# SQL JOIN

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

JOIN Keyword is used in SQL queries for joining two or more tables. Minimum required condition for joining table, is **(n-1)** where **n**, is number of tables. A table can also join to itself, which is known as, **Self Join**.

## Types of JOIN

Following are the types of JOIN that we can use in SQL:

* Inner
* Outer
* Left
* Right

## 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-listFROM table-name1 CROSS JOIN table-name2;

Copy

#### Example of Cross JOIN

Following is the **class** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |
| 4 | alex |

and the **class\_info** table,

|  |  |
| --- | --- |
| **ID** | **Address** |
| 1 | DELHI |
| 2 | MUMBAI |
| 3 | CHENNAI |

Cross JOIN query will be,

SELECT \* FROM

class CROSS JOIN class\_info;

Copy

The resultset 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 |

As you can see, this join returns the cross product of all the records present in both the tables.

## INNER Join or EQUI Join

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the SQL query.

Inner Join Syntax is,

SELECT column-name-list FROM table-name1 INNER JOIN table-name2 WHERE table-name1.column-name = table-name2.column-name;

Copy

#### Example of INNER JOIN

Consider a **class** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |
| 3 | alex |
| 4 | anu |

and the **class\_info** table,

|  |  |
| --- | --- |
| **ID** | **Address** |
| 1 | DELHI |
| 2 | MUMBAI |
| 3 | CHENNAI |

**Inner** JOIN query will be,

SELECT \* from class INNER JOIN class\_info where class.id = class\_info.id;

Copy

The resultset table will look like,

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

### 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.

The syntax for Natural Join is,

SELECT \* FROM table-name1 NATURAL JOIN table-name2;

Copy

#### Example of Natural JOIN

Here is the **class** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |
| 3 | alex |
| 4 | anu |

and the **class\_info** table,

|  |  |
| --- | --- |
| **ID** | **Address** |
| 1 | DELHI |
| 2 | MUMBAI |
| 3 | CHENNAI |

**Natural join query will be,**

SELECT \* from class NATURAL JOIN class\_info;

Copy

The resultset 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,

1. Left Outer Join
2. Right Outer Join
3. Full Outer Join

### LEFT Outer Join

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

Syntax for Left Outer Join is,

SELECT column-name-list FROM table-name1 LEFT OUTER JOIN table-name2ON table-name1.column-name = table-name2.column-name;

Copy

To specify a condition, we use the ON keyword with Outer Join.

Left outer Join Syntax for **Oracle** is,

SELECT column-name-list FROM table-name1, table-name2 on table-name1.column-name = table-name2.column-name(+);

Copy

#### Example of Left Outer Join

Here is the **class** table,

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

and 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);

Copy

The resultset 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 resultset table with the **matched data** from the two tables being joined, then the remaining rows of the **right** table and null for the remaining **left** table's columns.

Syntax for Right Outer Join is,

SELECT column-name-list FROM table-name1 RIGHT OUTER JOIN table-name2 ON table-name1.column-name = table-name2.column-name;

Copy

Right outer Join Syntax for **Oracle** is,

SELECT column-name-list FROM table-name1, table-name2 ON table-name1.column-name(+) = table-name2.column-name;

Copy

#### Example of Right Outer Join

Once again the **class** table,

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

and 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);

Copy

The resultant 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 resultset table with the **matched data** of two table then remaining rows of both **left** table and then the **right** table.

Syntax of Full Outer Join is,

SELECT column-name-list FROM table-name1 FULL OUTER JOIN table-name2ON table-name1.column-name = table-name2.column-name;

Copy

#### Example of Full outer join is,

The **class** table,

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

and 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);

Copy

The resultset 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 |

# SET Operations in SQL

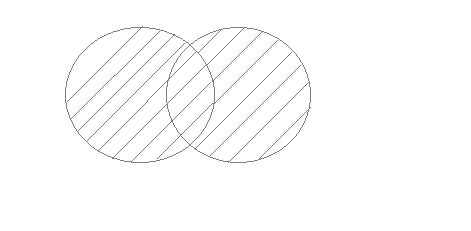
SQL supports few Set operations which can be performed on the table data. These are used to get meaningful results from data stored in the table, under different special conditions.

In this tutorial, we will cover 4 different types of SET operations, along with example:

1. UNION
2. UNION ALL
3. INTERSECT
4. MINUS

## UNION Operation

**UNION** is used to combine the results of two or more SELECT statements. However it will eliminate duplicate rows from its resultset. In case of union, number of columns and datatype must be same in both the tables, on which UNION operation is being applied.



#### Example of UNION

The **First** table,

|  |  |
| --- | --- |
| **ID** | **Name** |
| 1 | abhi |
| 2 | adam |

The **Second** table,

|  |  |
| --- | --- |
| **ID** | **Name** |
| 2 | adam |
| 3 | Chester |

Union SQL query will be,

SELECT \* FROM First UNIONSELECT \* FROM Second;

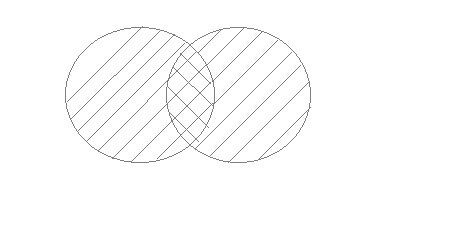
Copy

The resultset table will look like,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |
| 3 | Chester |

## UNION ALL

This operation is similar to Union. But it also shows the duplicate rows.



#### Example of Union All

The **First** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |

The **Second** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 2 | adam |
| 3 | Chester |

Union All query will be like,

SELECT \* FROM First UNION ALLSELECT \* FROM Second;

Copy

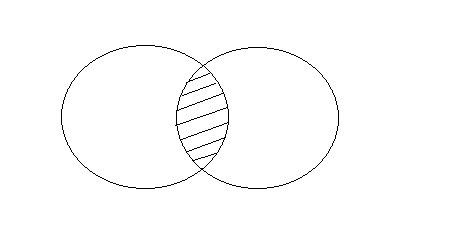
The resultset table will look like,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |
| 2 | adam |
| 3 | Chester |

## INTERSECT

Intersect operation is used to combine two SELECT statements, but it only retuns the records which are common from both SELECT statements. In case of **Intersect** the number of columns and datatype must be same.

**NOTE:** MySQL does not support INTERSECT operator.



#### Example of Intersect

The **First** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |

The **Second** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 2 | adam |
| 3 | Chester |

Intersect query will be,

SELECT \* FROM First INTERSECTSELECT \* FROM Second;

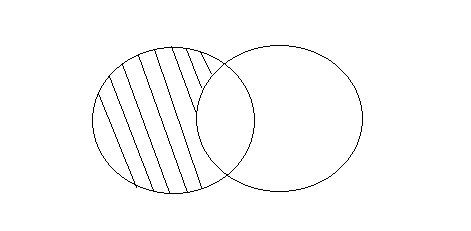
Copy

The resultset table will look like

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 2 | adam |

## MINUS

The Minus operation combines results of two SELECT statements and return only those in the final result, which belongs to the first set of the result.



#### Example of Minus

The **First** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |
| 2 | adam |

The **Second** table,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 2 | adam |
| 3 | Chester |

Minus query will be,

SELECT \* FROM First

MINUSSELECT \* FROM Second;

Copy

The resultset table will look like,

|  |  |
| --- | --- |
| **ID** | **NAME** |
| 1 | abhi |