneously.**

## Question 59:

Which of the following statements is true about the default retention period of the job runs?

* **The details of a failed job run can be accessed for 45 days whereas the details of a successful job run can only be accessed for 30 days.**
* **The details of a failed as well as a successful job run can only be accessed for 60 days.**
* **The details of a failed job run can be accessed for 60 days whereas the details of a successful job run can only be accessed for 30 days.**
* **The details of a failed job run can only be accessed for 30 days whereas the details of a successful job run can be accessed for 60 days.**
* **The details of a failed as well as a successful job run can only be accessed for 45 days.**

**Explanation**

Databricks stores the job run details for only **60 days** irrespective of the **status** of the job run. To retain the logs for **more** than 60 days, Databricks suggests **exporting** the job run **logs** to a storage location like **DBFS** and **S3.**

More Info: [Databricks logs retention](https://docs.databricks.com/workflows/jobs/jobs.html#:~:text=Databricks%20maintains%20a%20history%20of%20your%20job%20runs%20for%20up%20to%2060%20days.%20If%20you%20need%20to%20preserve%20job%20runs%2C%20Databricks%20recommends%20that%20you%20export%20results%20before%20they%20expire)

## Question 60:

The Databricks admin installed some Python libraries as part of cluster creation. Now, a member of the team runs the following statement in one of their notebooks to remove the **pandas** library from their notebook:

**%pip uninstall pandas**

Which of the statements describes the outcome?

* **The library will be uninstalled from the notebook but will remain available to all other notebooks.**
* **The library will not be uninstalled as the libraries installed on clusters cannot be uninstalled through %pip command.**
* **The library will be uninstalled from the cluster and will not be available to any other notebook attached to the cluster.**
* **The library will not be uninstalled as -y should be used to uninstall the libraries installed on a cluster.**
* **The library will not be uninstalled but it will not be available for further use in the notebook.**

**Explanation**

***A library installed on the cluster cannot be uninstalled through notebooks.***

To **uninstall** a library installed on the **cluster**, you can go to **Compute** menu and **select** the appropriate cluster.

Also note, if you need to uninstall a library installed on your notebook you can use:

**%pip uninstall -y <library\_name>**

To print the list of libraries already installed, you can run the following set of commands:

**import sys**

**print(sys.path)**

Bottom of Form