Question 1

A data engineer has joined a team which is using Databricks notebooks with Git integration. The data engineer needs to clone the Git repository in Databricks to start collaborating with the team. Which of the following tabs from the left menu bar should the data engineer select to clone the remote repository?

- ● **Repos**
- ○ **Jobs**
- ○ **Data**
- ○ **Git**
- ○ **VCS**

**Explanation**
If you need to clone a Git repository, you need to select **Repos** tab from the left side menu. You, then, need to click on the **Add Repo** option and clone the required Git repository using the **URL** and the **Git provider** of the repository.

Some of the most commonly used Git providers supported by Databricks are:

- BitBucket
- GitHub
- GitLab
- AWS CodeCommit

More Info: Clone a remote Git repository

Question 2

Which of the following is TRUE about a Job cluster in Databricks?

- [✓ **A Job Cluster can be created using the UI, CLI or REST API**]
- **Multiple users can share a Job cluster**
- **Job clusters can be restarted as per need**
- **The Job cluster terminates when the Job ends**
- **Job cluster works only with Python Language notebooks**

**Explanation**
Let us look at the options and see which option is correct for a *job cluster*.

*A Job Cluster can be created using the UI, CLI or REST API*

**INCORRECT**. A *job cluster* is created by the job scheduler when you run a job. It cannot be created using the UI, CLI or REST API.

*Multiple users can share a Job cluster*

**INCORRECT**. A *job cluster* cannot be shared between users as it is automatically created and terminated.

*Job clusters can be restarted as per need*

**INCORRECT**. An *all-purpose cluster* can be restarted as per need, but a *job cluster* is for one-time use only.

*The Job cluster terminates when the Job ends*

**CORRECT**. As it is a one-time use cluster, it is terminated as the job ends.

*Job cluster works only with Python language notebooks*

**INCORRECT**. A *job cluster* is not confined to Python language notebooks, it can be used for other languages notebooks too.

As you would know, there are two types of clusters in Databricks - a **job cluster** and an **all-purpose cluster**.

Additionally, let's quickly check which of the above options are true for an *all-purpose cluster*.

1. An *all-purpose cluster* can be created using the UI, CLI or REST API - **TRUE**

2. Multiple users can share an *all-purpose* cluster - **TRUE**

3. *All-purpose clusters* can be restarted as per need - **TRUE**

4. An *all-purpose cluster* terminates when the job ends - **FALSE**

5. *All-purpose clusters* works only with Python language notebooks - **FALSE**

Also note, the clusters *(any type)* can be used with any language supported by Databricks notebooks. **A cluster is not bound to a language.** Same cluster can run Python code and can also be used for executing SQL statements. The clusters are not language dependent.

More Info: Clusters in Databricks

Question 3

Which of the following is NOT one of the magic commands that can be used in a Databricks notebook?

- [✓ `%sql`]
- `%java`
- `%python`
- `%r`
- `%scala`

**Explanation**

**Magic commands** are an integral part of Databricks notebooks. Let's take a case where you need to use a Scala code inside a notebook that has its default language set to Python. In this case, you can easily use `%scala` as the first line of the cell which turns the cell into a Scala cell while enjoying Python in the rest of the cells of the notebook.

There are 4 language specific magic commands supported in Databricks notebook i.e. `%sql`, `%python`, `%r` and `%scala` while `%java` is not a valid magic command.

More Info: Language Magic Commands

Question 4

As a data engineer, you have seen that a large number of files for `employees` table is taking a lot of memory. These files are nothing but the versioned history of the `employees` table. You need to remove the files from the system and keep only the files that are maximum 2 days old. Your colleague has written the following query.

`VACUUM employees RETAIN 2 DAYS`

What should be corrected to run this query?

- ○

  **`RESTORE` command should be used instead of `VACUUM`**

- ○

  **`RETAIN FOR 2 DAYS` should be used**

- ○

  **You cannot delete the old files for just one table, you need to add database name as `VACUUM` is a database level operation**

- [✓ ●]

  **`VACUUM` accepts value in `HOURS` and not `DAYS`, `2 DAYS` should be replaced with `48 HOURS`**

- ○

  **`DRY RUN` should be used at the end of the SQL statement**

**Explanation**
Correct code block:

`VACUUM employees RETAIN 48 HOURS`

The reason behind this answer is that the `VACUUM` command accepts values in **hours** and **not days**.

More Info: [Using VACUUM to remove unused files]()

Question 5

You are working as a data engineer in XYZ company. A delta table `department` already exists but now you need to change the schema. Another data engineer has written a SQL statement to create the table with the latest Databricks Runtime, but it is not working as desired. What can be the error in the following SQL statement?

```
CREATE OR REPLACE department
(roll_no int, name string);
```

- ○

  **`USING DELTA` should be added at the end of SQL statement**

- [✓ ●]

  **`TABLE` keyword is missing before `department`**

- ○

  **The brackets around schema should be removed**

- ○

  **`CREATE TABLE IF NOT EXISTS` should be used instead of `CREATE OR REPLACE`**

- ○

  **The `department` table should be dropped first using `DROP` command as you cannot overwrite a Delta table**

**Explanation**

Let's look at all the options one-by-one.

`USING DELTA` *should be added at the end of SQL statement*

**INCORRECT**. As all the tables are created using DELTA by default, adding `USING DELTA` at the end of the SQL statement is redundant.

`TABLE` *keyword is missing before* `department`

**CORRECT**. To create a table in SQL the following syntax is used. `CREATE TABLE table_name(schema);`

*The brackets around schema should be removed*

**INCORRECT**. The brackets around the schema are correctly placed.

`CREATE TABLE IF NOT EXISTS` *should be used instead of* `CREATE OR REPLACE`

**INCORRECT**. `CREATE TABLE IF NOT EXISTS` is used when you need to create a table ONLY if it does not exist whereas in this case you need to re-create the table which already exists.

*The* `department` *table should be dropped first using* `DROP` *command as you cannot overwrite a Delta table*

**INCORRECT**. A Delta table can be overwritten in the same way a RDBMS table is overwritten using SQL command

More Info: CREATE TABLE command

Question 6

Which of the following is NOT true about a Data Lake?

