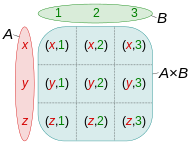
MySQL Joins

 SQL CROSS JOIN

1. A cross join is a join operation that produces the Cartesian product of two or more tables.
2. In Math, a Cartesian product is a mathematical operation that returns a product set of multiple sets.
3. The following picture illustrates the Cartesian product of A and B:



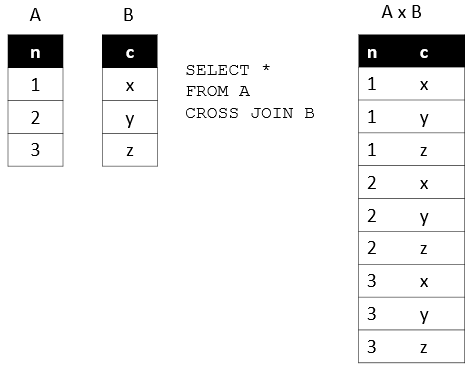
1. Similarly, in SQL, a Cartesian product of two tables A and B is a result set in which each row in the first table (A) is paired with each row in the second table (B). Suppose the A table has n rows and the B table has m rows, the result of the cross join of the A and B tables have n x m rows.
2. The following illustrates syntax of the CROSS JOIN clause:

SELECT column\_list

FROM A

CROSS JOIN B;

1. The following picture illustrates the result of the cross join between the table A and table B. In this illustration, the table A has three rows 1, 2 and 3 and the table B also has three rows x, y and z. As the result, the Cartesian product has nine rows:



Example: select employee\_id , department\_name from employees cross join departments;



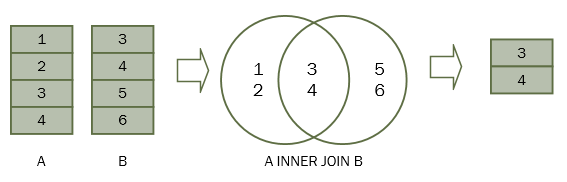
## SQL INNER JOIN

1. MySQL JOINS are used with SELECT statement. The process of linking tables is called joining.
2. It is used to retrieve data from multiple tables. It is performed whenever you need to fetch records from two or more tables.
3. Suppose, you have two tables: A and B.

Table A has four rows: (1,2,3,4) and table B has four rows: (3,4,5,6)

When table A joins with table B using the inner join, you have the result set (3,4) that is the intersection of table A and table B.

See the following picture.



1. For each row in table A, the inner join clause finds the matching rows in table B. If a row is matched, it is included in the final result set.
2. The INNER JOIN clause appears after the FROM clause. The condition to match between table A and table B is specified after the ON keyword. This condition is called join condition
3. The INNER JOIN clause can join three or more tables as long as they have relationships, typically foreign key relationships.
4. Syntax:

**SELECT** columns

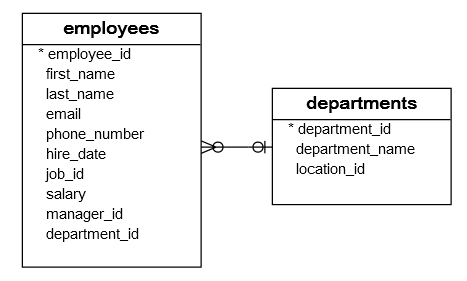
**FROM** table1

**INNER** JOIN table2

**ON** table1.**column**\_name1= table2.**column\_name2;**

### **Using SQL INNER JOIN to join two tables example**

We will use the employees and departments tables from the [sample database](https://www.sqltutorial.org/sql-sample-database/) to demonstrate how the INNER JOIN clause works.



Each employee belongs to one and only one department while each department can have more than one employee. The relationship between the departments and employees is one-to-many.

The department\_id column in the employees table is the foreign key column that links the employees to the departments table.

**To get the information of the employees who works for the department id 1,2, and 3, frame the query as:**

SELECT first\_name, last\_name, employees.department\_id, departments.department\_id, department\_name

FROM

employees

INNER JOIN

departments ON departments.department\_id = employees.department\_id

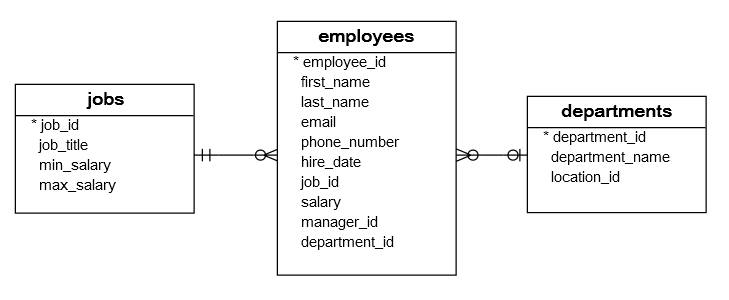
WHERE

employees.department\_id IN (1 , 2, 3);

### **SQL INNER JOIN 3 tables example**

Each employee holds one job while a job may be held by many employees. The relationship between the jobs table and the employees table is one-to-many.

