Union

This lesson demonstrates how to combine results from several queries.

Union

The **UNION** clause allows us to combine the results from several queries together. The clause doesn't join the table but merely clubs the two results together.

Syntax

> **<Query1>**
>
> UNION
>
> **<Query2>**

> 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/27lesson.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
UNION
SELECT URL FROM DigitalAssets;

-- Query 2
(SELECT CONCAT(FirstName, ' ', SecondName) AS "Actor Name"
FROM Actors
ORDER BY NetworthInMillions DESC
```

```
ORDER BY NetworthInMillions DESC
LIMIT 2)
UNION
(SELECT CONCAT(FirstName, ' ', SecondName) AS "ThisAliasIsIgnored"
FROM Actors
ORDER BY NetworthInMillions ASC
LIMIT 2);

-- Query 3
SELECT FirstName, Id FROM Actors
UNION
SELECT FirstName FROM Actors;

-- Query 4
SELECT FirstName, Id FROM Actors
UNION
SELECT FirstName, null FROM Actors;

-- Query 5
SELECT MaritalStatus FROM Actors
UNION
SELECT Gender FROM Actors;

-- Query 6
SELECT MaritalStatus FROM Actors
UNION ALL
SELECT Gender FROM Actors;

-- Query 7
(SELECT CONCAT(FirstName, ' ', SecondName) AS "Actor Name"
FROM Actors
ORDER BY NetworthInMillions DESC  LIMIT 2)
UNION
(SELECT NetworthInMillions
FROM Actors
ORDER BY NetworthInMillions ASC);

-- Query 8
(SELECT CONCAT(FirstName, ' ', SecondName) AS "Actor Name"
FROM Actors
ORDER BY NetworthInMillions DESC  LIMIT 2)
UNION
(SELECT NetworthInMillions
FROM Actors
ORDER BY NetworthInMillions ASC LIMIT 3);
```

Terminal

1. As a contrived example, we'll write a query that prints all the first names from the **Actors** table and all the URLs from the **DigitalAssets** table.

   ```
   SELECT FirstName FROM Actors
   ```

```
UNION



SELECT URL FROM DigitalAssets;
```

```
mysql> SELECT FirstName FROM Actors
    ->
    -> UNION
    ->
    -> SELECT URL FROM DigitalAssets;
+-----------------------------------------------+
| FirstName                                     |
+-----------------------------------------------+
| Brad                                          |
| Jennifer                                      |
| Angelina                                      |
| Johnny                                        |
| Natalie                                       |
| Tom                                           |
| Kylie                                         |
| Kim                                           |
| Amitabh                                       |
| Shahrukh                                      |
| priyanka                                      |
| http://jennifer-aniston.org                   |
| http://www.angelina-jolie.com                 |
| http://www.tomcruise.com                      |
| https://twitter.com/iamsrk                    |
| https://twitter.com/jenniferannistn           |
| https://twitter.com/joliestweet               |
| https://twitter.com/KimKardashian             |
| https://twitter.com/natpdotcom                |
| https://twitter.com/TomCruise                 |
| https://www.bradpittweb.com                   |
| https://www.facebook.com/IamSRK               |
| https://www.facebook.com/JenniferAniston      |
| https://www.facebook.com/JohnChristopherOfficial |
| https://www.facebook.com/KimKardashian        |
| https://www.facebook.com/natalieportmandotcom |
| https://www.facebook.com/officialtomcruise    |
| https://www.instagram.com/bradpittoficial     |
| https://www.kkwbeauty.com                     |
| https://www.natalieportman.com                |
| https://www.pinterest.com/angelinajolie5601   |
| https://www.pinterest.com/natalieportmandotcom |
+-----------------------------------------------+
32 rows in set (0.00 sec)
```

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

2. A more realistic example would be a query where you are required to print the top two richest actors and the least two richest.

```
(SELECT CONCAT(FirstName, ' ', SecondName) AS "Actor Name"
FROM Actors
ORDER BY NetworthInMillions DESC
LIMIT 2)

UNION

(SELECT CONCAT(FirstName, ' ', SecondName) AS "ThisAliasIsIgnore
d"
FROM Actors
ORDER BY NetworthInMillions ASC
LIMIT 2);
```

```
mysql> (SELECT CONCAT(FirstName, ' ', SecondName) AS "Actor Name" FROM Actors ORDER BY NetworthInMillions DESC LIMIT 2)
    ->
    -> UNION
    ->
    ->
    -> (SELECT CONCAT(FirstName, ' ', SecondName) AS "ThisAliasIsIgnored" FROM Actors ORDER BY NetworthInMillions ASC LIMIT 2);
+-----------------+
| Actor Name      |
+-----------------+
| Kylie Jenner    |
| Shahrukh Khan   |
| priyanka Chopra |
| Natalie Portman |
+-----------------+
4 rows in set (0.00 sec)
```

Note we have used the various techniques learned so far to enhance the above query. We use the concat function to join first and second names with a space separating them and use the alias "Actor Name" for the resulting column. The alias from the second query is ignored. Furthermore, we wrap the two queries in parentheses which is a requirement when using the order by or limit clause in subqueries of a union query.

3. When using the **UNION** clause, the two result sets being combined should have the same number and order of columns. The columns from the result sets should be of the same type or types that are compatible. For instance, the following query will error out:

```
SELECT FirstName, Id FROM Actors

UNION
```

```
SELECT FirstName FROM Actors;
```

```
mysql> SELECT FirstName, Id FROM Actors
    ->
    -> UNION
    ->
    -> SELECT FirstName FROM Actors;
ERROR 1222 (21000): The used SELECT statements have a different number of columns
```

To make the above query work, we can insert a *fake column* or null as follows:

```
SELECT FirstName, Id FROM Actors

UNION

SELECT FirstName, null FROM Actors;
```

```
mysql> SELECT FirstName, Id FROM Actors
    ->
    -> UNION
    ->
    -> SELECT FirstName, null FROM Actors;
+-----------+------+
| FirstName | Id   |
+-----------+------+
| Brad      |    1 |
| Jennifer  |    2 |
| Angelina  |    3 |
| Johnny    |    4 |
| Natalie   |    5 |
| Tom       |    6 |
| Kylie     |    7 |
| Kim       |    8 |
| Amitabh   |    9 |
| Shahrukh  |   10 |
| priyanka  |   11 |
| Brad      | NULL |
| Jennifer  | NULL |
| Angelina  | NULL |
| Johnny    | NULL |
| Natalie   | NULL |
| Tom       | NULL |
| Kylie     | NULL |
| Kim       | NULL |
| Amitabh   | NULL |
| Shahrukh  | NULL |
| priyanka  | NULL |
+-----------+------+
22 rows in set (0.00 sec)
```

4. Observe the output of the following query:

```
SELECT MaritalStatus FROM Actors

UNION

SELECT Gender FROM Actors;
```

```
mysql> SELECT MaritalStatus FROM Actors
    ->
    -> UNION
    ->
    -> SELECT Gender FROM Actors;
+---------------+
| MaritalStatus |
+---------------+
| Single        |
| Married       |
| Divorced      |
| Male          |
| Female        |
+---------------+
5 rows in set (0.00 sec)
```

Note that the union clause doesn't output duplicate values and works similarly to the distinct clause. If we want duplicate values to be included in the query result, we need to use the **UNION ALL** clause as follows:

```
SELECT MaritalStatus FROM Actors

UNION ALL

SELECT Gender FROM Actors;
```

```
mysql> SELECT MaritalStatus FROM Actors
    ->
    -> UNION ALL
    ->
    -> SELECT Gender FROM Actors;
+---------------+
| MaritalStatus |
+---------------+
| Single        |
| Single        |
| Single        |
| Single        |
| Married       |
| Divorced      |
| Married       |
| Married       |
| Married       |
| Married       |
| Married       |
| Male          |
| Female        |
| Female        |
| Male          |
| Male          |
| Male          |
| Female        |
| Female        |
| Male          |
| Male          |
| Female        |
+---------------+
```

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

5. Another quirk of the **UNION** clause is that it may ignore the **ORDER BY** clause when used without the **LIMIT** clause in a subquery. Consider the below query:

```
(SELECT CONCAT(FirstName, ' ', SecondName) AS "Actor Name"
FROM Actors
ORDER BY NetworthInMillions DESC  LIMIT 2)

UNION

(SELECT NetworthInMillions
FROM Actors
ORDER BY NetworthInMillions ASC);
```

```
mysql> (SELECT CONCAT(FirstName, ' ', SecondName) AS "Actor Name"
    -> FROM Actors
    -> ORDER BY NetworthInMillions DESC  LIMIT 2)
    ->
    -> UNION
    ->
    -> (SELECT NetworthInMillions
    -> FROM Actors
    -> ORDER BY NetworthInMillions ASC);
+---------------+
| Actor Name    |
+---------------+
| Kylie Jenner  |
| Shahrukh Khan |
| 240           |
| 100           |
| 200           |
| 60            |
| 570           |
| 1000          |
| 370           |
| 400           |
| 600           |
| 28            |
+---------------+
12 rows in set (0.00 sec)
```

Note that in the second subquery we order the rows in an ascending fashion by the column NetworthInMillions. The values in the result set are, however, printed disregarding the order. When we tack on a **LIMIT** clause to the second subquery the IDs show up in descending

LIMIT clause to the second subquery the IDs show up in descending order as shown below:

```
(SELECT CONCAT(FirstName, ' ', SecondName) AS "Actor Name"
FROM Actors
ORDER BY NetworthInMillions DESC  LIMIT 2)

UNION

(SELECT NetworthInMillions
FROM Actors
ORDER BY NetworthInMillions ASC LIMIT 3);
```

```
mysql> (SELECT CONCAT(FirstName, ' ', SecondName) AS "Actor Name"
    -> FROM Actors
    -> ORDER BY NetworthInMillions DESC  LIMIT 2)
    ->
    -> UNION
    ->
    -> (SELECT NetworthInMillions
    -> FROM Actors
    -> ORDER BY NetworthInMillions ASC LIMIT 3);
+---------------+
| Actor Name    |
+---------------+
| Kylie Jenner  |
| Shahrukh Khan |
| 28            |
| 60            |
| 100           |
+---------------+
5 rows in set (0.01 sec)
```

The astute reader would also notice that the types of the two columns in the result set aren't the same. The query works because MySQL converts the int to varchar.