Drop, Show, & Rename Views

This lesson explains how to list, rename, and delete views.

SHOW, DROP & RENAME Views

There are two ways to list all views in a database; one is the SHOW FULL TABLES command and the other is querying the information_schema database. The DROP VIEW command is used to delete a view from the database. A view can be renamed in two ways. One is by using the RENAME TABLE command and the other is by deleting and recreating.

Syntax

```
SHOW FULL TABLES

{FROM | IN} db_name

WHERE table_type = 'VIEW'

LIKE pattern;
```

```
DROP VIEW [IF EXISTS] view1, view2,...viewn;
```

```
RENAME TABLE old_name

TO new_name;
```

Connect to the terminal below by clicking in the widget. Once connected,

the command line prompt will show up. Enter or copy and paste the

command **./DataJek/Lessons/45lesson.sh** and wait for the MySQL prompt to start-up.

```
-- The lesson queries are reproduced below for convenient copy/paste into the terminal.

-- Query 1
SHOW FULL TABLES
WHERE table_type = 'VIEW';

-- Query 2
SHOW FULL TABLES
LIKE '%Actor%';

-- Query 3
SELECT table_name
FROM information_schema.TABLES
WHERE table_type = 'VIEW'
AND table_schema = 'MovieIndustry';

-- Query 4
DROP VIEW DigitalAssetCount, ActorAssets;

-- Query 5
DROP VIEW IF EXISTS DigitalAssetCount, ActorAssets;

-- Query 6
CREATE VIEW ActorAge AS
SELECT *
FROM Actors
WHERE TIMESTAMPDIFF(YEAR, DoB, CURDATE()) > 50;

-- Query 7
RENAME TABLE ActorAge
TO ActorsOlderThan50;
```

Terminal

1. In the previous lessons we have used the **SHOW TABLES** command to see the views created. Since the tables and views share the same namespace, this command lists both of them. The **SHOW FULL TABLES** command used with a WHERE clause can be used to show only the views in a database. The **FROM | IN** clause is optional and can be used to see the views from another database.

   ```
   SHOW FULL TABLES
   ```

```
WHERE table_type = 'VIEW';
```

```
mysql> SHOW FULL TABLES
    -> WHERE table_type = 'VIEW';
+-------------------------+------------+
| Tables_in_MovieIndustry | Table_type |
+-------------------------+------------+
| ActorDetails            | VIEW       |
| ActorView               | VIEW       |
| ActorsTwitterAccounts   | VIEW       |
| ActorsView1             | VIEW       |
| ActorsView2             | VIEW       |
| ActorsView3             | VIEW       |
| DigitalAssetCount       | VIEW       |
| RichActors              | VIEW       |
| RichFemaleActors        | VIEW       |
| SingleActors            | VIEW       |
+-------------------------+------------+
10 rows in set (0.00 sec)
```

The LIKE operator can be used to shortlist views based on a word or pattern.

```
SHOW FULL TABLES
LIKE '%Actor%';
```

```
mysql> SHOW FULL TABLES
    -> LIKE '%Actor%';
+-----------------------------------+------------+
| Tables_in_MovieIndustry (%Actor%) | Table_type |
+-----------------------------------+------------+
| ActorDetails                      | VIEW       |
| ActorView                         | VIEW       |
| Actors                            | BASE TABLE |
| ActorsTwitterAccounts             | VIEW       |
| ActorsView1                       | VIEW       |
| ActorsView2                       | VIEW       |
| ActorsView3                       | VIEW       |
| RichActors                        | VIEW       |
| RichFemaleActors                  | VIEW       |
| SingleActors                      | VIEW       |
+-----------------------------------+------------+
10 rows in set (0.00 sec)
```

The query returns 10 rows containing the word 'Actor' of which 9 are views and 1 is a table in our database.

