# **MySQL Aliases**

**Summary**: in this tutorial, you will learn how to use **MySQL aliases** to assign temporary names to columns or tables in a query.

MySQL supports two kinds of aliases: column aliases and table aliases.

## **Column aliases**

In MySQL, you use column aliases to assign a temporary name to a column in the query’s result set.

For example, column names sometimes are so technical that make the query’s output very difficult to understand. To give a column a descriptive name, you can use a column alias.

The following statement illustrates how to use the column alias:

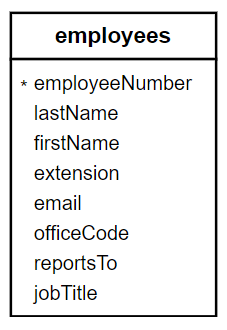


To assign a column an alias, you use the AS keyword followed by the alias. If the alias contains spaces, you must enclose it in quotes as follows:

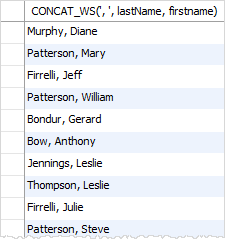


Because the AS keyword is optional, you can omit it in the statement. Note that you can also assign an expression an alias.

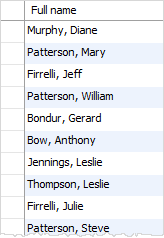
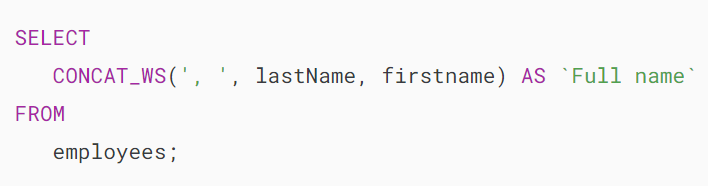
Let’s look at the employees table in the [sample database.](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/)



The following query selects the first names and last names of employees. It uses the [CONCAT\_WS()](https://www.mysqltutorial.org/mysql-string-functions/mysql-concat_ws/) function to concatenate first name and last name into full name:

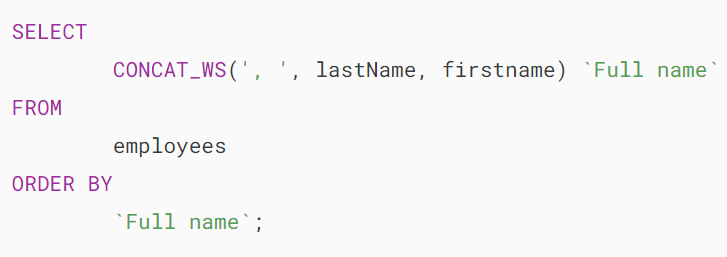


The column heading can be challenging to read. To address this issue, you can assign a column alias to the output column, as shown in the following query:

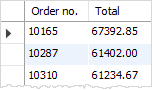
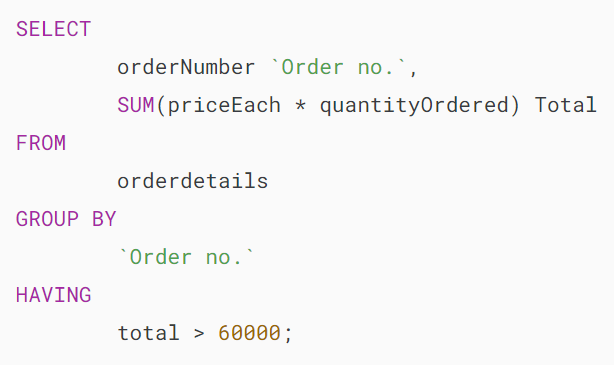


In MySQL, you can use the column alias in the [ORDER BY](https://www.mysqltutorial.org/mysql-basics/mysql-order-by/), [GROUP BY](https://www.mysqltutorial.org/mysql-basics/mysql-group-by/) and [HAVING](https://www.mysqltutorial.org/mysql-basics/mysql-having/) clauses to reference the column.

