ORDER BY

This lesson discusses how to use the ORDER BY clause.

ORDER BY Clause

In the previous couple of sections, we have seen how to query data from a table. The retrieved rows aren't printed in any particular order. The **ORDER BY** clause allows us to print the retrieved rows in an orderly fashion based on the criteria we specify.

Example Syntax

> SELECT **col1, col2, ... coln**
>
> FROM **table**
>
> WHERE **col3 LIKE "%some-string%"**
>
> ORDER BY **col3**

> 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/11lesson.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 * FROM Actors ORDER BY FirstName;

-- Query 2
```

```
SELECT * FROM Actors ORDER BY FirstName DESC;

-- Query 3

SELECT * FROM Actors ORDER BY NetWorthInMillions, FirstName;

-- Query 4
SELECT * FROM Actors ORDER BY NetWorthInMillions, SecondName;

-- Query 5
SELECT * FROM Actors ORDER BY NetWorthInMillions DESC, FirstName ASC;

-- Query 6
SELECT * FROM Actors ORDER BY NetWorthInMillions DESC, FirstName DESC;

-- Query 7
SELECT * FROM Actors ORDER BY BINARY FirstName;

-- Query 8
SELECT * FROM Actors ORDER BY NetWorthInMillions;

-- Query 9
SELECT * FROM Actors ORDER BY CAST(NetWorthInMillions AS CHAR);
```

Terminal

1. Suppose we want to print the names of all the actors sorted in alphabetical order. We can do so using the following query:

   ```
   SELECT * FROM Actors ORDER BY FirstName;
   ```

   ```
   mysql> SELECT * FROM Actors ORDER BY FirstName;
   +----+-----------+------------+------------+--------+---------------+--------------------+
   | Id | FirstName | SecondName | DoB        | Gender | MaritalStatus | NetWorthInMillions |
   +----+-----------+------------+------------+--------+---------------+--------------------+
   |  9 | Amitabh   | Bachchan   | 1942-10-11 | Male   | Married       |                400 |
   |  3 | Angelina  | Jolie      | 1975-06-04 | Female | Single        |                100 |
   |  1 | Brad      | Pitt       | 1963-12-18 | Male   | Single        |                240 |
   |  2 | Jennifer  | Aniston    | 1969-11-02 | Female | Single        |                240 |
   |  4 | Johnny    | Depp       | 1963-06-09 | Male   | Single        |                200 |
   |  8 | Kim       | Kardashian | 1980-10-21 | Female | Married       |                370 |
   |  7 | Kylie     | Jenner     | 1997-08-10 | Female | Married       |               1000 |
   |  5 | Natalie   | Portman    | 1981-06-09 | Male   | Married       |                 60 |
   | 11 | priyanka  | Chopra     | 1982-07-18 | Female | Married       |                 28 |
   | 10 | Shahrukh  | Khan       | 1965-11-02 | Male   | Married       |                600 |
   |  6 | Tom       | Cruise     | 1962-07-03 | Male   | Divorced      |                570 |
   +----+-----------+------------+------------+--------+---------------+--------------------+
   11 rows in set (0.00 sec)
   ```

   The **ORDER BY** clause is followed by the column name on which we intend to sort. This column is called the **sort key**. By default, the sorting is case-insensitive and in ascending order. Sorting of string columns depends on the character set being used and the collation order.

2. We can also print the rows in descending order as the following query demonstrates:

```
SELECT * FROM Actors ORDER BY FirstName DESC;
```

```
mysql> SELECT * FROM Actors ORDER BY FirstName DESC;
+----+-----------+------------+------------+--------+---------------+--------------------+
| Id | FirstName | SecondName | DoB        | Gender | MaritalStatus | NetWorthInMillions |
+----+-----------+------------+------------+--------+---------------+--------------------+
|  6 | Tom       | Cruise     | 1962-07-03 | Male   | Divorced      |                570 |
| 10 | Shahrukh  | Khan       | 1965-11-02 | Male   | Married       |                600 |
| 11 | priyanka  | Chopra     | 1982-07-18 | Female | Married       |                 28 |
|  5 | Natalie   | Portman    | 1981-06-09 | Male   | Married       |                 60 |
|  7 | Kylie     | Jenner     | 1997-08-10 | Female | Married       |               1000 |
|  8 | Kim       | Kardashian | 1980-10-21 | Female | Married       |                370 |
|  4 | Johnny    | Depp       | 1963-06-09 | Male   | Single        |                200 |
|  2 | Jennifer  | Aniston    | 1969-11-02 | Female | Single        |                240 |
|  1 | Brad      | Pitt       | 1963-12-18 | Male   | Single        |                240 |
|  3 | Angelina  | Jolie      | 1975-06-04 | Female | Single        |                100 |
|  9 | Amitabh   | Bachchan   | 1942-10-11 | Male   | Married       |                400 |
+----+-----------+------------+------------+--------+---------------+--------------------+
11 rows in set (0.01 sec)
```

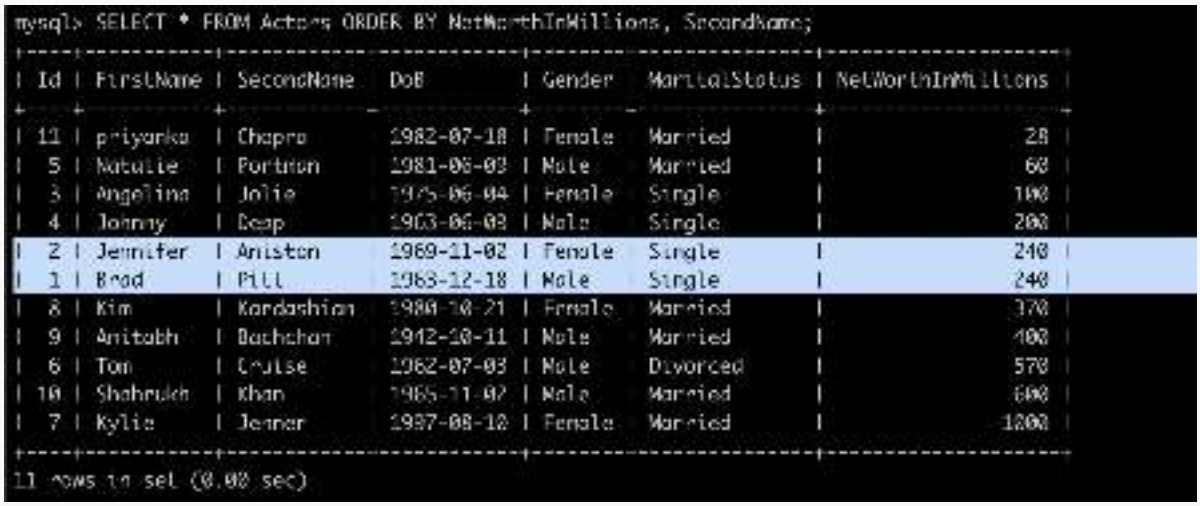
3. We can also specify more than one sort key. In this case if a tie occurs based on the first sort key, it is broken using the second sort key. For example, some actors in our table have the same net worth. We can specify net worth as the first sort key and the first name as the second sort key as follows:

```
SELECT * FROM Actors ORDER BY NetWorthInMillions, FirstName;
```

```
mysql> SELECT * FROM Actors ORDER BY NetWorthInMillions, FirstName;
+----+-----------+------------+------------+--------+---------------+--------------------+
| Id | FirstName | SecondName | DoB        | Gender | MaritalStatus | NetWorthInMillions |
+----+-----------+------------+------------+--------+---------------+--------------------+
| 11 | priyanka  | Chopra     | 1982-07-18 | Female | Married       |                 28 |
|  5 | Natalie   | Portman    | 1981-06-09 | Male   | Married       |                 60 |
|  3 | Angelina  | Jolie      | 1975-06-04 | Female | Single        |                100 |
|  4 | Johnny    | Depp       | 1963-06-09 | Male   | Single        |                200 |
|  1 | Brad      | Pitt       | 1963-12-18 | Male   | Single        |                240 |
|  2 | Jennifer  | Aniston    | 1969-11-02 | Female | Single        |                240 |
|  8 | Kim       | Kardashian | 1980-10-21 | Female | Married       |                370 |
|  9 | Amitabh   | Bachchan   | 1942-10-11 | Male   | Married       |                400 |
|  6 | Tom       | Cruise     | 1962-07-03 | Male   | Divorced      |                570 |
| 10 | Shahrukh  | Khan       | 1965-11-02 | Male   | Married       |                600 |
|  7 | Kylie     | Jenner     | 1997-08-10 | Female | Married       |               1000 |
+----+-----------+------------+------------+--------+---------------+--------------------+
11 rows in set (0.00 sec)
```

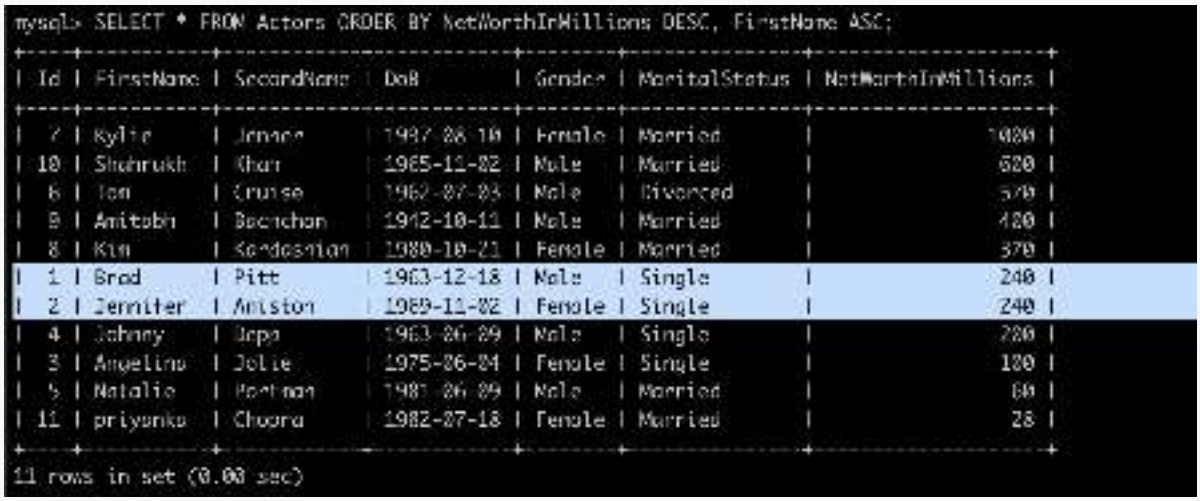
Note the highlighted lines above show Brad Pitt first and Jennifer Aniston second even though both have the same net worth as the tie is broken by the second sort key which is the first name. If we make the second sort key second name, then the row for Jennifer Aniston will rank higher as shown below:

```
SELECT * FROM Actors ORDER BY NetWorthInMillions, SecondName;
```



4. We can also control the ascending or descending order we desire for each sort key. Consider the following query:

```
SELECT * FROM Actors ORDER BY NetWorthInMillions DESC, FirstName
ASC;
```



Note that Kylie Jenner now ranks at the top with a net worth of 1 billion USD. The rows are first sorted in descending order based on net worth of the actors. The highlighted rows show Brad and Jennifer who are tied on net worth. The second sort key decides who ranks higher and as the letter B occurs before the letter J, the row for Brad Pitt is shown first. If we change the sort order for the second sort key to descending, then the row for Jennifer Aniston will rank higher as shown below:

```
SELECT * FROM Actors ORDER BY NetWorthInMillions DESC, FirstName
DESC;
```

```
mysql> SELECT * FROM Actors ORDER BY NetWorthInMillions DESC, FirstName DESC;
+----+-----------+------------+------------+--------+---------------+---------------------+
| Id | FirstName | SecondName | DoB        | Gender | MaritalStatus | NetWorthInMillions  |
+----+-----------+------------+------------+--------+---------------+---------------------+
|  7 | Kylie     | Jenner     | 1997-08-10 | Female | Married       |                1000 |
| 10 | Shahrukh  | Khan       | 1965-11-02 | Male   | Married       |                 600 |
|  6 | Tom       | Cruise     | 1962-07-03 | Male   | Divorced      |                 570 |
|  9 | Amitabh   | Bachchan   | 1942-10-11 | Male   | Married       |                 400 |
|  8 | Kim       | Kardashian | 1980-10-21 | Female | Married       |                 370 |
|  2 | Jennifer  | Aniston    | 1969-11-02 | Female | Single        |                 240 |
|  1 | Brad      | Pitt       | 1963-12-18 | Male   | Single        |                 240 |
|  4 | Johnny    | Depp       | 1963-06-09 | Male   | Single        |                 200 |
|  3 | Angelina  | Jolie      | 1975-06-04 | Female | Single        |                 100 |
|  5 | Natalie   | Portman    | 1981-06-09 | Male   | Married       |                  60 |
| 11 | priyanka  | Chopra     | 1982-07-18 | Female | Married       |                  28 |
+----+-----------+------------+------------+--------+---------------+---------------------+
11 rows in set (0.00 sec)
```

5. Sort order is undefined in case sort keys have the same value for rows.

6. MySQL ignores case when comparing strings in the ORDER BY clause, which implies strings "Kim", "kIm" and "kim" are treated equally. If we want ASCII comparison we need to specify the **BINARY** keyword before the sort key. To demonstrate the effect of the **BINARY** keyword, execute the following query and observe the results:

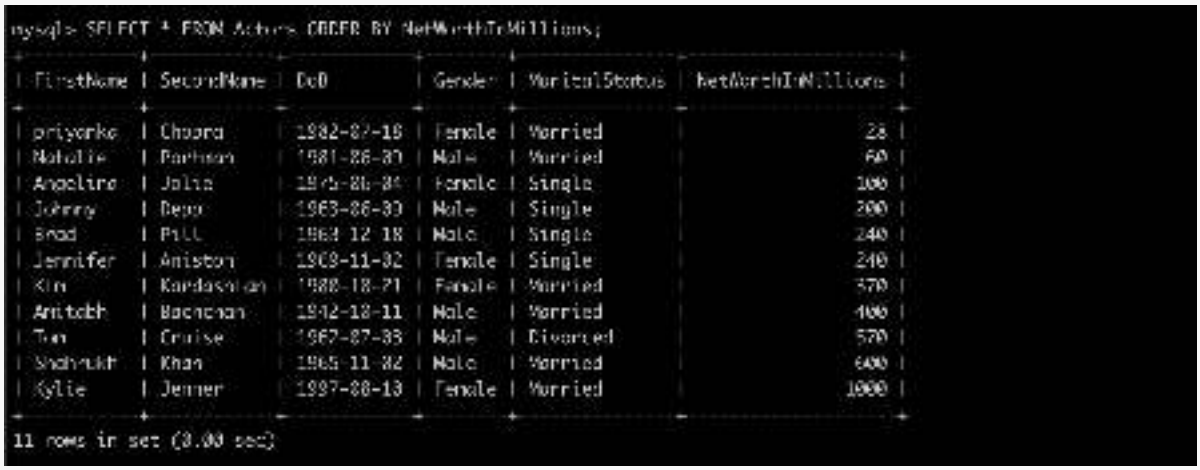
```
SELECT * FROM Actors ORDER BY BINARY FirstName;
```

```
mysql> SELECT * FROM Actors ORDER BY BINARY FirstName;
+----+-----------+------------+------------+--------+---------------+---------------------+
| Id | FirstName | SecondName | DoB        | Gender | MaritalStatus | NetWorthInMillions  |
+----+-----------+------------+------------+--------+---------------+---------------------+
|  9 | Amitabh   | Bachchan   | 1942-10-11 | Male   | Married       |                 400 |
|  3 | Angelina  | Jolie      | 1975-06-04 | Female | Single        |                 100 |
|  1 | Brad      | Pitt       | 1963-12-18 | Male   | Single        |                 240 |
|  2 | Jennifer  | Aniston    | 1969-11-02 | Female | Single        |                 240 |
|  4 | Johnny    | Depp       | 1963-06-09 | Male   | Single        |                 200 |
|  8 | Kim       | Kardashian | 1980-10-21 | Female | Married       |                 370 |
|  7 | Kylie     | Jenner     | 1997-08-10 | Female | Married       |                1000 |
|  5 | Natalie   | Portman    | 1981-06-09 | Male   | Married       |                  60 |
| 10 | Shahrukh  | Khan       | 1965-11-02 | Male   | Married       |                 600 |
|  6 | Tom       | Cruise     | 1962-07-03 | Male   | Divorced      |                 570 |
| 11 | priyanka  | Chopra     | 1982-07-18 | Female | Married       |                  28 |
+----+-----------+------------+------------+--------+---------------+---------------------+
11 rows in set (0.00 sec)
```

   Note that the row for Priyanka Chopra occurs last because upper case letters appear before lower case letters in ASCII ordering.

7. The **CAST** function can also be used with the **ORDER BY** clause. The **CAST** function allows us to treat a column as a different type. For example, the Actors table can be sorted on the NetWorthInMillions as follows:

```
SELECT * FROM Actors ORDER BY NetWorthInMillions;
```



The NetWorthInMillions column is sorted numerically from smallest to largest. We can also sort the NetWorthInMillions column as if strings using the **CAST** function as follows:

```
SELECT * FROM Actors ORDER BY CAST(NetWorthInMillions AS CHAR);
```