**MySQL WHERE Clause**

MySQL WHERE Clause is used with SELECT, INSERT, UPDATE and DELETE clause to filter the results. It specifies a specific position where you have to do the operation.

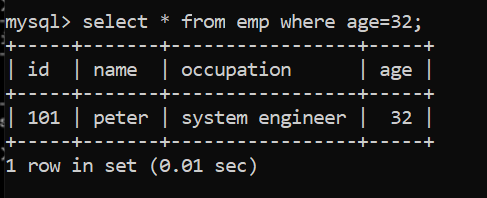
## **Syntax:**

**WHERE** conditions;

## **Parameter:**

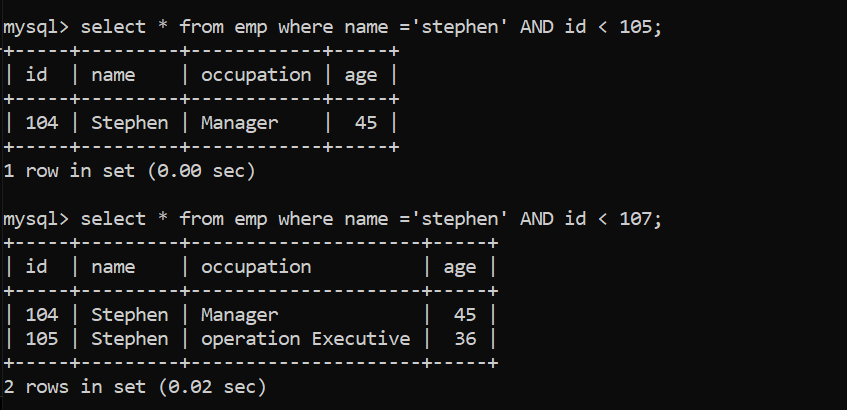
conditions: It specifies the conditions that must be fulfilled for records to be selected.

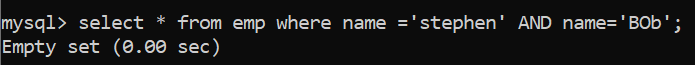
## **MySQL WHERE Clause with single condition**

******

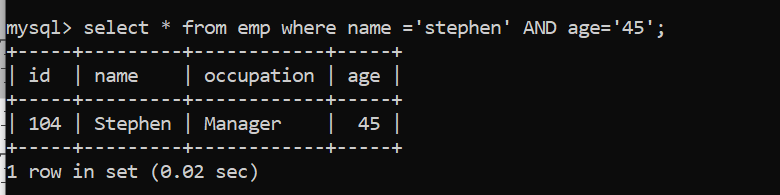
## ***MySQL WHERE Clause with AND condition***

***Note: -*** The difference between AND, OR is that **AND evaluates both conditions must be true for the overall condition to be true**. The **OR** **evaluates one condition must be true for the overall condition to be true**. In the OR result, if name is **John**, then condition will be true. If any row has the age **22**, then it will be true.