The following query uses the column alias in the ORDER BY clause to alphabetically sort the employee’s full names:



The following statement selects orders whose total amount is greater than 60000. It uses column aliases in GROUP BY and HAVING clauses.

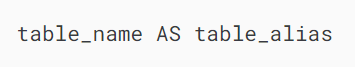


Notice that you cannot use a column alias in the [WHERE](https://www.mysqltutorial.org/mysql-basics/mysql-where/) clause. The reason is that when MySQL evaluates the WHERE clause, the values of columns specified in the SELECT clause have not been evaluated yet.

## **Table aliases**

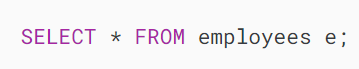
Similar to column aliases, you can assign a temporary name to a table in a query using a table alias.

Here’s the basic syntax of table alias:



In this syntax, the AS keyword is optional, so you can choose to omit it.

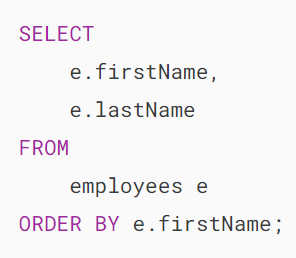
This query shows how to assign an alias e to the employees table:



Once you assign an alias to a table, you can reference a table column using the table alias like this:

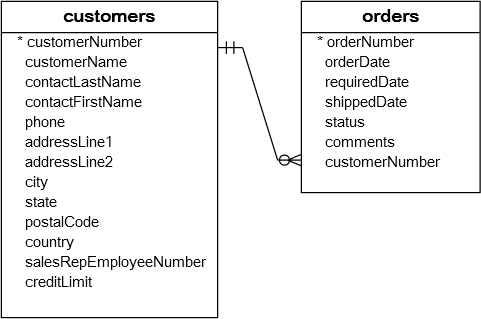


For example:



In practice, you often use table aliases in statements that contain [INNER JOIN](https://www.mysqltutorial.org/mysql-basics/mysql-inner-join/), [LEFT JOIN](https://www.mysqltutorial.org/mysql-basics/mysql-left-join/), [RIGHT JOIN](https://www.mysqltutorial.org/mysql-basics/mysql-right-join/) clauses or in [subqueries](https://www.mysqltutorial.org/mysql-basics/mysql-subquery/).

See the customers and orders tables from the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/):

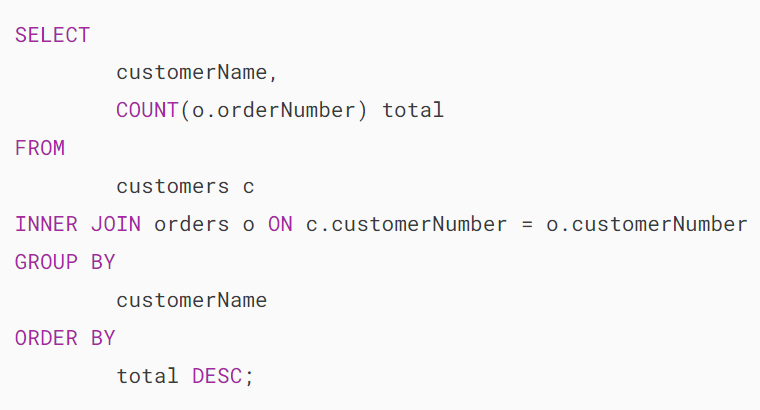


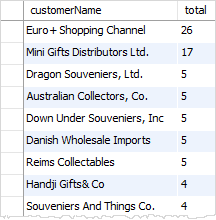
Both tables customers and orders have the same column name: customerNumber.