The following database diagram illustrates the relationships between employees, departments and jobs tables:



Write a query that uses the inner join clauses to join 3 tables: employees, departments, and jobs to get the first name, last name, job title, and department name of employees who work in department id 1, 2, and 3.

SELECT

first\_name,

last\_name,

job\_title,

department\_name

FROM

employees e

INNER JOIN departments d ON d.department\_id = e.department\_id

INNER JOIN jobs j ON j.job\_id = e.job\_id

WHERE

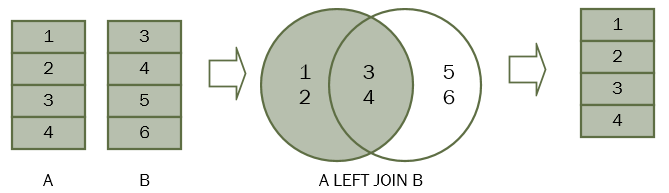
e.department\_id IN (1, 2, 3);

## SQL LEFT 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.

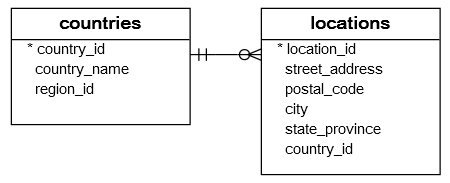
1. The inner join clause eliminates the rows that do not match with a row of the other table which somewhat means loss of data.
2. The left join, however, returns all rows from the left table whether or not there is a matching row in the right table.
3. Suppose we have two tables A and B. The table A has four rows 1, 2, 3 and 4. The table B also has four rows 3, 4, 5, 6.

When we join table A with table B, all the rows in table A (the left table) are included in the result set whether there is a matching row in the table B or not.



### **SQL LEFT JOIN two tables example**

Considering countries and locations tables.



Each location belongs to one and only one country while each country can have zero or more locations. The relationship between the countries and locations tables is one-to-many.

The country\_id column in the locations table is the foreign key that links to the country\_id column in the countries table.

To query the country names of US, UK, and China, you use the following statement.

SELECT

c.country\_name,

c.country\_id,

l.country\_id,

l.street\_address,

l.city

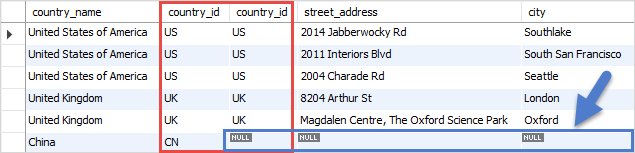
FROM

countries c

LEFT JOIN locations l ON l.country\_id = c.country\_id

WHERE

c.country\_id IN ('US', 'UK', 'CN') order by c.country\_id desc;



For each row in the countries table, the LEFT JOIN clause finds the matching rows in the locations table.

If at least one matching row found, the database engine combines the data from columns of the matching rows in both tables.

In case there is no matching row found e.g., with the country\_id CN, the row in the countries table is included in the result set and the row in the locations table is filled with NULL values.

Because non-matching rows in the right table are filled with the NULL values, you can apply the LEFT JOIN clause to miss-match rows between tables.

**For example, to find the country that does not have any locations in the locations table, you use the following query:**

**SELECT**

**country\_name**

**FROM**

**countries c**

**LEFT JOIN locations l ON l.country\_id = c.country\_id**

**WHERE**

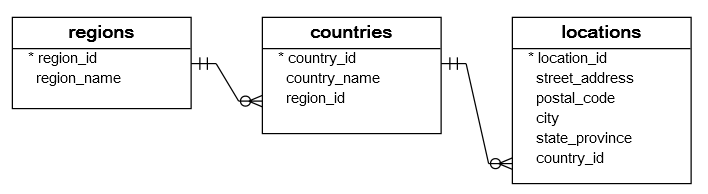
**l.location\_id IS NULL**

**ORDER BY**

**country\_name;**

### **SQL LEFT JOIN 3 tables example**

**Considering** regions, countries, and locations.



One region may have zero or many countries while each country is located in the one region. The relationship between countries and regions tables is one-to-many. The region\_id column in the countries table is the link between the countries and regions table.

