

Joins

Joins -- Joins are used to combine rows from two or more columns, tables based on related column between them.

A relational database consists of multiple related tables linking together using common columns, which are known as *foreign key* columns.

We use join whenever we want to fetch data from two or more tables.

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.

A join is **an SQL operation performed to establish a connection between two or more database tables based on matching columns, thereby creating a relationship between the tables.**

Joins help retrieving data from two or more database tables.

The tables are mutually related using primary and foreign keys.

Join statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are as follows:

- INNER JOIN(Simple Join)
- LEFT JOIN
- RIGHT JOIN

These Joins are explained below :--

(1.) Inner Join --> Inner join returns the record or selects the record which have matching values in both tables based on foreign key.

To get complete employee data we need to query data from both employee data and city tables data.

That's why joins come into the play.

A join is a method of linking data between one (*self-join*) or more tables based on values of the common column between the tables.

Query for Inner join :-

```
select * from table1 inner join table2 on table1.id = table2.id;
```

The inner join clause compares each row from the first table with every row from the second table.

If values from both rows satisfy the join condition, the inner join clause creates a new row whose column contains all columns of the two rows from both tables and includes this new row in the result set. In other words, the inner join clause includes only matching rows from both tables.

If the join condition uses the equality operator (=) and the column names in both tables used for matching are the same, and you can use the **USING** clause instead :-

```
SELECT column_list FROM table_1 INNER JOIN table_2 USING (column_name);
```

Example for inner join :--

```
create table employee_data(empId int primary key auto_increment, empName varchar(20), empSalary int);
```

```
desc employee_data;
```

```
insert into employee_data(empName, empSalary) values ("Amit", 33000);
```

```
insert into employee_data(empName, empSalary) values ("Raj", 5000);
```

```
insert into employee_data(empName, empSalary) values ("Dinesh", 25000);
```

```
insert into employee_data(empName, empSalary) values ("Shiva", 17000);
```

```
insert into employee_data(empName, empSalary) values ("Sudhir", 13000);
```

```
insert into employee_data(empName, empSalary) values ("Kiran", 30000);
```

```
insert into employee_data(empName, empSalary) values ("Rakesh", 28000);
```

```
select * from employee_data;
```

```
create table city(cityId int primary key auto_increment, cityName varchar(20));
```

```
desc city;
```

```
insert into city(cityName) value("Mumbai");
insert into city(cityName) value("Delhi");
insert into city(cityName) value("Pune");
insert into city(cityName) value("Banglore");
insert into city(cityName) value("Chennai");
```

```
select * from city;
```

```
select * from employeeedata;
desc employeeedata;
```

```
alter table employeeedata add column cityId int;
```

```
alter table employeeedata add foreign key (cityId) references city(cityId);
```

```
update employeeedata set cityId=1 where empId=1;
update employeeedata set cityId=5 where empId=2;
update employeeedata set cityId=4 where empId=3;
update employeeedata set cityId=4 where empId=4;
update employeeedata set cityId=2 where empId=5;
update employeeedata set cityId=3 where empId=6;
update employeeedata set cityId=1 where empId=7;
```

```
select * from employeeedata inner join city on employeeedata.cityId = city.cityId order by empId;
```

```
select employeeedata.empId,employeeedata.empName,employeeedata.empSalary,city.cityName from employeeedata inner join city on employeeedata.cityId = city.cityId;
```

```
select employeeedata.empId,employeeedata.empName,employeeedata.empSalary,city.cityName,city.cityId from employeeedata inner join city on employeeedata.cityId = city.cityId;
```



Employee Table --



Result Grid				
	empId	empName	empSalary	cityId
▶	1	Amit	33000	1
	2	Raj	5000	5
	3	Dinesh	25000	4
	4	Shiva	17000	4
	5	Sudhir	13000	2
	6	Kiran	30000	3
	7	Rakesh	28000	1
*	NULL	NULL	NULL	NULL