If you reference the customerNumber column in a query, you will get an error message:



To avoid this error, you use a table alias to qualify the customerNumber column:

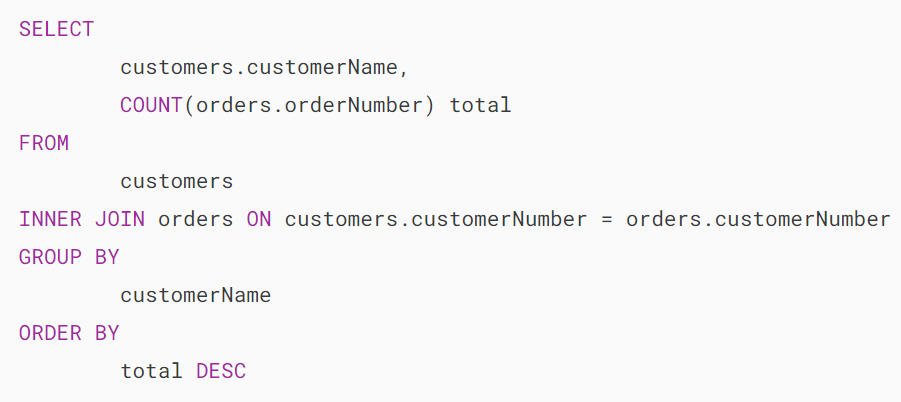
[**Try It Out**](https://www.mysqltutorial.org/tryit/query/mysql-alias/#5)



The query above selects the customer name and the number of orders from the customers and orders tables.

It uses c as a table alias for the customers table and o as a table alias for the orders table. The columns in the customers and orders tables are referred to via the table aliases.

If you do not use the alias in the query above, you have to use the table name to refer to its columns, which makes the query more verbose:



## **Summary**

* Use MySQL aliases to assign a column or a table a temporary name.
* Use a column alias to assign a temporary name to a column in a query.
* Use a table alias to assign a temporary name to a table in a query.

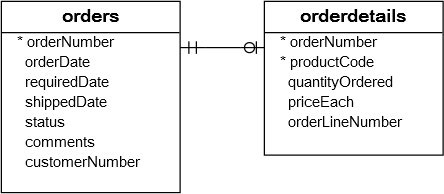
# **MySQL Join**

**Summary**: in this tutorial, you will learn various MySQL join clauses in the SELECT statement to query data from two tables.

## **Introduction to MySQL join clauses**

A relational database consists of multiple related tables linking together using common columns, which are known as [foreign key](https://www.mysqltutorial.org/www.mysqltutorial.org/mysql-foreign-key/) columns. Because of this, the data in each table is incomplete from the business perspective.

For example, in the [sample database](https://www.mysqltutorial.org/www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/), we have the orders and orderdetails tables that are linked using the orderNumber column:



To get complete order 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](https://www.mysqltutorial.org/www.mysqltutorial.org/mysql-self-join/)) or more tables based on the values of the common column between the tables.

MySQL supports the following types of joins:

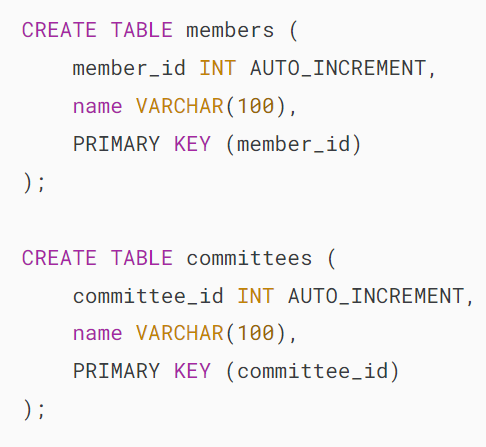
1. [Inner join](https://www.mysqltutorial.org/mysql-basics/mysql-inner-join/)
2. [Left join](https://www.mysqltutorial.org/mysql-basics/mysql-left-join/)
3. [Right join](https://www.mysqltutorial.org/mysql-basics/mysql-right-join/)
4. [Cross join](https://www.mysqltutorial.org/mysql-basics/mysql-cross-join/)

To join tables, you use the cross join, inner join, left join, or right join clause. The join clause is used in the [SELECT](https://www.mysqltutorial.org/www.mysqltutorial.org/mysql-basics/mysql-select-from/) statement appeared after the FROM clause.

