GROUP BY

This lesson demonstrates using the GROUP BY clause.

GROUP BY

The **GROUP BY**, as the name suggests, sorts rows together into groups. The clause returns one row for each group. Data is organized using a comma separated list of columns as the criteria specified after the **GROUP BY** clause. The GROUP BY statement is often used with aggregate functions such as COUNT, MAX, MIN, SUM, and AVG to calculate an aggregated stat for each group.

Syntactically, the **GROUP BY** clause must appear after the **FROM** and **WHERE** clauses and is also evaluated after them. However, **GROUP BY** is evaluated before the **ORDER BY**, **LIMIT**, and **HAVING** clauses.

Example Syntax

> SELECT **col1, AggregateFunction(col3)**
>
> FROM **table**;
>
> GROUP BY **col1, col2, ... coln**
>
> ORDER BY **col2**;

> 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/23lesson.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
SELECT FirstName FROM Actors GROUP BY FirstName;

-- Query 2
SELECT FirstName, SecondName FROM Actors GROUP BY FirstName;

-- Query 3
SELECT Gender, COUNT(*) FROM Actors GROUP BY Gender;

-- Query 4
SELECT Gender FROM Actors GROUP BY Gender;

-- Query 5
SELECT MaritalStatus, AVG(NetworthInMillions) FROM Actors GROUP BY MaritalStatus ORDER BY Mar
```

Terminal

1. To get a feel for using the **GROUP BY** clause, we'll write a query to group results by first name. We already know the table consists of rows with all unique first names and we'll be returned eleven groups equal to the number of rows in the table.

```
SELECT FirstName FROM Actors GROUP BY FirstName;
```

```
mysql> SELECT FirstName FROM Actors GROUP BY FirstName;
+-----------+
| FirstName |
+-----------+
| Amitabh   |
| Angelina  |
| Brad      |
| Jennifer  |
| Johnny    |
| Kim       |
| Kylie     |
| Natalie   |
| priyanka  |
| Shahrukh  |
| Tom       |
+-----------+
```

```
11 rows in set (0.00 sec)
```

2. Note that in the previous query the **SELECT** clause also specifies the same column name as the **GROUP BY** clause. The following query would fail:

```
SELECT FirstName, SecondName FROM Actors GROUP BY FirstName;
```

Recall that **GROUP BY** returns one row for each group. Each group will have the same value for FirstName but may have different values for the SecondName. Thus, it is nonsensical to associate any one value for SecondName with the entire group and display SecondName in the output. Attempting to do so results in a syntax error as shown below:

```
ERROR 1055 (42000): Expression #2 of SELECT list is not in GROUP
BY clause and contains nonaggregated column 'MovieIndustry.Actor
s.SecondName' which is not functionally dependent on columns in G
ROUP BY clause; this is incompatible with sql_mode=only_full_grou
p_by
```

We can't have non-aggregated columns in the **SELECT**, **ORDER BY**, and **HAVING** clauses when these columns don't appear in the **GROUP BY** clause or are functionally dependent on columns that do appear. Though there are exceptions, when the non-aggregated column has a single value it can appear in the **SELECT**, **ORDER BY**, and **HAVING** clauses. This restriction is ensured by setting **sql_mode** to **only_full_group_by**. We can unset this setting and retry our query, but the value chosen for SecondName would be arbitrary if multiple actors share the same first name.

3. Now we'll try something more useful with the **GROUP BY** clause. If we are asked to find the number of male and female actors, we can use group by gender to fulfill the query.

```
SELECT Gender, COUNT(*) FROM Actors GROUP BY Gender;
```

```
mysql> SELECT Gender, COUNT(*) FROM Actors GROUP BY Gender;
+--------+----------+
| Gender | COUNT(*) |
+--------+----------+
| Male   |        6 |
| Female |        5 |
+--------+----------+
2 rows in set (0.00 sec)
```

Each group has the count function applied to it which outputs the rows in each group. If we don't ask for the count, the query will return the two groups it finds:

```
SELECT Gender FROM Actors GROUP BY Gender;
```

```
mysql> SELECT Gender FROM Actors GROUP BY Gender;
+--------+
| Gender |
+--------+
| Male   |
| Female |
+--------+
2 rows in set (0.00 sec)
```

4. As another example, we can find the average net worth of actors according to their marital status as follows:

```
SELECT MaritalStatus, AVG(NetworthInMillions) FROM Actors GROUP B
Y MaritalStatus ORDER BY MaritalStatus ASC;
```