MySQL JOINS

MySQL JOINS are used with SELECT statement. It is used to retrieve data from multiple tables. It is performed whenever you need to fetch records from two or more tables.

There are three types of MySQL joins:

- MySQL INNER JOIN (or sometimes called simple join)
- MySQL LEFT OUTER JOIN (or sometimes called LEFT JOIN)
- MySQL RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)

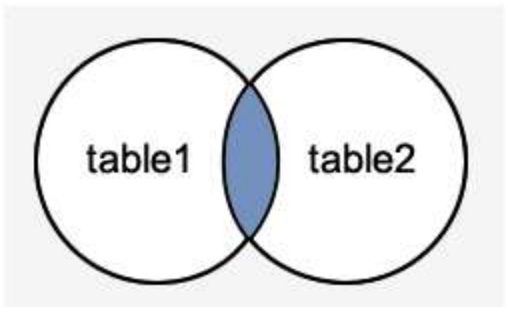
MySQL Inner JOIN (Simple Join)

The MySQL INNER JOIN is used to return all rows from multiple tables where the join condition is satisfied. It is the most common type of join.

Syntax:

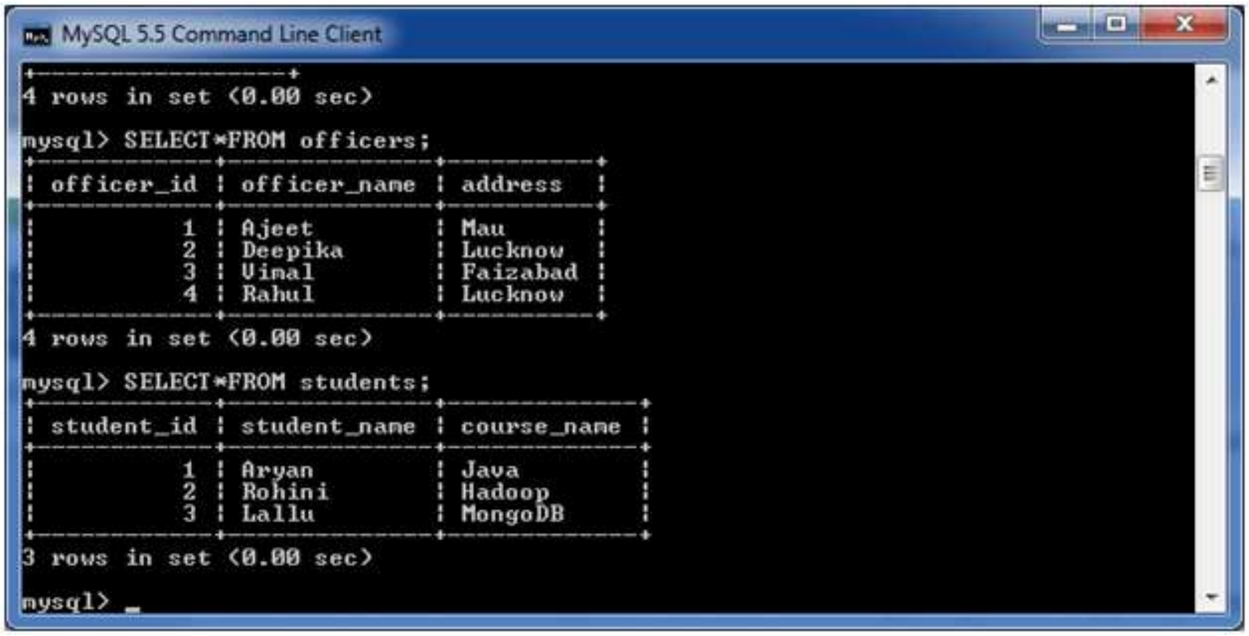
```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column = table2.column;
```

Image representation:



Let's take an example:

Consider two tables "officers" and "students", having the following data.