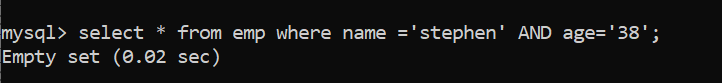
******

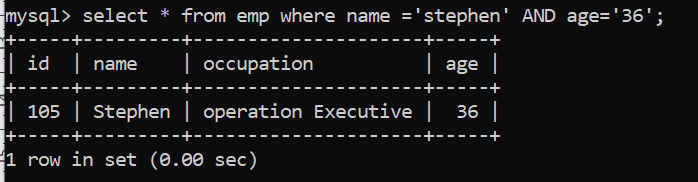


***Here the above one display the empty set because the both name fields are not in same row so condition is false.***

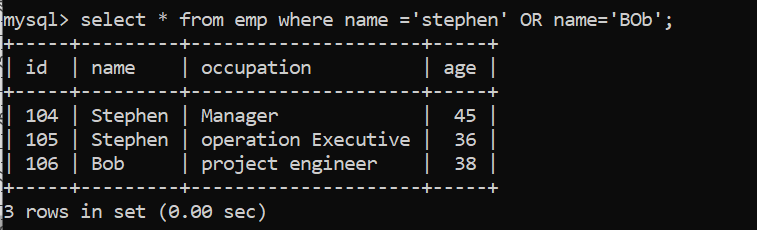
******

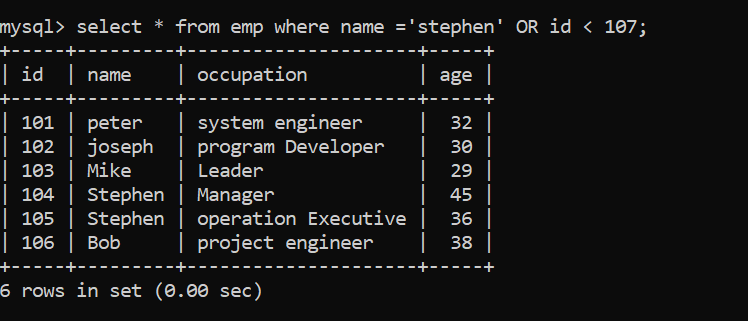
***Here the above one display the empty set because the both name fields are in same row so condition is True.***