- ○ **Data Lake can store structured, unstructured, and semi-structured data**
- ○ **Data in a Data Lake is stored in its original format**
- [✓ **Defining schema on load is necessary in a Data Lake**]
- ○ **Data Lake can be used to store real time data**
- ○ **Databricks Lakehouse combines the features of Data Lake and Data Warehouse**

**Explanation**

All the options except the third one are TRUE. Let’s put Data Lake in simpler terms. You can think of **Data Lake** as a folder in your computer in which you have stored some parquet files, a directory filled with JSON data, data coming from a Kafka Stream and RDBMS backup etc. There is **no schema change** as the data enters the Data Lake as the data enters the data lake in its original format. **Thus, defining schema on load is not required.**

More Info: Data Lakes | Databricks Lakehouse

Question 7

Your fellow data engineer is using a Databricks notebook which is defaulted to Python language. They need to have an interactive view of the data on which they can plot a graph. They try to run the following query on an aggregated Gold table `avg_scores`, but they are not able to see the output data.

```
spark.sql("SELECT * FROM avg_scores")
```

What is the reason that they are not able to view the data?

- ○ **Databricks does not support querying SQL table in a Python cell**
- ○ **As it is a Python cell, `show()` operation should be applied to get the contents**
- ○ **The cell's language should be changed to SQL and the cell should be executed again**
- [✓ ● **The `spark.sql()` function should be passed as an argument to the `display()` function to view the data**]
- ○ **The argument passed to the `spark.sql()` function should not be enclosed in quotes**

**Explanation**
This is an interesting

Question. Let us look at all the options one by one.

*Databricks does not support querying SQL table in a Python cell*

**INCORRECT**. Databricks supports this through a special Spark SQL function - `spark.sql()`

*As it is a Python cell,* `show()` *operation should be applied to get the contents*

**INCORRECT**. `show()` operation can be used to view the data but it cannot be used to view the interactive data on which plotting can be performed in Databricks.

*The cell's language should be changed to SQL and the cell should be executed again*

**INCORRECT**. By changing the cell's language to SQL, you cannot run `spark.sql()` as it is a Python specific function.

*The* `spark.sql()` *function should be passed as an argument to the* `display()` *function to view the data*

**CORRECT**. In order to view the desired result, you need to use `display()` function provided by Databricks i.e. `display(spark.sql("SELECT * FROM avg_scores"))`

*The argument passed to the* `spark.sql()` *function should not be enclosed in quotes*

**INCORRECT**. The SQL query passed to the `spark.sql()` function should always be a string.

More Info: Using display() function for visualization

Question 8

Which of the following is false about an external table?

- ○

  **An external table is also called as unmanaged table**

- ○

  **When an external table is dropped, only the metadata is deleted from the system and the data remains intact**

- [✓ ●]

  **External table never specify `LOCATION` while creating the table**

- ○

  **Registering an external table to a different database is easy as no data movement is required**

- ○

  **Databricks manages only metadata for an external table**

**Explanation**

All the options except the third one (Option C) are **TRUE**. When creating an unmanaged or external table specifying the **location** of the data is a **must**.

More Info: Unmanaged or External table in Databricks

Question 9

The following statement intends to select all the records from version 6 of table `testing_logs`. Which of the following should replace the blank to achieve the task?

`SELECT * FROM testing_logs _________`

- ○ `VERSION 6`
- ○ `ROLLBACK TO 6`
- [✓ ● `VERSION AS OF 6`]
- ○ `ROLLBACK AS OF 6`
- ○ `‘VERSION = 6’`

**Explanation**

`VERSION` command provides you with the ability to query **previous versions** of your table.

Correct code block: `SELECT * FROM testing_logs VERSION AS OF 6`

Also note, `SELECT * FROM testing_logs@v6` can also be used.

More Info: [Select data from previous version of a table](#) | [Using @ syntax](#)

Question 10

Which of the following commands fails to return the metadata of `flights` table?

- ○ `DESC EXTENDED flights`
- ○ `DESCRIBE DETAIL flights`
- ○ `DESC flights`
- ○ `DESCRIBE HISTORY flights`
- [✓ `DESC PRIVACY flights`]

**Explanation**
The `DESCRIBE` or `DESC` command is used to get the **basic structure** of data objects like **tables**, **databases** etc. Some more keywords like `EXTENDED`, `DETAIL`, `HISTORY` can be used in conjunction with the `DESCRIBE` or `DESC` command but `PRIVACY` is not one of them.

More Info: DESCRIBE COMMAND | DESCRIBE HISTORY

Question 11

Which of the following magic commands can be used to run a notebook from another notebook?

- ○ `%run_notebook`
- [✓ `%run`]
- ○ `%runNotebook`
- ○ `%start`
- ○ `%start_notebook`

**Explanation**

`%run` is a special magic command that can be used to run a notebook from another notebook. Rest of the options are not valid magic commands in Databricks notebook.

More Info: %run command in Databricks

Question 12

A junior data engineer from your team wants to insert 5 records in the `employees` table. They have come up with the following set of SQL queries.

```
INSERT INTO employees VALUES (234, ‘Erich Heard’);
INSERT INTO employees VALUES (209, ‘Paul Fosbury’);
INSERT INTO employees VALUES (141, ‘Ricky Matt’);
INSERT INTO employees VALUES (940, ‘Jeff Sims’);
INSERT INTO employees VALUES (744, ‘Chriss Holmes’);
```

Each of the statements is processed as a separate transaction and you need to modify the statement to be able to insert all 5 records in one go. Which of the following SQL statements can be used to insert these 5 records in a single transaction?

- 
  ```
  INSERT INTO TABLE employees VALUES
  (234, ‘Erich Heard’),
  (209, ‘Paul Fosbury’),
  (141, ‘Ricky Matt’),
  (940, ‘Jeff Sims’),
  (744, ‘Chriss Holmes’);
  ```
- 
  ```
  INSERT INTO employees MULTIPLE VALUES
  (234, ‘Erich Heard’),
  (209, ‘Paul Fosbury’),
  (141, ‘Ricky Matt’),
  (940, ‘Jeff Sims’),
  (744, ‘Chriss Holmes’);
  ```
- 
  ```
  INSERT INTO employees ‘5’ VALUES
  (234, ‘Erich Heard’),
  (209, ‘Paul Fosbury’),
  (141, ‘Ricky Matt’),
  (940, ‘Jeff Sims’),
  (744, ‘Chriss Holmes’);
  ```
- [✓]
  ```
  INSERT INTO employees VALUES
  (234, ‘Erich Heard’),
  (209, ‘Paul Fosbury’),
  (141, ‘Ricky Matt’),
  (940, ‘Jeff Sims’),
  (744, ‘Chriss Holmes’);
  ```
- 
  ```
  INSERT INTO employees
  (234, ‘Erich Heard’),
  (209, ‘Paul Fosbury’),
  (141, ‘Ricky Matt’),
  (940, ‘Jeff Sims’),
  (744, ‘Chriss Holmes’);
  ```