City Table --

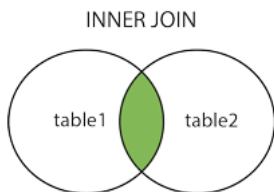
	cityId	cityName
▶	1	Mumbai
	2	Delhi
	3	Pune
	4	Banglore
	5	Chennai
*	NULL	NULL

After joining both tables --

Result Grid		 Filter Rows:			Export:	
	empId	empName	empSalary	cityId	cityId	cityName
▶	1	Amit	33000	1	1	Mumbai
	2	Raj	5000	5	5	Chennai
	3	Dinesh	25000	4	4	Banglore
	4	Shiva	17000	4	4	Banglore
	5	Sudhir	13000	2	2	Delhi
	6	Kiran	30000	3	3	Pune
	7	Rakesh	28000	1	1	Mumbai

Result Grid				Filter Rows:
	empId	empName	empSalary	cityName
▶	1	Amit	33000	Mumbai
	2	Raj	5000	Chennai
	3	Dinesh	25000	Banglore
	4	Shiva	17000	Banglore
	5	Sudhir	13000	Delhi
	6	Kiran	30000	Pune
	7	Rakesh	28000	Mumbai

Representation of Inner Join --



(2.) Left Join -- The SQL **LEFT JOIN** returns all rows from the left table, even if there are no matches in the right table.

Note -- In inner join the row which does not have matching value will not be shown. It will only be shown in left join and only the rows of left table with non-matching values will be show and not the right side ones.

```
insert into employeeedata(empName,empSalary) values("Ganesh",50000);
```

```
select * from employeeedata left join city on employeeedata.cityId = city.cityId;
```

Using Left Join --

Result Grid

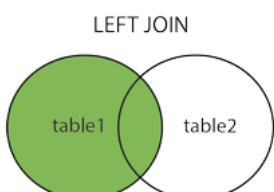
Filter Rows:

Export:

Wrap C

	empId	empName	empSalary	cityId	cityId	cityName
▶	1	Amit	33000	1	1	Mumbai
	2	Raj	5000	5	5	Chennai
	3	Dinesh	25000	4	4	Banglore
	4	Shiva	17000	4	4	Banglore
	5	Sudhir	13000	2	2	Delhi
	6	Kiran	30000	3	3	Pune
	7	Rakesh	28000	1	1	Mumbai
	8	Ganesh	50000	NULL	NULL	NULL

Representation of Left Join --



(3.) Right Join -- The RIGHT JOIN keyword Return all rows from the right table, even if there are no matches in the left table.

Note -- In inner join the row which does not have matching value will not be shown. It will only be shown in right join and only the rows of right table with non-matching values will be show and not the right side ones.

```
insert into city(cityName) value("Karnatak");  
  
select * from employeeData as e right join city as c on e.cityId = c.cityId order by empId;
```

Employee Table --

empId	empName	empSalary	cityId
1	Amit	33000	1
2	Raj	5000	5
3	Dinesh	25000	4
4	Shiva	17000	4
5	Sudhir	13000	2
6	Kiran	30000	3
7	Rakesh	28000	1
8	Ganesh	50000	NULL
NULL	NULL	NULL	NULL

City Table --

cityId	cityName
1	Mumbai
2	Delhi
3	Pune
4	Banglore
5	Chennai
6	Karnatak
NULL	NULL

Using right join --

empId	empName	empSalary	cityId	cityId	cityName
NULL	NULL	NULL	NULL	6	Karnatak
1	Amit	33000	1	1	Mumbai
2	Raj	5000	5	5	Chennai
3	Dinesh	25000	4	4	Banglore
4	Shiva	17000	4	4	Banglore
5	Sudhir	13000	2	2	Delhi
6	Kiran	30000	3	3	Pune
7	Rakesh	28000	1	1	Mumbai

Representation of RIGHT JOIN --