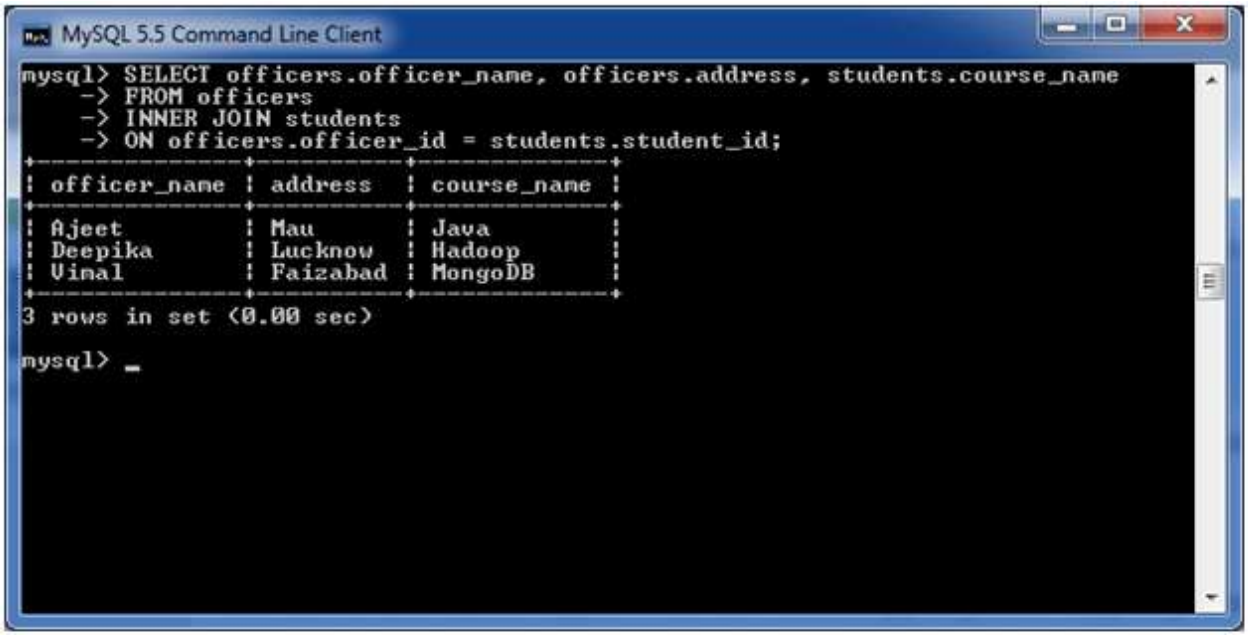


```
+-------------------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM students;
+------------+--------------+-------------+
| student_id | student_name | course_name |
+------------+--------------+-------------+
|          1 | Aryan        | Java        |
|          2 | Rohini       | Hadoop      |
|          3 | Lallu        | MongoDB     |
+------------+--------------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

Execute the following query:

1. **SELECT** officers.officer_name, officers.address, students.course_name
2. **FROM** officers
3. **INNER** JOIN students
4. **ON** officers.officer_id = students.student_id;

Output:



```
mysql> SELECT officers.officer_name, officers.address, students.course_name
    -> FROM officers
    -> INNER JOIN students
    -> ON officers.officer_id = students.student_id;
+--------------+----------+-------------+
| officer_name | address  | course_name |
+--------------+----------+-------------+
| Ajeet        | Mau      | Java        |
| Deepika      | Lucknow  | Hadoop      |
| Vimal        | Faizabad | MongoDB     |
+--------------+----------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

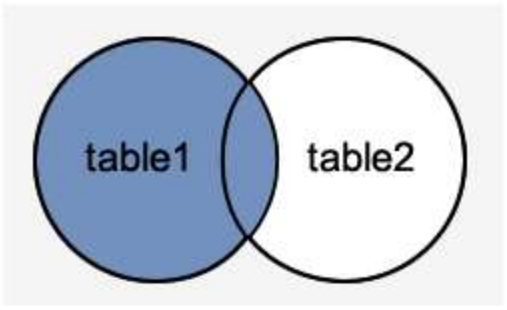
MySQL Left Outer Join

The LEFT OUTER JOIN returns all rows from the left hand table specified in the ON condition and only those rows from the other table where the join condition is fulfilled.