**Explanation**

The correct code block is:

```
INSERT INTO employees VALUES
```

```
(234, ‘Erich Heard’),
(209, ‘Paul Fosbury’),
(141, ‘Ricky Matt’),
(940, ‘Jeff Sims’),
(744, ‘Chriss Holmes’);
```

Rest of the options are not syntactically correct. The `INSERT` query works the same way for adding multiple records as it works for adding a single record. You can provide comma separated rows to add them in one go.

More Info: Multi-row insert into a table

Question 13

You have created a new managed table `members` in the `company` database using the following set of SQL statements:

```
1. CREATE DATABASE IF NOT EXISTS company;
2. USE company;
3. CREATE OR REPLACE TABLE members(id int, name string);
```

What will be the location of the newly created table?

- ○ `dbfs:/user/warehouse/company.db/`
- ○ `dbfs:/hive/warehouse/company/`
- ○ `dbfs:/user/hive/company/`
- [✓ `dbfs:/user/hive/warehouse/company.db/`]
- ○ `dbfs:/user/lakehouse/company.db/`

**Explanation**

This is a trick

Question. First of all, you should know that the database has been created without any `LOCATION` parameter which means that the database will be created at the **default location**. The default location for a database is always `dbfs:/user/hive/warehouse/`

Now, as the name of the database is `company`, a directory named `company.db` will be created in the above location and thus the data for the `members` table will be stored in `company.db` directory. So, correct path for the `members` data storage is `dbfs:/user/hive/warehouse/company.db/`

**`SCHEMA` and `DATABASE` can be used interchangeably i.e. `CREATE SCHEMA` is equivalent to `CREATE DATABASE`**

If you do not specify the location while creating the database, the database will always be created at the default location. A **trick to remember** the default location of the database is **UHW** i.e. **U**ser **H**ive **W**arehouse.

More Info: Default location for CREATE DATABASE command

Question 14

Which of the following statements defines a Python function `get_column()` which accepts a column name and prints the values in that column from `payments` table?

-
  ```
  def get_column(column_name):
      spark.sql(f“SELECT {column_name} from payments”)
  ```
- [✓]
  ```
  define get_column(column_name):
      display(spark.sql(f“SELECT {column_name} from payments”))
  ```

-
  ```
  function get_column(column_name):
      spark.sql(“SELECT {column_name} from payments”).show()
  ```

-
  ```
  def get_column(column_name):
      spark.sql(f“SELECT {column_name} from payments”).show()
  ```

-
  ```
  def get_column(column_name):
      spark.sql(“SELECT column_name from payments”)
  ```

**Explanation**

To answer this

Question you need to know the syntax of a Python function. The simplest Python function uses the following syntax:

```
def function_name(parameters):
    function_body
```

You can also provide parameters' data types and return type in the function definition but are completely optional.

```
def function_name(parameter1: data_type, parameter2: data_type) -> return_type:
    function_body
```

Please note, the Python function follows indentation and no brackets to define the function body.

Also, this

Question checks your knowledge on **formatted strings in Python**. Using formatted strings or f-strings, you can make your SQL statement more readable by using Python variables in between the SQL statements using `{}`

The formatted strings are mainly used to print the formatted data that makes your code easily traceable and maintainable.

Correct code block:

```
def get_column(column_name):
    spark.sql(f“SELECT {column_name} from payments”).show()
```

If you are new to Python language, do not forget that a function is defined in Python using `def` keyword. It will be helpful for you to remember this for the actual exam.

More Info: Formatted strings in Python

Question 15

You, as a data engineer, want to use a SQL query in a Python function. What approach can you follow?

- ●

  **`spark.sql()` function should be used to run the SQL query**

- ○

  **Change the cell's language to SQL as the SQL cell allows the usage of Python code as well**

- ○

  **SQL query cannot be accessed inside Python code**

- ○

  **`pyspark.sql()` function should be used to run the SQL query**

- ○

  **Install Spark SQL driver to run the query**

**Explanation**

If you are using a Databricks notebook, `spark.sql()` is an essential part of your code as it can be used to run SQL queries inside a Python cell. `sql()` is a function on a `SparkSession` object. In Databricks, `spark` is an object of `SparkSession` and that is the reason you can use `spark.sql(query)` to run a SQL query on a view or a table.

More Info: Using spark.sql() to run SQL queries in Python

Question 16

A data analyst has created an empty delta table named `passengers`. The data analyst needs to make sure that the `age` of the new passengers should be less than 60. Which of the following statements will ensure that all the incoming records have `age` column's value less than 60?

- ○ `ALTER TABLE passengers ADD check_age CHECK (age < 60)`
- ○ `ALTER TABLE passengers ADD CONSTRAINT check_age (age < 60)`
- ○ `ALTER passengers ADD CONSTRAINT check_age CHECK age < 60`
- [✓ `ALTER TABLE passengers ADD CONSTRAINT check_age CHECK (age < 60)`]
- ○ `ALTER passengers ADD CONSTRAINT check_age CHECK (age < 60)`

**Explanation**
Once a table has been created, a need to add or drop a constraint can arrive. You can use the `ALTER` command to add or drop a constraint. According to the

Question, you need to add a check constraint to the table. The following is the syntax if you want to add a check constraint to the table.

```
ALTER TABLE table_name ADD CONSTRAINT constraint_name CHECK (condition)
```

The only option which correctly follows the syntax is option D. Hence, is correct.