The following statement demonstrates how to join 3 tables: regions, countries, and locations:

**SELECT**

**r.region\_name,**

**c.country\_name,**

**l.street\_address,**

**l.city**

**FROM**

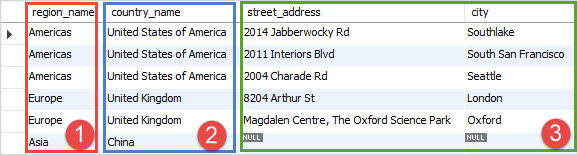
**regions r**

**LEFT JOIN countries c ON c.region\_id = r.region\_id**

**LEFT JOIN locations l ON l.country\_id = c.country\_id**

**WHERE**

**c.country\_id IN ('US', 'UK', 'CN');**



## SQL RIGHT 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 the join condition is fulfilled.

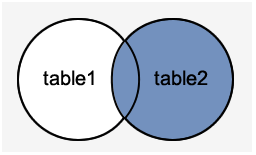
## Syntax:

**SELECT** columns

**FROM** table1

RIGHT [OUTER] JOIN table2

**ON** table1.**column** = table2.**column**;

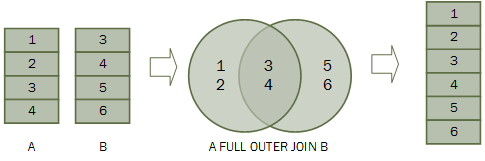


Same examples can be executed for understanding the concept of Right Join.

## SQL FULL OUTER JOIN

In theory, a full outer join is the combination of a [left join](https://www.sqltutorial.org/sql-left-join/) and a right join. The full outer join includes all rows from the joined tables whether or not the other table has the matching row.

If the rows in the joined tables do not match, the result set of the full outer join contains NULL values for every column of the table that lacks a matching row. For the matching rows, a single row that has the columns populated from the joined table is included in the result set.



**# SELF JOIN**

A self join is a join in which a table is joined with itself (which is also called Unary relationships), especially when the table has a FOREIGN KEY which references its own PRIMARY KEY. To join a table itself means that each row of the table is combined with itself and with every other row of the table.

The self join can be viewed as a join of two copies of the same table. The table is not actually copied, but SQL performs the command as though it were.

Syntax:

SELECT a.column\_name, b.column\_name...

FROM table1 a, table1 b

WHERE a.common\_field = b.common\_field;

Example:

CREATE TABLE employee(emp\_id varchar(5) NOT NULL,

emp\_name varchar(20) NULL,

dt\_of\_join date NULL,

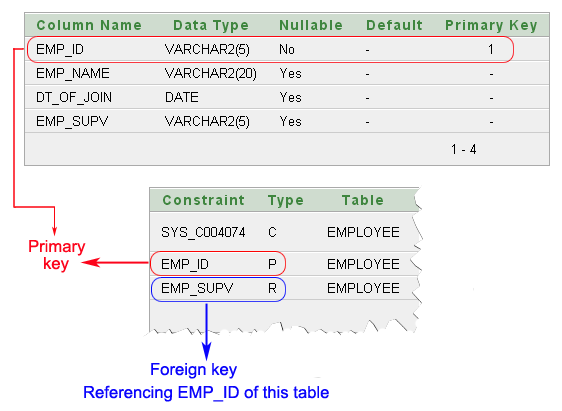
emp\_supv varchar(5) NULL,

CONSTRAINT emp\_id PRIMARY KEY(emp\_id) ,

CONSTRAINT emp\_supv FOREIGN KEY(emp\_supv)

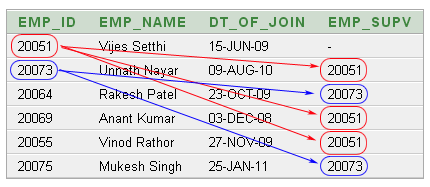
REFERENCES employee(emp\_id));

Desc employee;



**How the employees are related to themselves:**

* An employee may report to another employee (supervisor).
* An employee may supervise himself (i.e. zero) to many employees (subordinates).



The above data shows:

Unnath Nayar's supervisor is Vijes Setthi

Anant Kumar and Vinod Rathor can also report to Vijes Setthi.

Rakesh Patel and Mukesh Singh are under supervison of Unnith Nayar.

**Ques:** **To get the list of employees and their supervisor**

SELECT a.emp\_id AS "Emp\_ID",a.emp\_name AS "Employee Name",

b.emp\_id AS "Supervisor ID",b.emp\_name AS "Supervisor Name"

FROM employee a, employee b

WHERE a.emp\_supv = b.emp\_id;