Syntax:

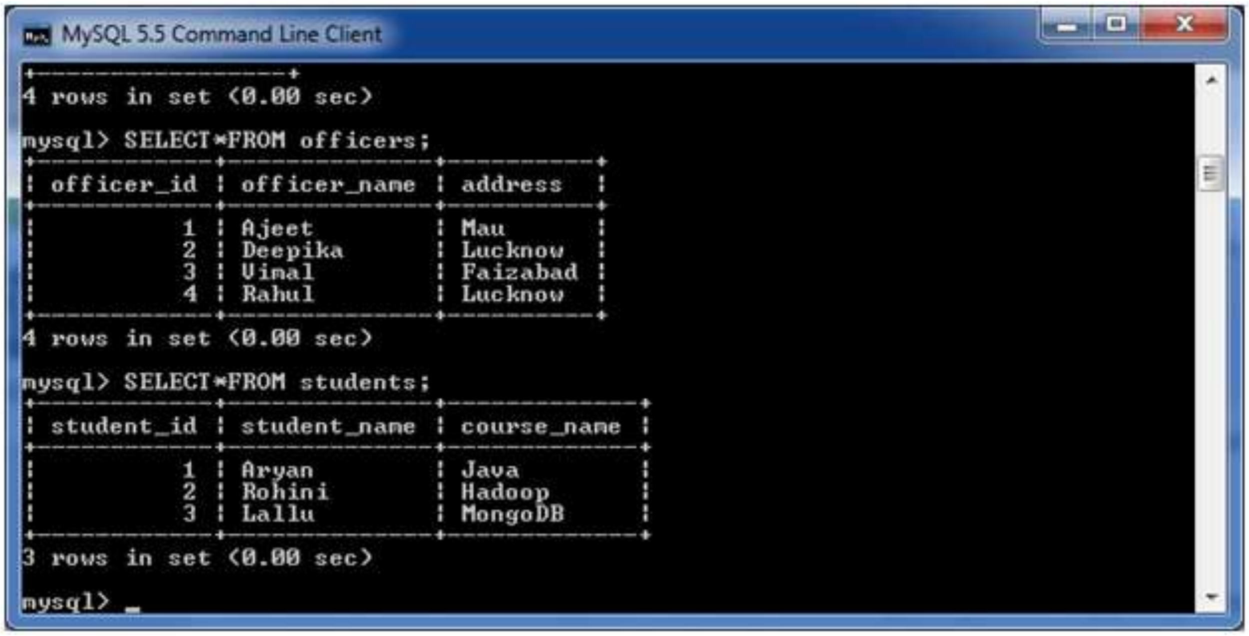
1. **SELECT** columns
2. **FROM** table1
3. LEFT [OUTER] JOIN table2
4. **ON** table1.**column** = table2.**column**;

Image representation:



Let's take an example:

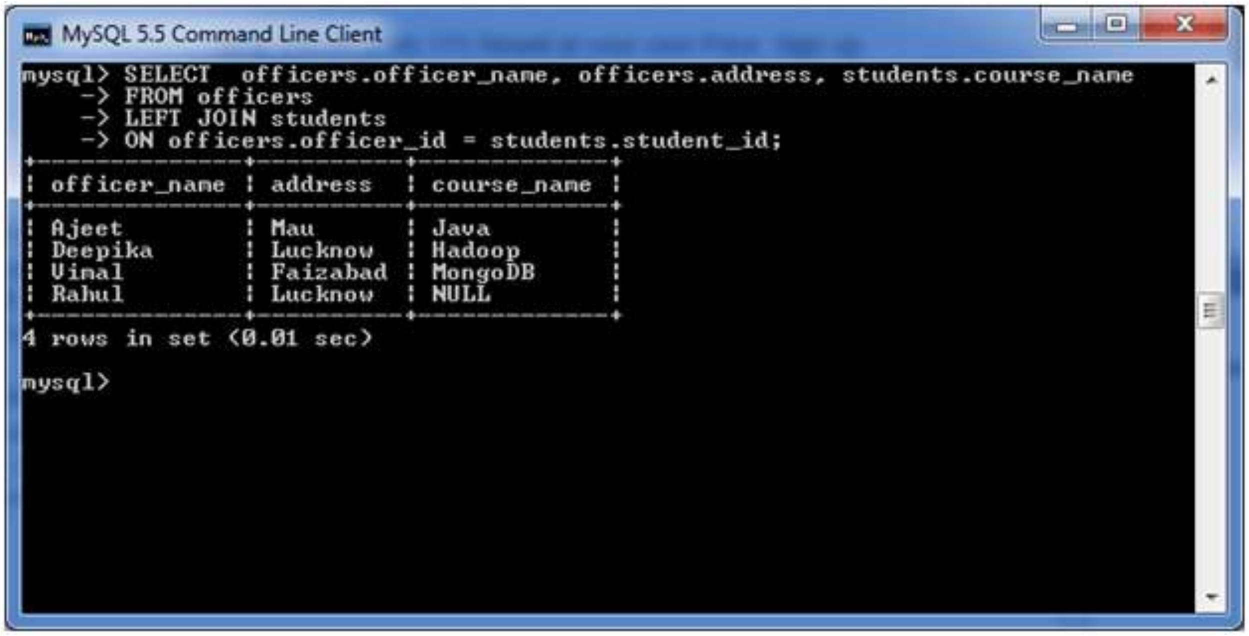
Consider two tables "officers" and "students", having the following data.



```
+--------------------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM students;
+------------+--------------+-------------+
| student_id | student_name | course_name |
+------------+--------------+-------------+
|          1 | Aryan        | Java        |
|          2 | Rohini       | Hadoop      |
|          3 | Lallu        | MongoDB     |
+------------+--------------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

Execute the following query:

1. **SELECT** officers.officer_name, officers.address, students.course_name
2. **FROM** officers
3. LEFT JOIN students
4. **ON** officers.officer_id = students.student_id;



```
mysql> SELECT  officers.officer_name, officers.address, students.course_name
    -> FROM officers
    -> LEFT JOIN students
    -> ON officers.officer_id = students.student_id;
+--------------+----------+-------------+
| officer_name | address  | course_name |
+--------------+----------+-------------+
| Ajeet        | Mau      | Java        |
| Deepika      | Lucknow  | Hadoop      |
| Vimal        | Faizabad | MongoDB     |
| Rahul        | Lucknow  | NULL        |
+--------------+----------+-------------+
4 rows in set (0.01 sec)