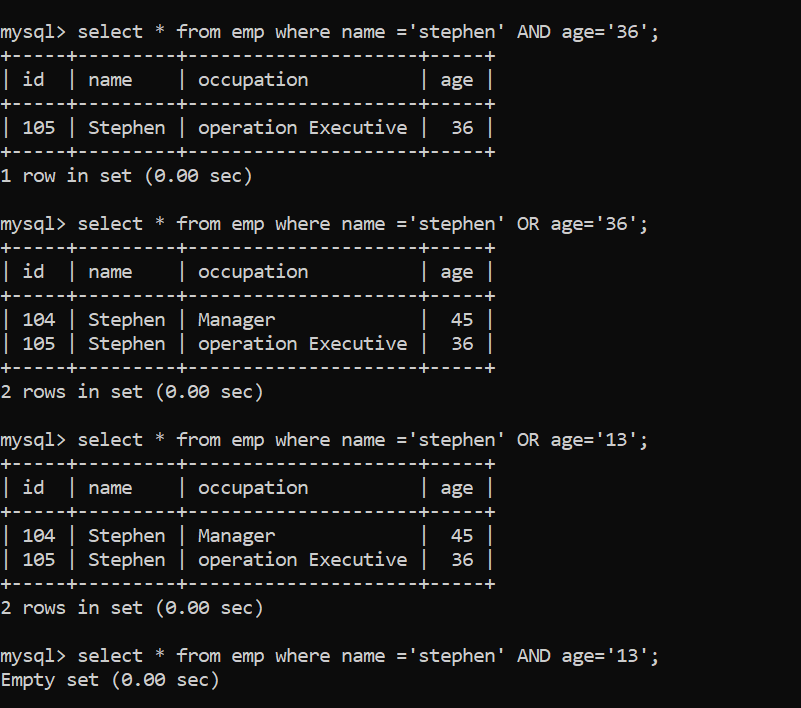
******

******

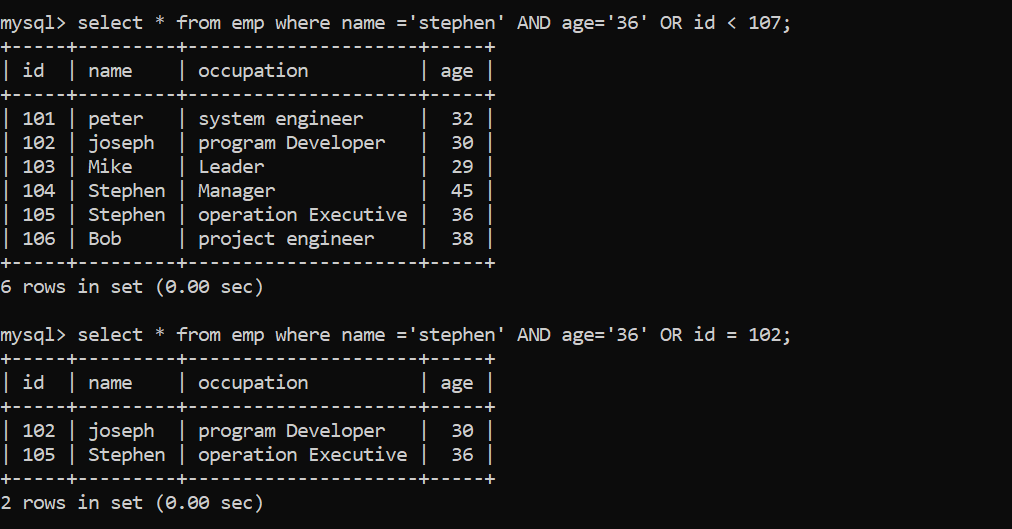
## ***WHERE Clause with OR condition***

******

******

******

**MySQL WHERE Clause with combination of AND & OR conditions**

******

# MySQL Distinct Clause

MySQL DISTINCT clause is used to remove duplicate records from the table and fetch only the unique records. The DISTINCT clause is only used with the SELECT statement.

**Syntax:**

**SELECT** **DISTINCT** expressions

**FROM** tables

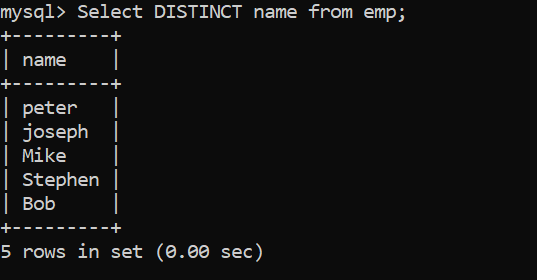
[**WHERE** conditions];

## **Parameters**

**expressions:** specify the columns or calculations that you want to retrieve.

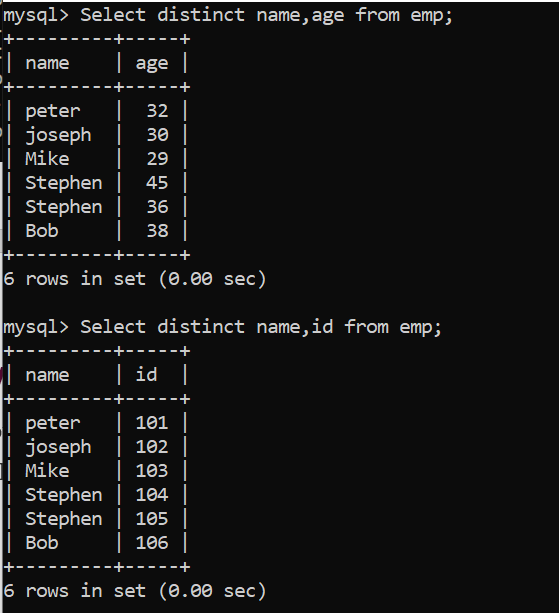
## **MySQL DISTINCT Clause with single expression**