2. The **information_schema** database is a catalogue of all MYSQL databases and contains metadata such as database names, tables, privileges, and datatypes of columns, etc. A query against this

database can also list all views of a particular database as follows:

```
SELECT table_name
FROM information_schema.TABLES
WHERE table_type = 'VIEW'
AND table_schema = 'MovieIndustry';
```

```
mysql> SELECT table_name
    -> FROM information_schema.TABLES
    -> WHERE table_type = 'VIEW'
    -> AND table_schema = 'MovieIndustry';
+----------------------+
| table_name           |
+----------------------+
| ActorDetails         |
| ActorView            |
| ActorsTwitterAccounts |
| ActorsView1          |
| ActorsView2          |
| ActorsView3          |
| DigitalAssetCount    |
| RichActors           |
| RichFemaleActors     |
| SingleActors         |
+----------------------+
10 rows in set (0.00 sec)
```

3. We can delete one or more views at a time using the **DROP VIEW** statement. In the absence of the **IF EXISTS** clause, MYSQL gives an error if the view to be dropped does not exist. With this clause, a warning is generated if a view we wish to delete is not found in the database. Execute the following query:

```
DROP VIEW DigitalAssetCount, ActorAssets;
```

```
mysql> DROP VIEW DigitalAssetCount, ActorAssets;
ERROR 1051 (42S02): Unknown table 'MovieIndustry.ActorAssets'
mysql>
```

Mysql throws an error message. Re-run the above query with the IF EXISTS clause:

```
DROP VIEW IF EXISTS DigitalAssetCount, ActorAssets;
```

```
mysql>
mysql> DROP VIEW IF EXISTS DigitalAssetCount, ActorAssets;
```

```
Query OK, 0 rows affected, 2 warnings (0.00 sec)
```

This time, we get a warning and the query is executed as can be seen from the list of views:

```
mysql> SELECT table_name
    -> FROM information_schema.TABLES
    -> WHERE table_type = 'VIEW'
    -> AND table_schema = 'MovieIndustry';
+----------------------+
| table_name           |
+----------------------+
| ActorDetails         |
| ActorView            |
| ActorsTwitterAccounts |
| ActorsView1          |
| ActorsView2          |
| ActorsView3          |
| RichActors           |
| RichFemaleActors     |
| SingleActors         |
+----------------------+
9 rows in set (0.00 sec)
```

4. Views are stored in the same namespace as tables, hence, the **RENAME TABLE** command can be used for renaming views. To show how RENAME works, we will create a view as follows:

```
CREATE VIEW ActorAge AS
SELECT *
FROM Actors
WHERE TIMESTAMPDIFF(YEAR, DoB, CURDATE()) > 50;
```

```
mysql> CREATE VIEW ActorAge AS
    -> SELECT *
    -> FROM Actors
    -> WHERE TIMESTAMPDIFF(YEAR, DoB, CURDATE()) > 50;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW FULL TABLES WHERE table_type ='VIEW';
+--------------------------+------------+
| Tables_in_MovieIndustry | Table_type |
+--------------------------+------------+
| ActorAge                 | VIEW       |
| ActorDetails             | VIEW       |
| ActorView                | VIEW       |
| ActorsTwitterAccounts    | VIEW       |
| ActorsView1              | VIEW       |
| ActorsView2              | VIEW       |
| ActorsView3              | VIEW       |
| RichActors               | VIEW       |
| RichFemaleActors         | VIEW       |
| SingleActors             | VIEW       |
+--------------------------+------------+
10 rows in set (0.01 sec)
```

Next, we will change its name to ActorsOlderThan50.

```
RENAME TABLE ActorAge
TO ActorsOlderThan50;
```

The RENAME was successful as seen below:

```
mysql> RENAME TABLE ActorAge
    -> TO ActorsOlderThan50;
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW FULL TABLES
    -> WHERE table_type = 'VIEW';
+--------------------------+------------+
| Tables_in_MovieIndustry | Table_type |
+--------------------------+------------+
| ActorDetails             | VIEW       |
| ActorView                | VIEW       |
| ActorsOlderThan50        | VIEW       |
| ActorsTwitterAccounts    | VIEW       |
| ActorsView1              | VIEW       |
| ActorsView2              | VIEW       |
| ActorsView3              | VIEW       |
| RichActors               | VIEW       |
| RichFemaleActors         | VIEW       |
| SingleActors             | VIEW       |
+--------------------------+------------+
10 rows in set (0.00 sec)
```

There is another method to change the name of a view without using the RENAME clause. First, copy the query used to create the view, then drop the view, and lastly create a new one from the DDL copied

then drop the view, and lastly create a new one from the DDL copied

in the first step. The **SHOW CREATE VIEW** query is used to show the DDL of the view.