mysql>
```

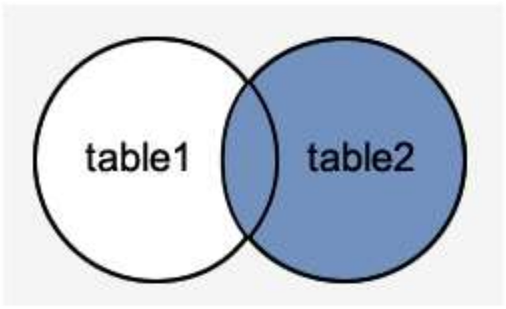
MySQL Right Outer Join

The MySQL Right Outer Join returns all rows from the RIGHT-hand table specified in the ON condition and only those rows from the other table where he join condition is fulfilled.

Syntax:

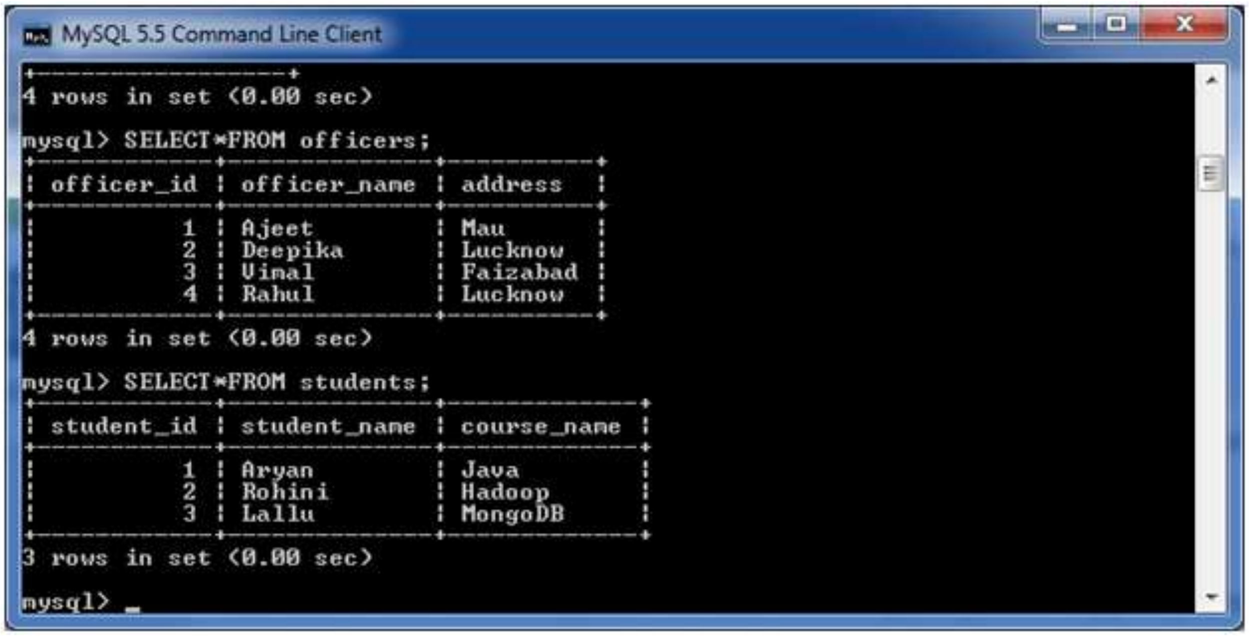
```
1. SELECT columns
2. FROM table1
3. RIGHT [OUTER] JOIN table2
4. ON table1.column = table2.column;
```

Image representation:



Let's take an example:

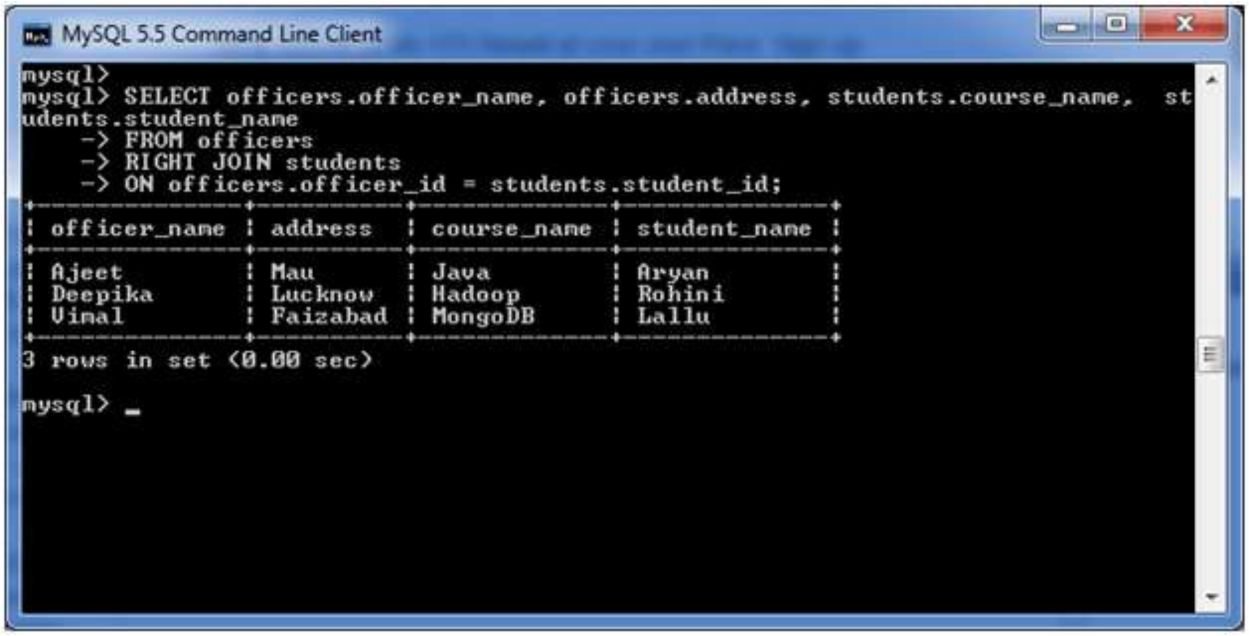
Consider two tables "officers" and "students", having the following data.



