Alias

This lesson explains the concept of using an alias for a table.

Alias

Aliases are like nicknames, a temporary name given to a table or a column to write expressive and readable queries. We can use aliases with columns, tables, and MySQL functions.

Example Syntax

SELECT **col1**

AS **aliasCol1**

FROM **table**;

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/20lesson.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 AS PopularName from Actors;

-- Query 2
SELECT CONCAT(FirstName,' ', SecondName) AS FullName FROM Actors;

-- Query 3
SELECT CONCAT(FirstName,' ', SecondName) AS FullName FROM Actors ORDER BY FullName;
```

```
-- Query 4
SELECT CONCAT(FirstName,' ', SecondName) FROM Actors ORDER BY CONCAT(FirstName,' ', SecondNam

-- Query 5
SELECT FirstName FROM Actors AS tbl WHERE tbl.FirstName='Brad' AND tbl.NetWorthInMillions > 2

-- Query 6
SELECT tbl.FirstName FROM Actors AS tbl WHERE tbl.FirstName='Brad' AND tbl.NetWorthInMillions

-- Query 7
SELECT t1.FirstName, t1.NetworthInMillions
FROM Actors AS t1,
Actors AS t2
WHERE t1.NetworthInMillions = t2.NetworthInMillions
AND t1.Id != t2.Id;
```

Terminal

1. We have the first name column in the Actors table. Since most actors are known by their first names, we can alias the FirstName column as PopularName in the following select query:

```
SELECT FirstName AS PopularName from Actors;
```

```
mysql> SELECT FirstName AS PopularName from Actors;
+-------------+
| PopularName |
+-------------+
| Brad        |
| Jennifer    |
| Angelina    |
| Johnny      |
| Natalie     |
| Tom         |
| Kylie       |
| Kim         |
| Amitabh     |
| Shahrukh    |
| priyanka    |
+-------------+
11 rows in set (0.00 sec)
```

Observe that the column heading in the output shows the alias PopularName instead of FirstName column.

PopularName instead of FirstName column.

2. We can also use aliases with MySQL functions. For instance, we use the concat function to print the full name for an actor as follows:

```
SELECT CONCAT(FirstName,' ', SecondName) AS FullName FROM Actors;
```

```
mysql> SELECT CONCAT(FirstName,' ', SecondName) AS FullName FROM Actors;
+------------------+
| FullName         |
+------------------+
| Brad Pitt        |
| Jennifer Aniston |
| Angelina Jolie   |
| Johnny Depp      |
| Natalie Portman  |
| Tom Cruise       |
| Kylie Jenner     |
| Kim Kardashian   |
| Amitabh Bachchan |
| Shahrukh Khan    |
| priyanka Chopra  |
+------------------+
11 rows in set (0.00 sec)
```

So far so good, but you may question the utility of aliases as the effects may seem only cosmetic. Aliases become powerful when they are used in subqueries or other clauses as shown next.

3. We can now sort the actors by their full names which are displayed by the concat function as follows:

```
SELECT CONCAT(FirstName,' ', SecondName) AS FullName FROM Actors
ORDER BY FullName;
```

```
mysql> SELECT CONCAT(FirstName,' ', SecondName) AS FullName FROM Actors ORDER BY FullName;
+------------------+
| FullName         |
+------------------+
| Amitabh Bachchan |
| Angelina Jolie   |
| Brad Pitt        |
| Jennifer Aniston |
| Johnny Depp      |
| Kim Kardashian   |
| Kylie Jenner     |
| Natalie Portman  |
| priyanka Chopra  |
| Shahrukh Khan    |
| Tom Cruise       |
+------------------+
11 rows in set (0.00 sec)
```

Note the sorted output. Without using the alias feature the same query would be written as follows:

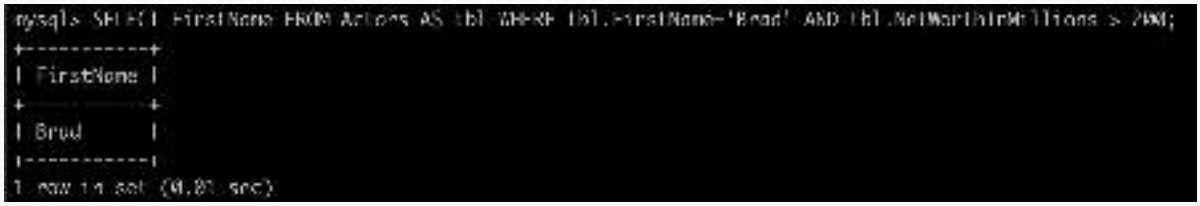
```
SELECT CONCAT(FirstName,' ', SecondName) FROM Actors ORDER BY CON
CAT(FirstName,' ', SecondName);
```

The above query is verbose compared to the one written using the alias.

4. Aliases can be used in **GROUP BY**, **HAVING**, and **ORDER BY** clauses. Notably, aliases for columns can't be used in the WHERE clause but aliases for table can, as shown next.

5. We can use aliases as shorthand for tables too. Table aliases come in handy when working with complex queries that involve joins, which we'll cover later. For now, we'll show a simple example of using an alias for the Actors table in the following select query:

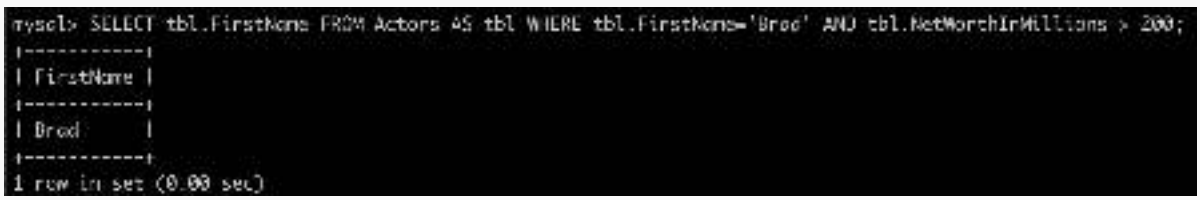
```
SELECT FirstName FROM Actors AS tbl WHERE tbl.FirstName='Brad' AN
D tbl.NetWorthInMillions > 200;
```



Note in the above query, we use **tbl** as the alias for the Actors table and then in the WHERE clause the shorthand is used to refer to the two column names of the Actors table.

6. We can also use the table alias in the SELECT clause before we actually define the alias. We rewrite the previous query and use the alias in the SELECT clause as follows:

```
SELECT tbl.FirstName FROM Actors AS tbl WHERE tbl.FirstName='Bra
d' AND tbl.NetWorthInMillions > 200;
```



7. For some queries table aliases are inevitable. For instance, we can

alias the Actors table twice to find out all the actors with the same net worth in a single query. Think of picking each row and comparing it with the rest of the rows in the table to find two rows with the same **NetWorthInMillions** column. However, the caveat is that we want to skip the row when it tries to match with itself. The complete query is presented below:

```
SELECT t1.FirstName, t1.NetworthInMillions
FROM Actors AS t1,
Actors AS t2
WHERE t1.NetworthInMillions = t2.NetworthInMillions
AND t1.Id != t2.Id;
```