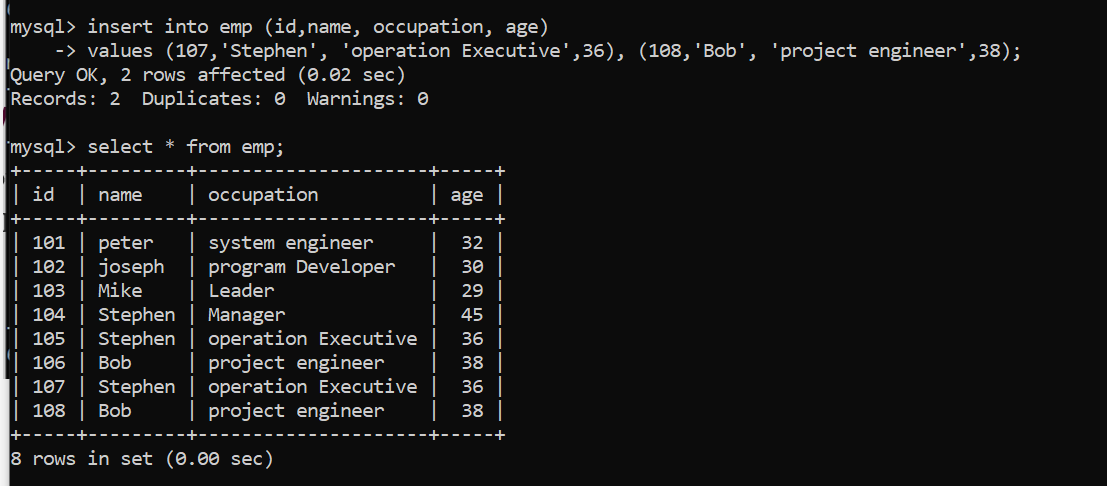
If you use a single expression then the MySQL DISTINCT clause will return a single field with unique records (no duplicate record).

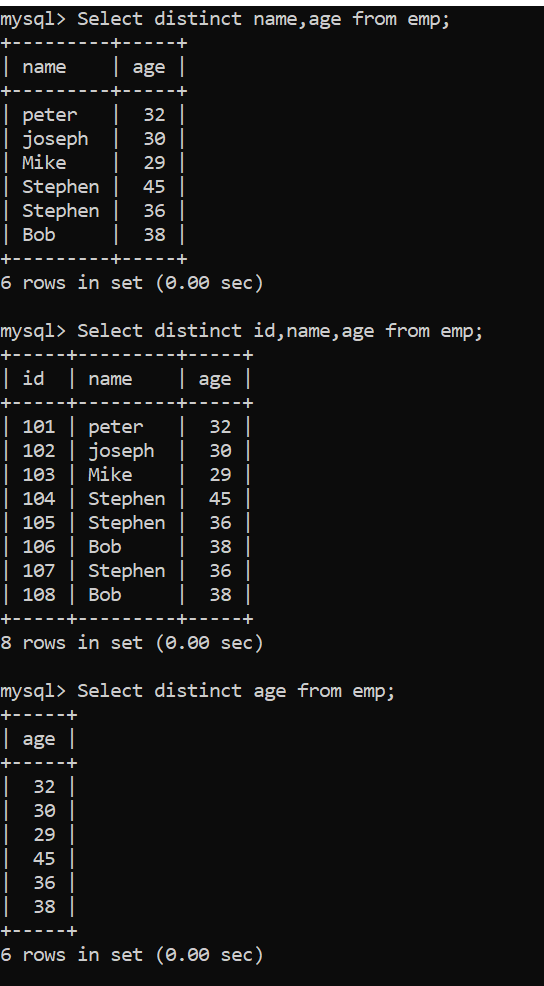
******

## **MySQL DISTINCT Clause with multiple expressions**

If you use multiple expressions with DISTINCT Clause then MySQL DISTINCT clause will remove duplicates from more than one field in your SELECT statement.





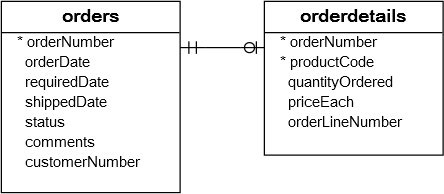
******

***MYSQL JOINS***

**Introduction to MySQL join clauses**

A relational database consists of multiple related tables linking together using common columns, which are known as **foreign key** columns. Because of this, data in each table is incomplete from the business perspective.

For example, in the sample database, we have the orders and **orderdetails** tables that are linked using the **orderNumber**column:



To get complete order’s information, you need to query data from both **orders** and **orderdetails** tables.

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.