```
+--------------------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM students;
+------------+--------------+-------------+
| student_id | student_name | course_name |
+------------+--------------+-------------+
|          1 | Aryan        | Java        |
|          2 | Rohini       | Hadoop      |
|          3 | Lallu        | MongoDB     |
+------------+--------------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

Execute the following query:

```
1. SELECT officers.officer_name, officers.address, students.course_name, students.student_name
2. FROM officers
3. RIGHT JOIN students
4. ON officers.officer_id = students.student_id;
```

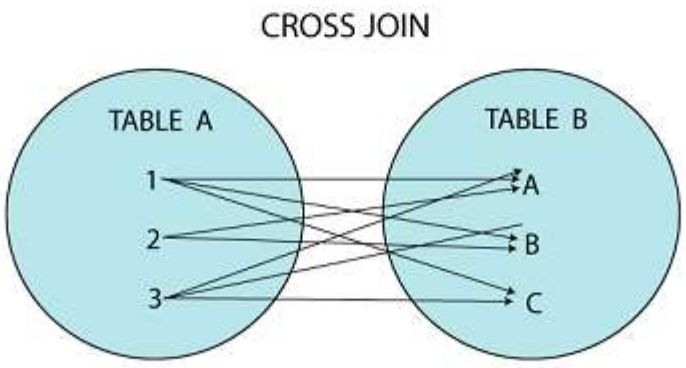


```
mysql>
mysql> SELECT officers.officer_name, officers.address, students.course_name,  st
udents.student_name
    -> FROM officers
    -> RIGHT JOIN students
    -> ON officers.officer_id = students.student_id;
+--------------+----------+-------------+--------------+
| officer_name | address  | course_name | student_name |
+--------------+----------+-------------+--------------+
| Ajeet        | Mau      | Java        | Aryan        |
| Deepika      | Lucknow  | Hadoop      | Rohini       |
| Vimal        | Faizabad | MongoDB     | Lallu        |
+--------------+----------+-------------+--------------+
3 rows in set (0.00 sec)

mysql> _
```

MySQL CROSS JOIN

MySQL CROSS JOIN is used to combine all possibilities of the two or more tables and returns the result that contains every row from all contributing tables. The CROSS JOIN is also known as CARTESIAN JOIN, which provides the Cartesian product of all associated tables. The Cartesian product can be explained as all rows present in the first table multiplied by all rows present in the second table. It is similar to the Inner Join, where the join condition is not available with this clause.

We can understand it with the following visual representation where CROSS JOIN returns all the records from table1 and table2, and each row is the combination of rows of both tables.



MySQL CROSS JOIN Syntax

The CROSS JOIN keyword is always used with the SELECT statement and must be written after the FROM clause. The following syntax fetches all records from both joining tables:

```
SELECT column-lists
FROM table1
CROSS JOIN table2;
```

In the above syntax, the column-lists is the name of the column or field that you want to return and table1 and table2 is the table name from which you fetch the records.