More Info: Adding a Check constraint on a table

Question 17

A data engineer has created a Global Temp View `new_trains`. Which of the following SQL statements will show the contents of `new_trains` view?

- ○ `SELECT * FROM new_trains`
- ○ `SELECT * FROM global.new_trains`
- [✓ ● `SELECT * FROM global_temp.new_trains`]
- ○ `SELECT * FROM temp_global.new_trains`
- ○ `SELECT * FROM temp.new_trains`

**Explanation**

The global temporary views are registered in a temporary database named `global_temp`. So, to access the view you need to add `global_temp` before the name of the view i.e. `global_temp.new_trains`

More Info: Global Temporary View

Question 18

The following SQL statement intends to delete all the records from the `employees` table that contains string `‘Ex’` in `employee_code` column. Find the error in the statement.

```
DELETE VALUES from employees WHERE employee_code like ‘Ex%’
```

- ○ **Wildcards like `%` cannot be used in `WHERE` clause**
- ○ **`IF` should be used instead of `WHERE`**
- ○ **`TABLE` keyword should be used after table name i.e. `employees`**
- ● **`VALUES` should not be used in `DELETE` statements**
- ○ **`ALL VALUES` should be used instead of `VALUES` in `DELETE` statements**

**Explanation**
Correct SQL statement:

```
DELETE from employees WHERE employee_code like ‘Ex%’
```

Also note, the `VALUES` keyword is not used while deleting data from a table.

More Info: [Using DELETE for deleting data from a table]

Question 19

Which of the following Spark SQL functions can be used to convert an array column in a Delta table into multiple rows, with each row containing individual elements of the array?

- ○

  `SELECT`
- ○

  `FILTER`
- ○

  `TRANSFORM`
- ○

  `EXPLODE`

- ◉

  `EXPLODE_ARRAY`

**Explanation**

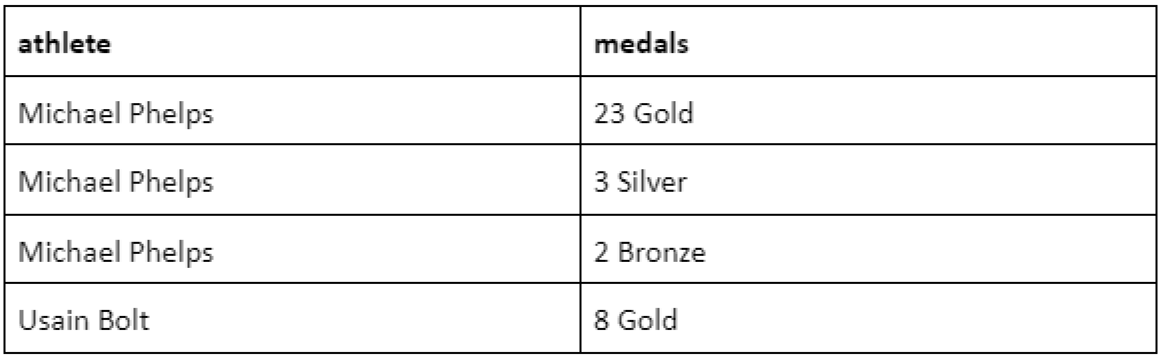
`EXPLODE` is an important function to deal with **complex data types** like arrays. This can explode the array to distribute each element into a new row.

Let us look at some sample data from the `olympics` table to understand more about `EXPLODE` function.

| athlete | medals |
|---|---|
| Michael Phelps | ["23 Gold", "3 Silver", "2 Bronze"] |
| Usain Bolt | ["8 Gold”] |

By running the following query on the `olympics` table, you can view the result of applying the `EXPLODE` function over the `medals` column.

`SELECT athlete, EXPLODE(medals) FROM olympics`



| athlete | medals |
|---|---|
| Michael Phelps | 23 Gold |
| Michael Phelps | 3 Silver |
| Michael Phelps | 2 Bronze |
| Usain Bolt | 8 Gold |

You can notice how `EXPLODE` function enables you to get each element of an array in its individual row.

More Info: EXPLODE function in Spark SQL

Question 20

A data analyst needs to replace the contents of table `tax_details` with the contents of table `new_tax_details` Both the tables have identical schema but table `new_tax_details` contain an extra column named `middle_name` The data analyst has written the following query to execute the overwrite:

```
INSERT OVERWRITE tax_details
SELECT * FROM new_tax_details;
```

What will be the outcome of the above query?

- ○

  **The data will be overwritten without any error**

- ○

  **The data will be overwritten with a schema mismatch warning message**

- [✓ ●]

  **The query will fail with a schema mismatch error**

- ○

  **The query will fail as `INSERT OVERWRITE` is not a valid command**

- ○

  **The query will be executed without any errors or warnings but the data will not be overwritten**

**Explanation**

`INSERT OVERWRITE` command helps to insert new data to a table while overwriting the old data but for the command to run smoothly, the schema of **source data** and **target table** should be **identical**. In this

Question, the schema of the source data and the target table is different which results in a **schema mismatch error**. Alternatively, to run the above query without any errors or warnings `option("overwriteSchema", "true")` can be added.

More Info: [Using INSERT OVERWRITE to overwrite a table in Spark SQL]

Question 21

Which of the following is not a valid operator which works on two or more tables?

- ○ `UNION`
- ○ `INTERSECT`
- ○ `MINUS`
- [✓ `PLUS`]
- ○ `EXCEPT`

**Explanation**

Let us see the usage of all the above operators one by one:

**UNION** – Used for combining rows from the tables with similar schemas. The resultant table is the collection of all the rows from the two tables. Also note, the duplicate rows are dropped by default. To preserve duplicate rows, `UNION ALL` should be used.

**INTERSECT** – Used for retrieving identical rows from the tables. The resultant table is the collection of rows which are present in both the tables. Also note, the duplicate rows are dropped, by default. To preserve duplicate rows, `INTERSECT ALL` should be used.

**MINUS** – Used for getting the rows which are present in the first table but not the second one.

**PLUS** – It is an invalid command

**EXCEPT** – Same as `MINUS` command.

More Info: Set Operators in SQL - EXCEPT | MINUS | UNION | INTERSECT

Question 22

Which of the following describes the advantage of using higher-order functions in Spark SQL?