MySQL supports the following types of joins:

***1. Inner join***

***2. Left join***

***3. Right join***

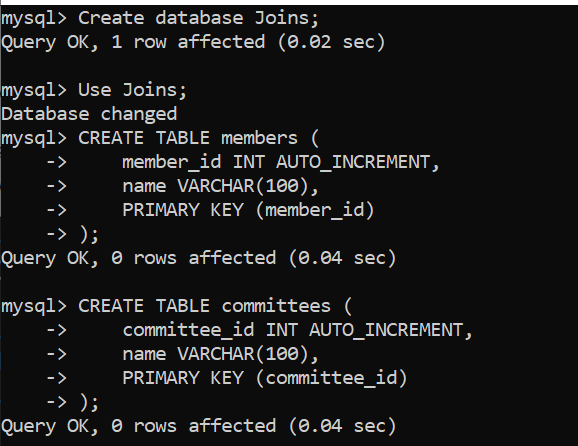
***4. Cross join***

To join tables, you use the cross join, inner join, left join, or right join clause. The join clause is used in the **SELECT** statement appeared after the FROM clause.

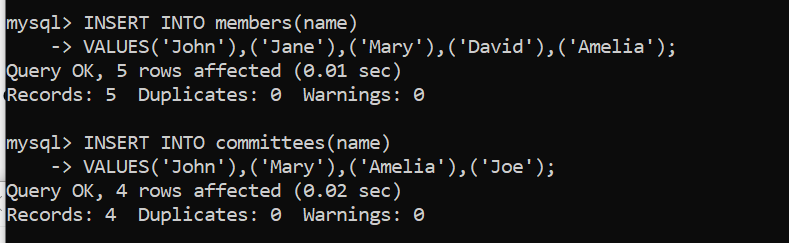
Note that MySQL hasn’t supported the FULL OUTER JOIN yet.

Setting up sample tables

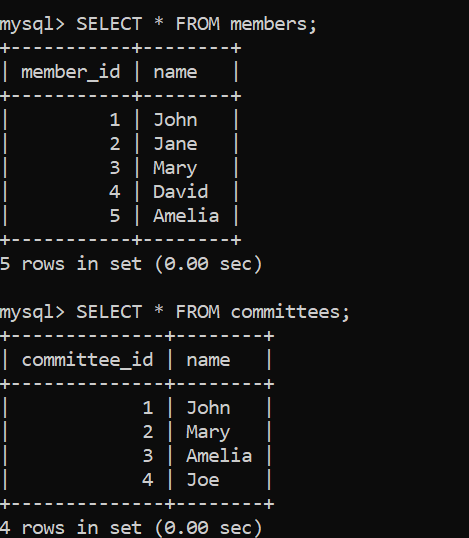
First, create two tables called members and committees:

******

Second, **insert** some rows into the tables members and committees:

******

Third, **query data** from the tables members and committees:



Some members are committee members, and some are not. On the other hand, some committee members are in the members table, some are not.

## **MySQL INNER JOIN clause**

The following shows the basic syntax of the inner join clause that joins two tables table\_1 and table\_2:

**SELECT column\_list**

**FROM table\_1**

**INNER JOIN table\_2 ON join\_condition;**

The **inner join** clause joins two tables based on a condition which is known as a join predicate.

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.

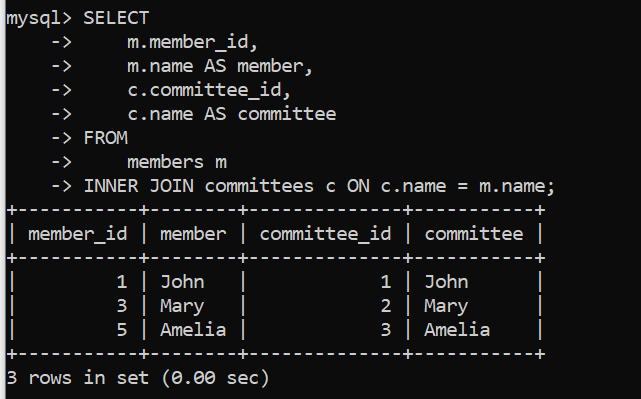
If the join condition uses the equality operator (=) and the column names in both tables used for matching are the same, and you can use the USING clause instead:

**SELECT column\_list**

**FROM table\_1**

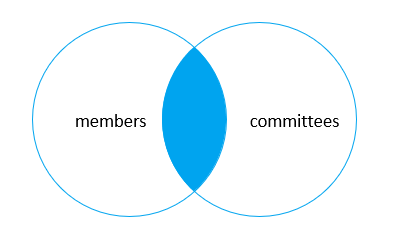
**INNER JOIN table\_2 USING (column\_name);**

The following statement uses an inner join clause to find members who are also the committee members:

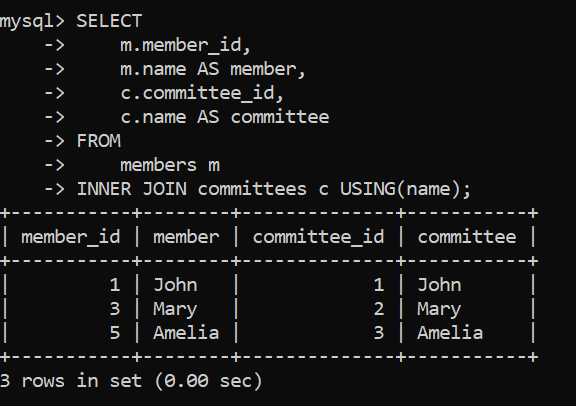
****

In this example, the inner join clause use the values in the name columns in both tables members and committees to match.

The following Venn diagram illustrates the inner join:



Because both tables use the same column to match, you can use the USING clause as shown in the following query:

****

## **MySQL LEFT JOIN clause**

Similar to an inner join, a **left join** also requires a join predicate. When joining two tables using a left join, the concepts of left and right tables are introduced.

The left join selects data starting from the left table. For each row in the left table, the left join compares with every row in the right table.

If the values in the two rows satisfy the join condition, the left join clause creates a new row whose columns contain all columns of the rows in both tables and includes this row in the result set.

If the values in the two rows are not matched, the left join clause still creates a new row whose columns contain columns of the row in the left table and NULL for columns of the row in the right table.

In other words, the left join selects all data from the left table whether there are matching rows exist in the right table or not.

In case there are no matching rows from the right table found, the left join uses NULLs for columns of the row from the right table in the result set.

Here is the basic syntax of a left join clause that joins two tables:

**SELECT column\_list**

**FROM table\_1**

**LEFT JOIN table\_2 ON join\_condition;**

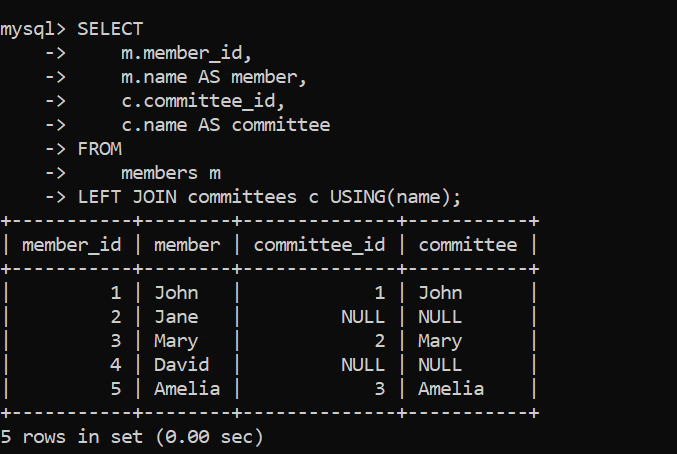
The left join also supports the USING clause if the column used for matching in both tables are the same:

**SELECT column\_list**

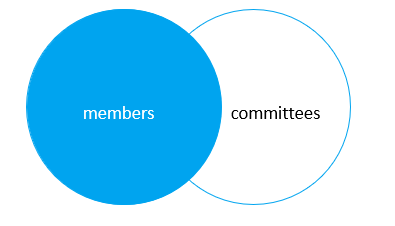
**FROM table\_1**

**LEFT JOIN table\_2 ON join\_condition;**

The following example uses a left join clause to join the members with the committees table:

****

The following Venn diagram illustrates the left join:



This statement uses the left join clause with the USING syntax:

**SELECT**

**m.member\_id,**

**m.name AS member,**

**c.committee\_id,**

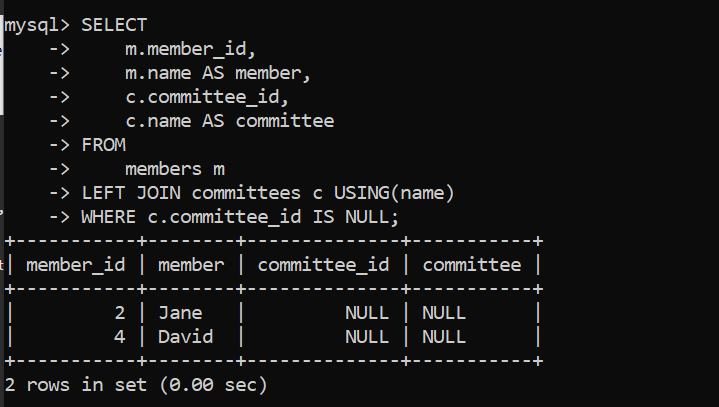
**c.name AS committee**

**FROM**

**members m**

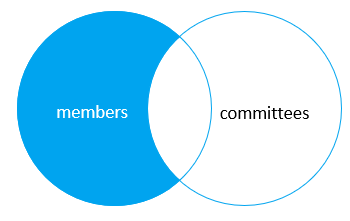
**LEFT JOIN committees c USING(name);**

To find members who are not the committee members, you add a **WHERE** clause and **IS NULL** operator as follows:

****

Generally, this query pattern can find rows in the left table that do not have corresponding rows in the right table.

This Venn diagram illustrates how to use the left join to select rows that only exist in the left table:



## **MySQL RIGHT JOIN clause**

The **right join** clause is similar to the left join clause except that the treatment of left and right tables is reversed. The right join starts selecting data from the right table instead of the left table.

The right join clause selects all rows from the right table and matches rows in the left table. If a row from the right table does not have matching rows from the left table, the column of the left table will have NULL in the final result set.

Here is the syntax of the right join:

**SELECT column\_list**

**FROM table\_1**

**RIGHT JOIN table\_2 ON join\_condition;**

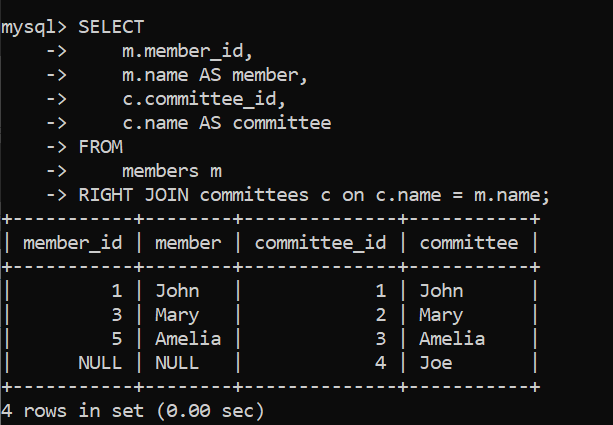
Similar to the left join clause, the right clause also supports the USING syntax:

**SELECT column\_list**

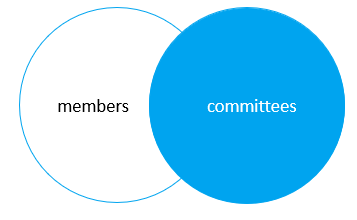
**FROM table\_1**

**RIGHT JOIN table\_2 USING (column\_name);**

To find rows in the right table that does not have corresponding rows in the left table, you also use a **WHERE** clause with the **IS NULL** operator:

****

This Venn diagram illustrates the right join:



The following statement uses the right join clause with the USING syntax:

**SELECT**

**m.member\_id,**

**m.name AS member,**

**c.committee\_id,**

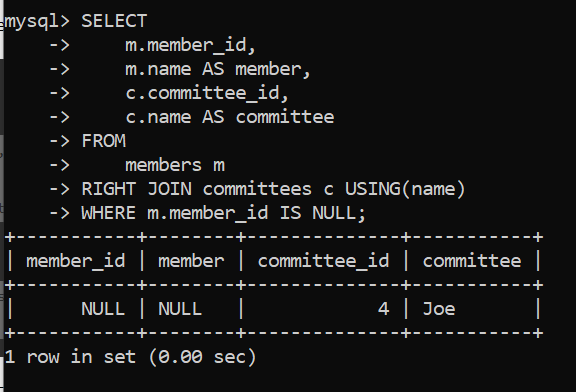
**c.name AS committee**

**FROM**

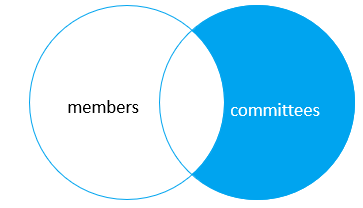
**members m**

**RIGHT JOIN committees c USING(name);**

To find the committee members who are not in the members table, you use this query:

****

This Venn diagram illustrates how to use the right join to select data that exists only in the right table:



## **MySQL CROSS JOIN clause**

Unlike the **inner join, left join**, and **right join**, the **cross join** clause does not have a join condition.

The cross join makes a Cartesian product of rows from the joined tables. The cross join combines each row from the first table with every row from the right table to make the result set.

Suppose the first table has n rows and the second table has m rows. The cross join that joins the tables will return nxm rows.

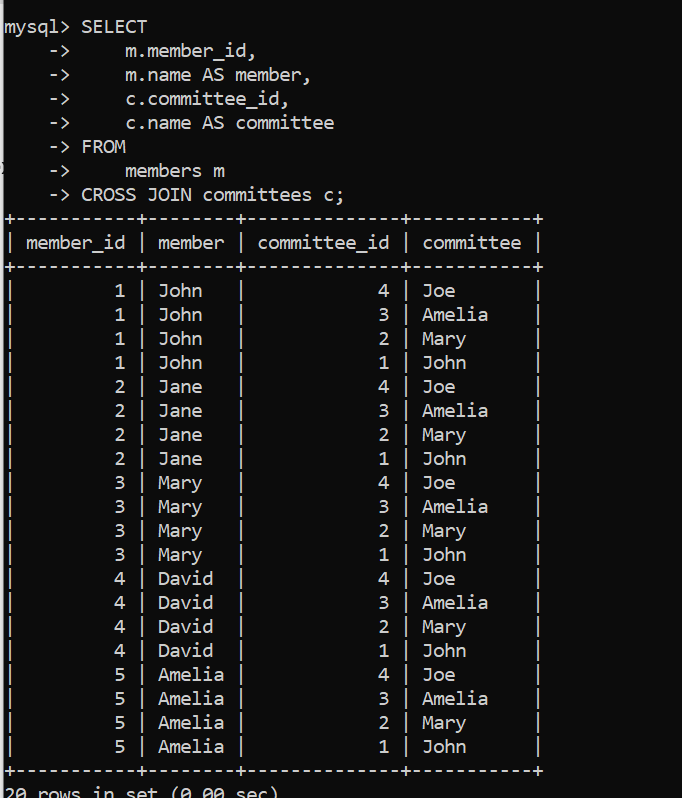
The following shows the syntax of the cross join clause:

**SELECT select\_list**

**FROM table\_1**

**CROSS JOIN table\_2;**

This example uses the cross join clause to join the members with the committees tables:

****

The cross join is useful for generating planning data. For example, you can carry the sales planning by using the cross join of customers, products, and years.

In this tutorial, you have learned various MySQL join statements, including cross join, inner join, left join, and right join, to query data from two tables.