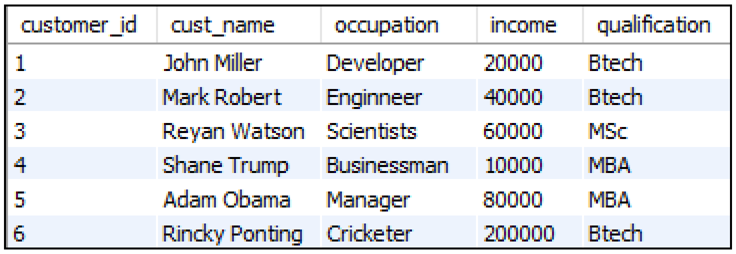
MySQL CROSS JOIN Example

Let us take some examples to understand the working of Left Join or Left Outer Join clause:

CROSS JOIN clause for joining two tables

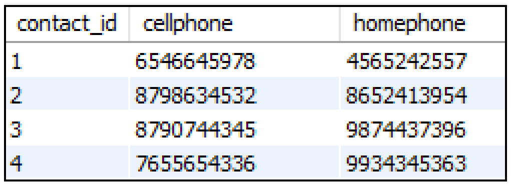
Here, we are going to create two tables **"customers"** and **"contacts"** that contains the following data:

Table: customers



customer_id	cust_name	occupation	income	qualification
1	John Miller	Developer	20000	Btech
2	Mark Robert	Enginneer	40000	Btech
3	Reyan Watson	Scientists	60000	MSc
4	Shane Trump	Businessman	10000	MBA
5	Adam Obama	Manager	80000	MBA
6	Rincky Ponting	Cricketer	200000	Btech

Table: contacts

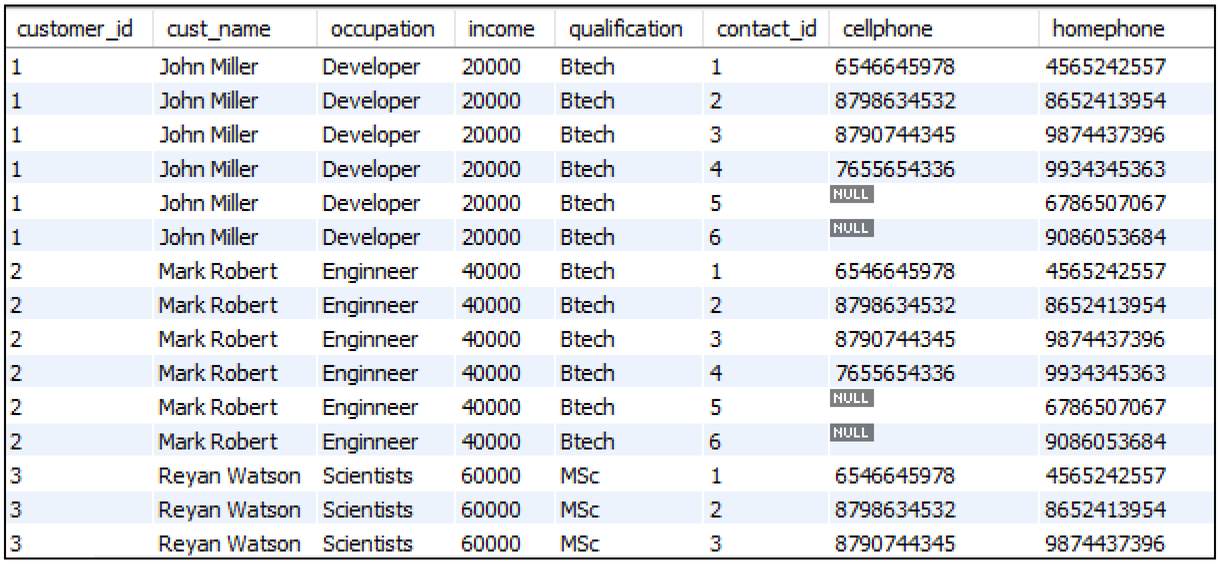


contact_id	cellphone	homephone
1	6546645978	4565242557
2	8798634532	8652413954
3	8790744345	9874437396
4	7655654336	9934345363

To fetch all records from both tables, execute the following query:

```
SELECT *
FROM customers
CROSS JOIN contacts;
```