- ○

  **The higher-order functions increase the number of clusters for running Spark SQL in Databricks**

- ○

  **The higher-order functions do not exist in Spark SQL**

- [✓ ●]

  **The higher-order functions help in directly working with complex data types**

- ○

  **The higher-order functions can be used to combine two tables using the `UNION` operator**

- ○

  **Higher-order functions can be used to speed up the `ORDER BY` query**

**Explanation**

The higher order functions like `FILTER`, `TRANSFORM`, `EXIST` etc. are used for directly working with the **complex data types like arrays**.

More Info: Higher-order functions in Spark SQL

Question 23

A data engineer is working on SQL UDF which adds two columns `salary` and `bonus` to view the total salary of all the employees. The data engineer has defined the following SQL UDF which intends to perform the required task but the last line has been deleted by mistake. What should come in the last line of the function definition?

```
CREATE FUNCTION total_salary(salary INT, bonus INT)
RETURNS INT
___________
```

- `CONCAT(salary, bonus)`
- `RETURN CONCAT(salary, bonus)`
- `salary + bonus`
- [✓ `RETURN salary + bonus`]
- `CONCAT(salary + bonus)`

**Explanation**

While defining a **UDF** you need to return something, for which `RETURN` keyword should be used. Without using the `RETURN` keyword, the function definition will return an **error**.

Now, two of the above options are using the `RETURN` keyword.

But, the `CONCAT` function used in option B, will not add the values but rather combine them using **string concatenation**. Suppose, for one of the employees, the `salary` is **500000** and the `bonus` is **60000**, concatenating the values will result in **50000060000**.

Surely, this is not required. So, you can discard this option.

Option D uses the `RETURN` keyword and adds the `salary` and the `bonus` using **+ operator** which sums the `salary` and `bonus` values and hence should be considered correct.

More Info: SQL UDF in Databricks

Question 24

Which of the following statements precisely describe the difference between `INSERT INTO` and `MERGE INTO` for Delta tables in Databricks?

- ○ **`INSERT INTO` can be used to insert and update to a table whereas `MERGE INTO` can be used for insert, update and delete.**
- ○ **`MERGE INTO` can be used to insert and update to a table whereas `INSERT INTO` can be used for insert and delete.**
- [✓ ◉] **`MERGE INTO` can be used to insert, update and delete from/to a table whereas `INSERT INTO` can be used only to insert values to a table.**
- ○ **`MERGE INTO` can be used to insert and delete from/to a table whereas `INSERT INTO` can be used to insert, update and delete**
- ○ **Both `MERGE INTO` and `INSERT INTO` can be used to insert, update and delete from/to a table.**

**Explanation**

The syntax for the simplest `MERGE INTO` statement is as follows:

```
MERGE INTO table1
USING table2
ON condition
WHEN MATCHED THEN action
WHEN NOT MATCHED THEN action
```

As you can see, there are **different sets of actions** when the record is **matched** or **not matched**. Usually, when the records are **matched**, they are **updated or deleted** whereas if the record is **not matched** it is **inserted**. Moreover, the `INSERT INTO` command can only be used to **insert** values in a table whereas `MERGE INTO` is more versatile as it can be used to perform **insertion, deletion and updation** in a table using a **single statement**.

More Info: MERGE INTO | INSERT INTO

Question 25

Which of the following about the Multi-hop architecture is true?

- ◯

  **Multi-hop architecture can be used only for batch workloads**

- [✓ ●]

  **In Multi-hop architecture, the main task of the Bronze table is to apply the schema to the raw data**

- ◯

  **Multi-hop architecture can only be performed in SQL**

- ◯

  **For multi-hop architecture to perform quickly, SQL endpoints are necessary**

- ◯

  **Most of the multi-hop architectures include Gold-Diamond-Platinum tables**

**Explanation**

In a multi-hop architecture, the **Bronze table** mainly performs the task of **applying schema** over the ingested data. Rest of the options are all false. The following are the major tasks of each table.

**Bronze** – *Apply schema to the raw data*

**Silver** – *Conversion of timestamps into human readable format and performing joins with other tables*

**Gold** – *Performing aggregations*

More Info: [Multi-hop (medallion) architecture in Databricks]()

Question 26

A data engineer is using AutoLoader for ingesting CSV data from S3 location. What should replace the blank to execute the code correctly.

```
dataDF = spark.readstream.____________
.option("cloudFiles.format", "csv")
.option("cloudFiles.schemaLocation", schemaLocation)
.load(source)
```

- `autoloader`
- `format("autoLoader")`
- [✓ `format("cloudFiles")`]
- `option("autoLoader")`
- `option("cloudFiles")`

**Explanation**

Whenever you need to use **AutoLoader** you need to use the `format` as `cloudFiles`. So, `format("cloudFiles")` will replace the blank to run the code correctly.

More Info: Using cloudFiles format in AutoLoader

Question 27

Which of the following users can use the Gold table as a source?

- ○

  **A user that needs to feed the raw data to a table**

- [✓ ●]

  **A user that needs to design a dashboard using aggregated data**

- ○

  **A user that needs to add a column to the table**

- ○

  **The Gold table is the end of multi-hop architecture and is not used by any user**

- ○

  **A user that needs to join static data with streaming data to make the data richer**

**Explanation**

**Gold Table** is the **last** layer in multi-hop architecture. It has all the **aggregated** data and can be consumed by the **dashboards** or **reporting tools**.

More Info: Uses of Gold table in medallion architecture

Question 28

A data analyst has created a Bronze table containing 20 million records by ingesting raw data. A data engineer wants to test the data by running a query over the table. The table, being a part of a production environment, cannot be used by the data engineer directly. The creation of views on the original table has also been restricted. Which of the following approaches can you suggest for the data engineer to quickly test the data?

- ○

  **The data engineer can create `DEEP CLONE` of the table and run the query over the newly created table**

- ○

  **The data engineer can request the data analyst to query over the original table**

- [✓ ●]

  **The data engineer can create `SHALLOW CLONE` of the table and run the query over the newly created table**

- ○

  **The data engineer can request the admin to run the query**

- ○

  **The data engineer can use Python's `spark.sql()` function to query the data**

**Explanation**

Let us see which of the above options can be used by the data engineer to quickly test the data.

