SQL Join
SQL Join is used to fetch data from two or more tables, which is joined to appear as single set of data. SQL Join is used for combining column from two or more tables by using values common to both tables.

Cross JOIN or Cartesian Product

This type of JOIN returns the cartesian product of rows from the tables in Join. It will return a table which consists of records which combines each row from the first table with each row of the second table.

Cross JOIN Syntax is,

```
SELECT column-name-list
from table-name1
CROSS JOIN
table-name2;
```

Example of Cross JOIN

The **class** table,

ID	NAME
1	abhi
2	adam
4	alex

The **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

Cross JOIN query will be,

```
SELECT *
 from class,
 cross JOIN class_info;
```

The result table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	1	DELHI
4	alex	1	DELHI
1	abhi	2	MUMBAI
2	adam	2	MUMBAI
4	alex	2	MUMBAI

1	abhi	3	CHENNAI
2	adam	3	CHENNAI
4	alex	3	CHENNAI

INNER 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;

OR

SELECT columns FROM table1 , table2 WHERE table1.column = table2.column;

Natural JOIN

Natural Join is a type of Inner join which is based on column having same name and same datatype present in both the tables to be joined.

Natural Join Syntax is,

```
SELECT *
from table-name1
NATURAL JOIN
table-name2;
```

Example of Natural JOIN

The **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu

The **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

Natural join query will be,

```
SELECT * from class NATURAL JOIN class_info;
```

The result table will look like,

ID	NAME	Address
1	abhi	DELHI
2	adam	MUMBAI
3	alex	CHENNAI

In the above example, both the tables being joined have ID column(same name and same datatype), hence the records for which value of ID matches in both the tables will be the result of Natural Join of these two tables.

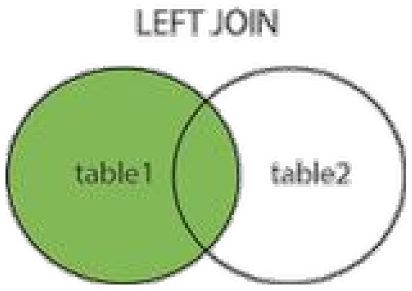
Outer JOIN

Outer Join is based on both matched and unmatched data. Outer Joins subdivide further into,

- Left Outer Join
- Right Outer Join
- Full Outer Join

Left Outer Join

The left outer join returns a result table with the **matched data** of two tables then remaining rows of the **left** table and null for the **right** table's column.



Left Outer Join syntax is,

```
SELECT column-name-list
from table-name1
LEFT OUTER JOIN
table-name2
on table-name1.column-name = table-name2.column-name;
```

Example of Left Outer Join

The **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

The **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI

7	NOIDA
8	PANIPAT

Left Outer Join query will be,

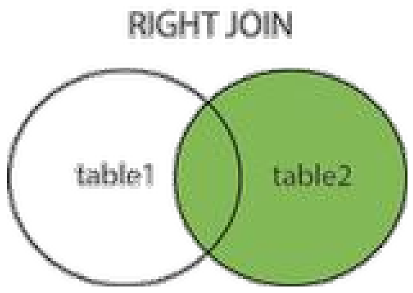
```
SELECT * FROM class LEFT OUTER JOIN class_info ON (class.id=class_info.id);
```

The result table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null

Right Outer Join

The right outer join returns a result table with the **matched data** of two tables then remaining rows of the **right table** and null for the **left** table's columns.



Right Outer Join Syntax is,

```
select column-name-list
from table-name1
RIGHT OUTER JOIN
table-name2
on table-name1.column-name = table-name2.column-name;
```

Example of Right Outer Join

The **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

The **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Right Outer Join query will be,

```
SELECT * FROM class RIGHT OUTER JOIN class_info on (class.id=class_info.id);
```

The result table will look like,

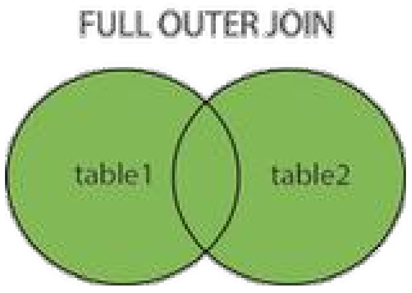
ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
null	null	7	NOIDA
null	null	8	PANIPAT

Full Outer Join

The full outer join returns a result table with the **matched data** of two table then remaining rows of both **left** table and then the **right** table.

Full Outer Join Syntax is,

```
SELECT * FROM t1
LEFT JOIN t2 ON t1.id = t2.id
UNION
SELECT * FROM t1
RIGHT JOIN t2 ON t1.id = t2.id
```



Example of Full outer join is,

The **class** table,

ID	NAME
1	abhi
2	adam
3	alex
4	anu
5	ashish

The **class_info** table,

ID	Address
1	DELHI
2	MUMBAI
3	CHENNAI
7	NOIDA
8	PANIPAT

Full Outer Join query will be like,

```
SELECT * FROM class FULL OUTER JOIN class_info on (class.id=class_info.id);
```

The result table will look like,

ID	NAME	ID	Address
1	abhi	1	DELHI
2	adam	2	MUMBAI
3	alex	3	CHENNAI
4	anu	null	null
5	ashish	null	null
null	null	7	NOIDA
null	null	8	PANIPAT

Using and ON clause

Query 1:

```
SELECT *
FROM users
JOIN orders ON (orders.user_id = users.user_id)
WHERE users.user_id = 1;
```

Query 2:

```
SELECT *
FROM users
JOIN orders USING (user_id)
WHERE user_id = 1;
```

The `USING` clause is something we don't need to mention in the `JOIN` condition when we are retrieving data from multiple tables. When we use `USING` clause, that particular column name should be present in both tables, and the `SELECT` query will automatically join those tables using the given column name in `USING` clause.

for e.g. if there are two common column names in the table, then Mention the desired common column name in the `USING` clause

- The `USING` clause: This allows you to specify the join key by name.
- The `ON` clause: This syntax allows you to specify the column names for join keys in both tables.

The USING clause

The `USING` clause is used if several columns share the same name but you don’t want to join using all of these common columns. The columns listed in the USING clause can’t have any qualifiers in the statement, including the WHERE clause.

The ON clause

The `ON` clause is used to join tables where the column names don’t match in both tables. The join conditions are removed from the filter conditions in the WHERE clause.