After successful execution of the query, it will give the following output:



customer_id	cust_name	occupation	income	qualification	contact_id	cellphone	homephone
1	John Miller	Developer	20000	Btech	1	6546645978	4565242557
1	John Miller	Developer	20000	Btech	2	8798634532	8652413954
1	John Miller	Developer	20000	Btech	3	8790744345	9874437396
1	John Miller	Developer	20000	Btech	4	7655654336	9934345363
1	John Miller	Developer	20000	Btech	5	NULL	6786507067
1	John Miller	Developer	20000	Btech	6	NULL	9086053684
2	Mark Robert	Enginneer	40000	Btech	1	6546645978	4565242557
2	Mark Robert	Enginneer	40000	Btech	2	8798634532	8652413954
2	Mark Robert	Enginneer	40000	Btech	3	8790744345	9874437396
2	Mark Robert	Enginneer	40000	Btech	4	7655654336	9934345363
2	Mark Robert	Enginneer	40000	Btech	5	NULL	6786507067
2	Mark Robert	Enginneer	40000	Btech	6	NULL	9086053684
3	Reyan Watson	Scientists	60000	MSc	1	6546645978	4565242557
3	Reyan Watson	Scientists	60000	MSc	2	8798634532	8652413954
3	Reyan Watson	Scientists	60000	MSc	3	8790744345	9874437396

When the CROSS JOIN statement executed, you will observe that it displays 42 rows. It means seven rows from customers table multiplies by the six rows from the contacts table.

Oracle SELF JOIN

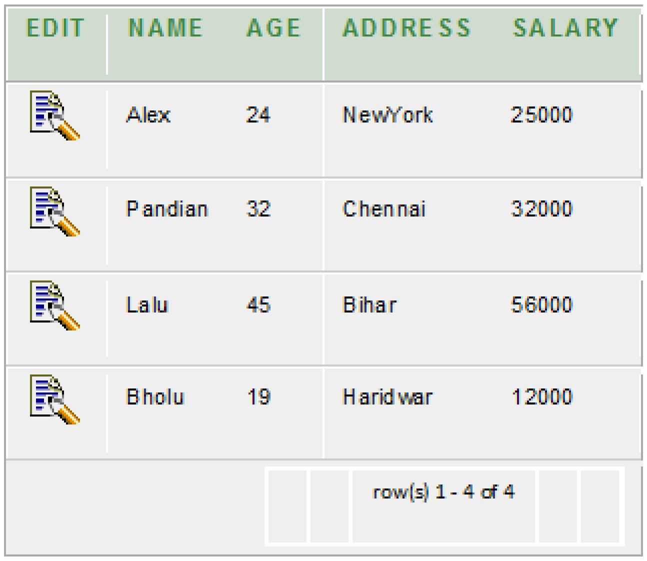
Self Join is a specific type of Join. In Self Join, a table is joined with itself (Unary relationship). A self join simply specifies that each rows of a table is combined with itself and every other row of the table.

Syntax

```
SELECT a.column_name, b.column_name...
FROM table1 a, table1 b
WHERE a.common_filed = b.common_field;
```

Oracle SELF JOIN Example

Let's take a table "customers".



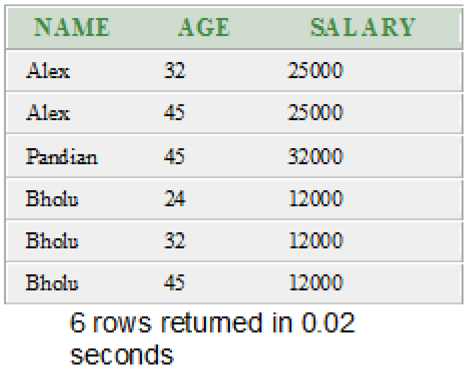
EDIT	NAME	AGE	ADDRESS	SALARY
✎	Alex	24	NewYork	25000
✎	Pandian	32	Chennai	32000
✎	Lalu	45	Bihar	56000
✎	Bholu	19	Haridwar	12000

row(s) 1 - 4 of 4

Join this table using SELF JOIN as follows:

```
SELECT  a.name, b.age, a.SALARY
FROM CUSTOMERS a, CUSTOMERS b
WHERE a.SALARY < b.SALARY;
```

Output



NAME	AGE	SALARY
Alex	32	25000
Alex	45	25000
Pandian	45	32000
Bholu	24	12000
Bholu	32	12000
Bholu	45	12000

6 rows returned in 0.02 seconds