*The data engineer can create* `DEEP CLONE` *of the table and run the query over the newly created table*

**INCORRECT**. This can be one of the solutions but as given in the

Question, the **Bronze table contains 20 million records** and creating a `DEEP CLONE` of the table will take a **lot of time**. `DEEP CLONE` command is used to create a copy of the table which needs to be used **repeatedly**. Also note, while creating the `DEEP CLONE` of the table all the data needs to be copied to a new location incrementally which is a **time taking process**.

*The data engineer can request the data analyst to query over the original table*

**INCORRECT**. According to the

Question itself, the **original table cannot be used**.

*The data engineer can create* `SHALLOW CLONE` *of the table and run the query over the newly created table*

**CORRECT**. Creation of `SHALLOW CLONE` is the **quickest** option as it does **not** involve any data movement. In `SHALLOW CLONE`, only the definition(schema) of the original table is copied but the newly created table refers to the **same data files** which makes the process faster. It is more suited for **testing and experimentation**.

*The data engineer can request the admin to run the query*

**INCORRECT**. It is not a valid option.

*The data engineer can use Python's* `spark.sql()` *function to query the data*

**INCORRECT**. Python does support `spark.sql()` function but querying the original table is **not** allowed, as per the

Question.

More Info: Selecting the correct type of cloning (Shallow or Deep)

Question 29

A team of data analysts is using CTE (Common Table Expression) as part of their SQL queries to be used for one of the Bronze tables in multi-hop architecture. All of the data analysts try to explain the usage of CTE to a newly joined member. Which of the following statements about CTE is/are correct?

Data Analyst 1 – CTE can be created using `WITH` command

Data Analyst 2 – CTE allows a result to be used multiple times

Data Analyst 3 – CTE cannot be nested

Data Analyst 4 – CTE can be created using `CTE` command

- ○ **Data analysts 1, 2, 3 are right but data analyst 4 Is wrong**
- ○ **All the data analysts are right**
- ○ **Data analysts 2 and 3 are right but the data analysts 1 and 4 are wrong**
- [✓ ● **Data analysts 1 and 2 are right but the data analysts 3 and 4 are wrong**]
- ○ **All the data analysts are wrong except data analyst 1**

**Explanation**

Let's look at each of the data analysts' statements:

**Data Analyst 1 –** *CTE can be created using WITH command*

**RIGHT**. `WITH` is the only option to create a CTE.

**Data Analyst 2 –** *CTE allows a result to be used multiple times*

**RIGHT**. The main purpose of CTE is to create a set of result which can be reused several times

**Data Analyst 3 –** ***CTE cannot be nested***

**WRONG**. There can be CTE within a CTE. So, they can be nested

**Data Analyst 4 –** *CTE can be created using* `CTE` *command*

**WRONG**. As discussed earlier, CTE can be created using `WITH` command only. There is no `CTE` command in SQL.

More Info: CTE(Common Table Expression) in SQL

Question 30

A data engineer is working on a project which involves writing a multi-hop architecture using Python. A data analyst also needs to work in the same project but knows only SQL. Which of the following can be done by the data engineer to assure both of them work in their own layers of the multi-hop architecture?

- ○ **Use `%python` in the cell and run SQL queries**
- ● **Change the default language of the notebook to SQL**
- ○ **Register a UDF**
- ○ **Contact Databricks Administrator to change the language**
- ○ **Create a temporary view using `createOrReplaceTempView()` function**

**Explanation**
Whenever you need to use a **Python data object in SQL**, a **temporary view** can be created which can be used to run SQL commands. The temporary view can be registered using the `createOrReplaceTempView()` function.

More Info: Using Python's DataFrames in SQL

Question 31

Which of the following SQL statements counts the number of unique rows from the Silver table `routes`?

- ○ `SELECT count(*) FROM routes;`
- [✓ ● `SELECT count(DISTINCT *) FROM routes;`]
- ○ `SELECT count_if(* is DISTINCT) FROM routes;`
- ○ `SELECT count(*) FROM routes WHERE * is DISTINCT;`
- ○ `SELECT count(UNIQUE (*)) FROM routes;`

**Explanation**
Let us look at all the options one by one

`SELECT count(*) FROM routes`

**INCORRECT**. It will count the total number of rows from the table `routes` and not the unique (distinct) rows.

`SELECT count(DISTINCT *) FROM routes`

**CORRECT**. This will print the count of distinct rows from the table `routes`. Also note, `DISTINCT (*)` can also be used instead of `DISTINCT *`

`SELECT count_if(* is DISTINCT) FROM routes`

**INCORRECT**. This is not a valid statement and will return an error.

`SELECT count(*) FROM routes WHERE * is DISTINCT`

**INCORRECT**. This is also an invalid statement and will return an error.

`SELECT count(UNIQUE (*)) FROM routes`

**INCORRECT**. `UNIQUE` is not a valid keyword, `DISTINCT` should be used instead.

More Info: count() function in SQL

Question 32

In which of the following layers of multi-hop architecture the most common operation is aggregation?

- [✓ **Gold**]
- **Silver**
- **Bronze**
- **Raw**
- **Diamond**

**Explanation**
Let us go through the **layers of medallion architecture** or the multi-hop architecture in Databricks.

Firstly, the **raw data** is ingested into the **Bronze** table.

Next, the **Silver** table is created using columns from different Bronze tables optionally **joined** with some static tables. Also, a major addition in the Silver table is conversion of Linux based timestamp column into **human readable time** (e.g. HH:mm:ss).

Lastly, The **Gold** table is the collection of major **aggregations** from the Silver table(s). The **Gold** table is then used for **reporting and dashboards**.

Coming back to this

Question, **aggregation** is the most common operation in the **Gold** table.

More Info: Multi-hop (medallion) architecture in Databricks

Question 33

Which of the following can be a source for the Bronze table in Incremental multi-hop architecture?