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

## **Setting up sample tables**

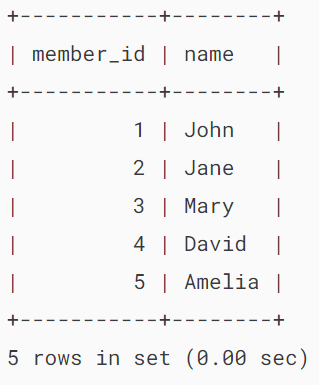
First, [create two tables](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) called members and committees:

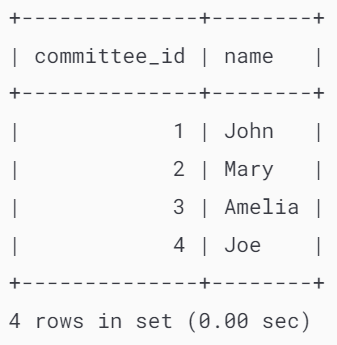
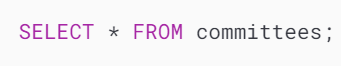


Second, [insert](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) some rows into the tables members and committees :



Third, [query data](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) from the tables members and committees:

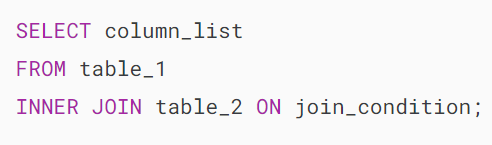




Some members are committee members, and some are not. On the other hand, some committee members are in the members table, some are not.

## **MySQL INNER JOIN clause**

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

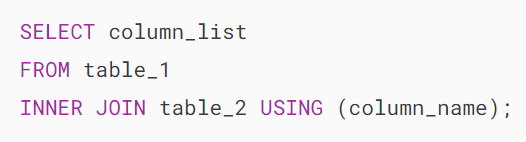


The [inner join](https://www.mysqltutorial.org/mysql-basics/mysql-inner-join/) clause joins two tables based on a condition which is known as a join predicate.

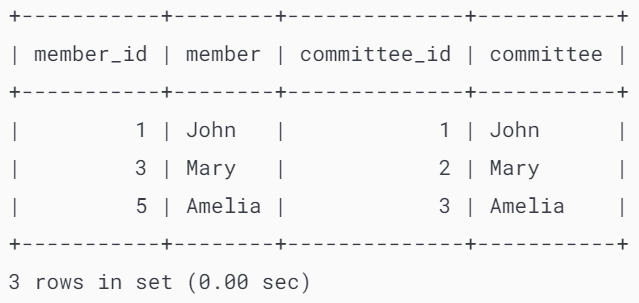
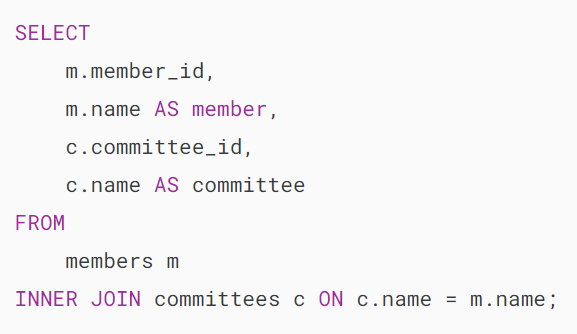
The inner join clause compares each row from the first table with every row from the second table.

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:

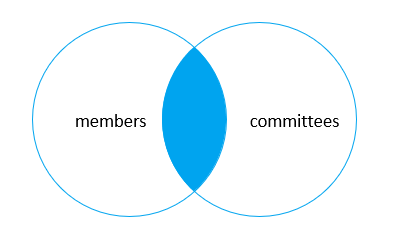


The following statement uses an inner join clause to find members who are also the committee members:



In this example, the inner join clause uses 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](https://www.mysqltutorial.org/mysql-basics/mysql-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.

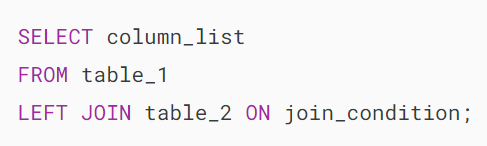
If the values in the two rows satisfy the join condition, the left join clause creates a new row whose columns contain all columns of the rows in both tables and includes this row in the result set.

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.

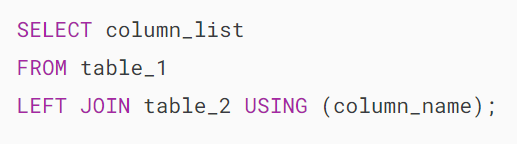
In other words, the left join selects all data from the left table whether there are matching rows exist in the right table or not.

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:

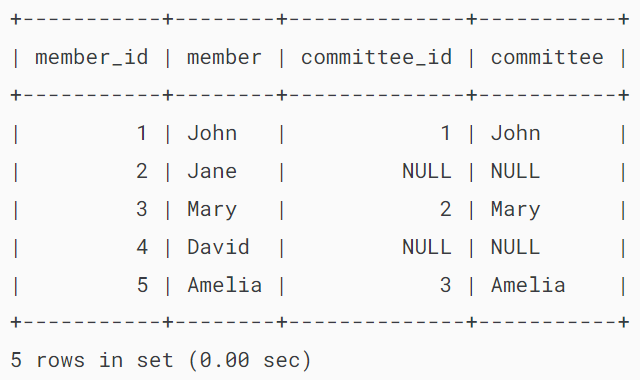


The left join also supports the USING clause if the column used for matching in both tables is the same:

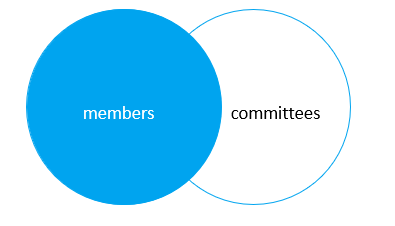


The following example uses a left join clause to join the members with the committees table:





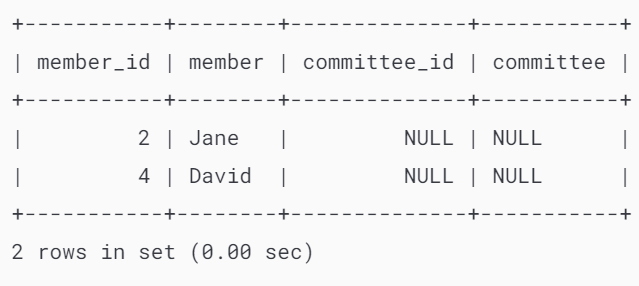
The following Venn diagram illustrates the left join:



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

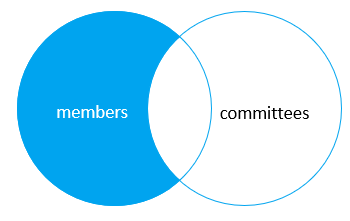


To find members who are not the committee members, you add a [WHERE](https://www.mysqltutorial.org/mysql-basics/mysql-where/) clause and [IS NULL](https://www.mysqltutorial.org/mysql-basics/mysql-is-null/) operator as follows:



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:

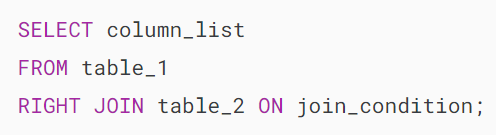


## **MySQL RIGHT JOIN clause**

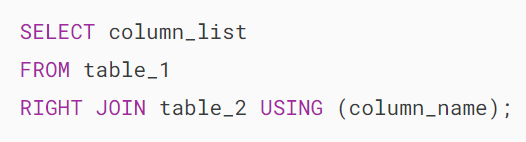
The [right join](https://www.mysqltutorial.org/mysql-basics/mysql-right-join/) clause is similar to the left join clause except that the treatment of left and right tables is reversed. The right join starts selecting data from the right table instead of the left table.