1. Kafka Stream
2. Silver table
3. Gold Table
4. JSON data
5. Raw data

- ○ **1, 2, 3, 4**
- ○ **1, 2, 3**
- ○ **1, 3, 4, 5**
- [✓ **1, 4, 5**]
- ○ **2, 3, 5**

**Explanation**

The data flows from **Bronze to Silver** and then to **Gold**. The Bronze table contains **raw data** and **cannot** accept **enriched data** from the **Silver** table or **aggregated data** from the **Gold** table.

The Bronze table accepts sources like raw streams including **Kafka Stream**, various data formats including **JSON, CSV, AVRO** etc.

So, all the options given above **except Silver and Gold table** can become a source for Bronze table in a multi-hop architecture.

More Info: [Bronze table in multi-hop (medallion) architecture]

Question 34

A data engineer is using AutoLoader for a streaming ETL process and a new file has arrived with a different schema in the source directory. This newly added file has an extra column named `last_name`. What could be the possible outcome when the AutoLoader processes this file?

- ●

  **The AutoLoader process will fail with an error**

- ○

  **The AutoLoader will discard that column with no reference to that column**

- ○

  **The process will stop and the user can choose the next step**

- ○

  **The process will continue with the details of added column `last_name` stored in `_rescued_data` column**

  (Correct)

- ○

  **The new file will be auto deleted and will not be processed**

**Explanation**

**AutoLoader** can rescue the data which does not fit the schema by using an **extra column** in the schema named `_rescued_data`. The process continues as usual and the column `_rescued_data` gets updated as soon as an **incompatible record** arrives.

More Info: [Use of _rescued_data column in AutoLoader]()

Question 35

A junior data engineer has joined a team and needs to create a DLT pipeline. Which of the following describes the flow of actions to create a new pipeline?

1. Select the **Jobs** pane from the left side menu
2. Select **Run Pipeline**
3. Select **Create Pipeline**
4. Select **Compute** pane from the left side menu
5. Click on **Create DLT Pipeline**
6. Click on **Delta Live Tables** tab
7. Drag the **DLT** pane from left side menu to the notebook cell

- ○ **1 -> 5 -> 3**
- ○ **7 -> 6 -> 2**
- [✓ ●] **1 -> 6 -> 3**

  **(Correct)**
- ○ **7 -> 6 -> 3**
- ○ **4 -> 7 -> 2**

**Explanation**

This

Question involves detailing about the creation of **DLT pipeline**. If you have created the DLT pipelines, this would be an easy one.

To start, you should know that to create a DLT pipeline you need to select **Jobs** pane from the left side menu. Once you are in the Jobs pane, you need to select **Delta Live Tables** tab and then click on **Create Pipeline**. Correct flow of actions would be 1 -> 6 -> 3.

More Info: [Steps to create a DLT pipeline]

Question 36

A junior data engineer has joined a team working on a project involving creation of Delta Live Table pipelines using SQL. They see the following constraint added to the table:

```
CONSTRAINT age_in_range EXPECT (age > 0 AND age < 60) ON VIOLATION DROP
ROW
```

What will be the effect of this statement on the working of the DLT pipeline?

- ○

  **Every time the value of `age` column is between 0 and 60, the pipeline will fail**

- ○

  **Every time the value of `age` column is between 0 and 60, the row will be dropped**

- ○

  **Every time the value of `age` column is not between 0 and 60, the pipeline will fail**

- ●

  **Every time the value of `age` column is not between 0 and 60, the row will be dropped**

  **(Correct)**

- ○

  **The statement has no effect**

**Explanation**

A constraint can be added to a Delta Live Table using `CONSTRAINT` command. Let us look at the statement:

```
CONSTRAINT age_in_range EXPECT (age > 0 AND age < 60) ON VIOLATION DROP
ROW
```

Now, the confusing part of the

Question is the condition. Let us review the condition:

`(age > 0 and age < 60)`

This condition is the **expected condition**, which means that if the condition is not met the **row will be dropped**. In simpler terms, if the `age` is not between 0 and 60, the row will be dropped.

More Info: Dropping invalid records from a Delta Live Table

Question 37

Which of the following can be combined with a DLT pipeline?

- ○

  **Medallion architecture – Bronze, Silver and Gold tables**

- ○

  **AutoLoader**

- ○

  **Constraints on tables**

- [✓ ●]

  **All of these**

  **(Correct)**

- ○

  **None of these**

**Explanation**

All these techniques mentioned above can be combined with a DLT pipeline.

More Info: Using medallion architecture and AutoLoader with a DLT pipeline | Constraints on Delta Live Tables

Question 38

A team of data analysts is running queries on a SQL endpoint and the queries are running at a decent speed. Now, the number of users has increased massively from 1 to 20 and because of that the queries have been running very slow. The cluster size has already been set to the maximum but still the queries are on a slower side. What can you suggest to make the queries run faster for all the users?

- ○

  **Request Databricks Administrator to increase the speed**

- ○

  **Turn on the Auto-stop feature**

- ○

  **Increase the Max bound of cluster scaling range**

  **(Correct)**

- [✓ ●]

  **Turn on the Serverless feature**

- ○

  **Decrease the cluster size and increase the Max bound of cluster scaling range**

**Explanation**

***With the latest update from Databricks, SQL endpoint is now known as SQL warehouse.***

To answer this

Question you should know the **different options** for creating a new **SQL endpoint(warehouse)**. While creating a SQL endpoint(warehouse) some of the **common features** which you can **configure** as per your need are as follows:

1. **Name** – Name given to each endpoint(warehouse) to **uniquely identify** the endpoints(warehouses).
2. **Cluster size** – Required size of the cluster (*T-shirt size e.g. Small, Medium, X-Large etc.*) to be selected from the drop down menu. **Higher the cluster size**, **lower the latency** in running SQL queries.
3. **Scaling** – **Min and Max cluster values** for scaling to be entered. Scaling the endpoint(warehouse) to **more** clusters will **increase the speed** for all the users working on that endpoint(warehouse).
4. **Advanced features** – It includes features like **adding Tags** or **turning on the Serverless** feature.

As far as this

Question is concerned, the **cluster size** has **already** been set to the **largest available**. So, the **Max bound** of cluster **scaling range** needs to be **increased**. If **multiple** users are using the **same** SQL endpoint(warehouse), **increasing the Max bound** of cluster **scaling range** will allow **more** users to run their queries smoothly.