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.

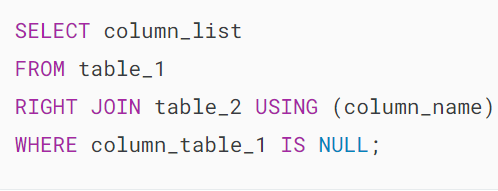
Here is the syntax of the right join:



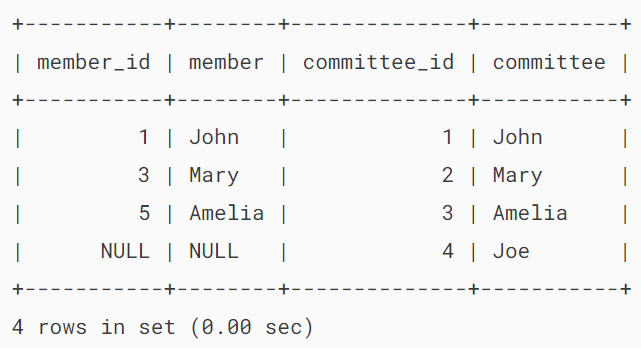
Similar to the left join clause, the right clause also supports the USING syntax:



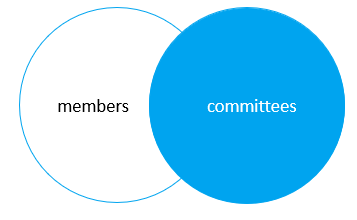
To find rows in the right table that do not have corresponding rows in the left table, you also use a [WHERE](https://www.mysqltutorial.org/mysql-basics/mysql-where/) clause with the [IS NULL](https://www.mysqltutorial.org/mysql-basics/mysql-is-null/) operator:



This statement uses the right join to join the members and committees tables:



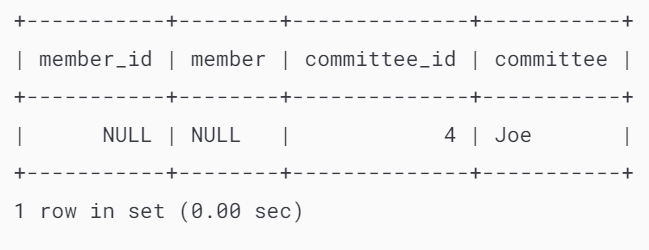
This Venn diagram illustrates the right join:



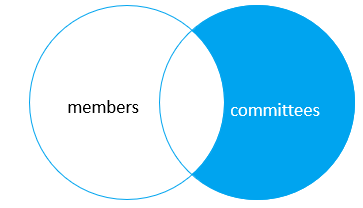
The following statement uses the right join clause with the USING syntax:



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:



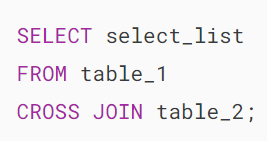
## **MySQL CROSS JOIN clause**

Unlike the [inner join](https://www.mysqltutorial.org/mysql-basics/mysql-inner-join/), [left join](https://www.mysqltutorial.org/mysql-basics/mysql-left-join/), and [right join](https://www.mysqltutorial.org/mysql-basics/mysql-right-join/), the [cross join](https://www.mysqltutorial.org/mysql-basics/mysql-cross-join/) clause does not have a join condition.

The cross join makes a [Cartesian product](https://en.wikipedia.org/wiki/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.