More Info: SQL endpoint(warehouse) properties

Question 39

Which of the following statements is true for cluster pools in Databricks?

- ○

  **By using cluster pools, you can reduce start time for a cluster**

- ○

  **It maintains several clusters in idle state and can be used when necessary**

- ○

  **By using cluster pools, you can reduce auto-scaling time for a cluster**

- ○

  **Same cluster pool can be used for a driver node and worker nodes**

- [✓ ●]

  **All the above statements are true for a cluster pool**

  **(Correct)**

**Explanation**
**Cluster pools** help **speed up** the cluster and its ability to **auto scale**. A set of clusters are attached to the pool. Whenever a cluster is required, it is created from the pool. **Worker nodes and driver node** can share the same pool but **different pools** for worker nodes and driver node can also exist.

***So, if a cluster is taking a lot of time to start, one of the reasons could be that it is not attached to a cluster pool.***

More Info: Cluster pools in Databricks

Question 40

A scheduled query is running every 5 seconds to ingest data from a networking system which contains various network related attributes. The network engineer should be informed through email if the value in the `fault` column increases to 10 or more. Which of the following is the best approach?

- ○

  **A manual email can be sent by the data engineer if the value increases the threshold**

- ○

  **The Databricks administrator can send the email to the network engineer**

- [✓ ●]

  **Databricks Alerts can be used to notify the network engineer through email**

  **(Correct)**

- ○

  **Use the dashboard to check the value of the `fault` column**

- ○

  **The alerting system through email is not yet supported in Databricks**

**Explanation**

To access the alerting system on Databricks, you need to select **Alerts** from the left side navigation menu. (*Make sure you are in Databricks SQL to access Alerts*). The Alerts page allows you to **create, edit and view the alerts**.

An alert can be sent as soon as a column or group of columns reaches a threshold. The alerts can be either **one-time or recurring** based on the selection made while creating the alert. Similarly, the alert destination can also be set including **email, Slack, Webhook** etc.

More Info: Alerts in Databricks

Question 41

Which of the following is optional while creating a Delta Live Table pipeline?

- [✓ **Target Database**]

  **(Correct)**

- **Pipeline name**

- **Notebook Library**

- **Minimum and maximum workers**

- **Pipeline mode**

**Explanation**

Every option except the **Target Database** name is mandatory. **Target Database name is optional** and can be skipped while creating a pipeline.

More Info: Use of target database name in a DLT pipeline

Question 42

Which of the following has been built to provide fine-grained data governance and security in the Lakehouse?

- ○

  **AutoLoader**

- [✓ ●]

  **Unity Catalog**

  **(Correct)**

- ○

  **SQL endpoint**

- ○

  **Cluster**

- ○

  **Data Explorer**

**Explanation**

**Unity Catalog** has been specially built for having a **fine-grained data governance and security** in Lakehouse. Before the introduction of Unity Catalog, it was close to impossible to provide **access to a part of data** since the permissions were set to **files and directories**. Unity Catalog eradicates this problem by allowing access on **row, column and view level**.

More Info: Unity Catalog in Databricks

Question 43

Which of the following queries can be used to `REVOKE` all the permissions from user `bob@candes.db` on database `courses`?

- ○ `REVOKE ALL PRIVILEGES ON courses DB FROM `bob@candes.db``
- ○ `REVOKE ALL PERMISSIONS ON DATABASE courses FROM `bob@candes.db``
- ○ `REVOKE ALL PRIVILEGES FROM courses SCHEMA TO `bob@candes.db``
- ○ `REVOKE ALL PRIVILEGES ON courses DATABASE FROM USER `bob@candes.db``
- [✓ `REVOKE ALL PRIVILEGES ON DATABASE courses FROM `bob@candes.db``]

  (Correct)

**Explanation**

The syntax for revoking permissions from a user or a group is as follows:

```
REVOKE [privilege_type] ON [data_object_type] [data_object_name] FROM
[user_or_group_name]
```

**REVOKE** – Command to revoke permission

**privilege_type** – It includes type of privileges like `SELECT`, `VIEW` or `ALL PRIVILEGES`

**data_object_type** – It can be one of `TABLE`, `SCHEMA` or `DATABASE`, `CATALOG`, `FUNCTION` etc.

**data_object_name** – The name of the database, table, catalog etc.

**user_or_group_name** – Name of the user or group from which the privileges need to be revoked.

After understanding the `REVOKE` syntax, you can easily tell why the last option is correct.

More Info: Using REVOKE command

Question 44

A data engineer is the owner of the `organization` database. Which of the following permissions cannot be controlled by the Databricks administrator?

- ○ **Grant permissions to users to the tables in `organization` database**
- ○ **Revoke permissions from users for accessing the `organization` database**
- ○ **View all the grants on `organization` database**
- [✓ ◉] **Revoke permissions from the owner of the `organization` database**

  **(Correct)**
- ○ **Grant permission to other users to the `organization` database**

**Explanation**

The **Databricks administrator** is the one who can change the owner of a table or a database but **cannot revoke privileges from an owner of the table or database**. The admin **can** view permissions, grant permissions to or revoke permissions from the users or groups.

More Info: [Databricks administrator limitations]

Question 45

A Databricks administrator needs to view all the grants to the user `abc@def.com` for database `university`. Which of the following commands can be used?

- ○

  `` SHOW GRANTS `abc@def.com` ON university ``

- ●

  `` SHOW ALL GRANTS TO `abc@def.com` ON DATABASE university ``

- ○

  `` SHOW GRANTS `abc@def.com` ON DATABASE university ``

  **(Correct)**

- ○

  `` VIEW GRANTS TO `abc@def.com` ON DATABASE university ``

- ○

  `` SHOW ALL GRANTS TO `abc@def.com` university ``

**Explanation**

Firstly, to view grants on a data object (table, database, view etc.) you need to be one of the following:

**1. Databricks admin** or

**2. Owner of the data object** or

**3. The user for which the grants are being viewed**.

Syntax for viewing the grants for a user on a specific database is:

`` SHOW GRANTS `user_name` ON DATABASE database_name ``

Also note the usage of **backticks** (``) instead of inverted commas(“).

More Info: [Viewing grants on a schema(database) for a user]