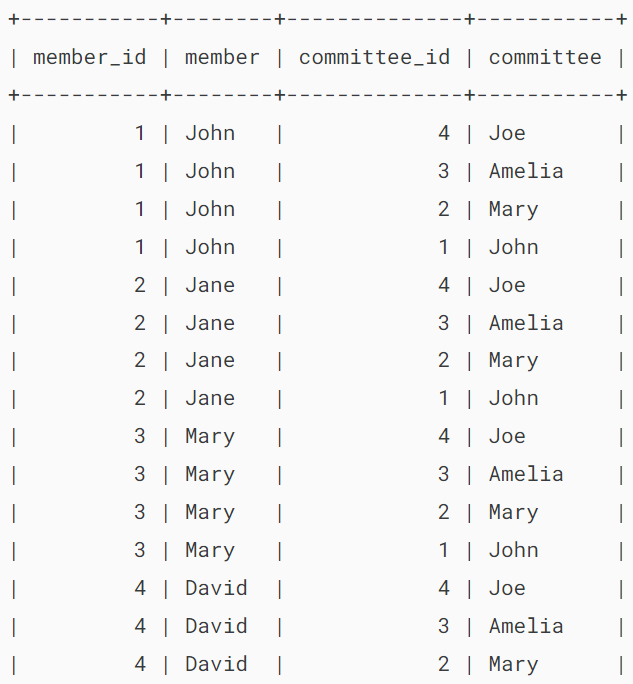
Suppose the first table has n rows and the second table has m rows. The cross-join that joins the tables will return nxm rows.

The following shows the syntax of the cross-join clause:

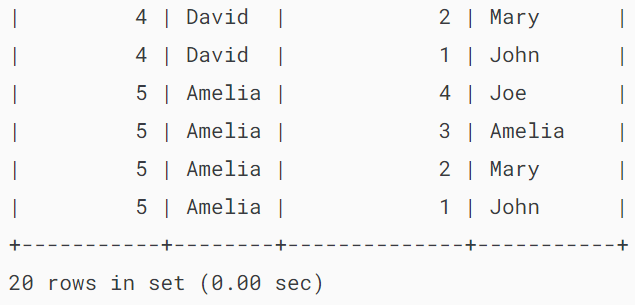


This example uses the cross join clause to join the members with the committees tables:





…



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.

# **MySQL INNER JOIN**

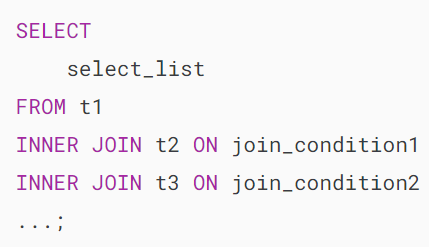
**Summary**: in this tutorial, you will learn how to use the MySQL INNER JOIN clause to select data from multiple tables based on join conditions.

## **Introduction to MySQL INNER JOIN clause**

The INNER JOIN matches each row in one table with every row in other tables and allows you to query rows that contain columns from both tables.

The INNER JOIN is an optional clause of the [SELECT](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) statement. It appears immediately after the FROM clause.

Here is the syntax of the INNER JOIN clause:



In this syntax:

* First, specify the main table that appears in the FROM clause (t1).
* Second, specify the table that will be joined with the main table, which appears in the INNER JOIN clause (t2, t3,…).
* Third, specify a join condition after the ON keyword of the INNER JOIN clause. The join condition specifies the rule for matching rows between the main table and the table that appeared in the INNER JOIN clause.

Assuming that you want to join two tables t1 and t2.

The following statement illustrates how to join two tables t1 and t2 using the INNER JOIN clause:



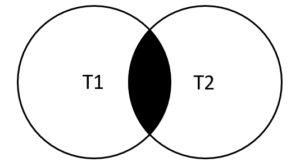
The INNER JOIN clause compares each row in the t1 table with every row in the t2 table based on the join condition.

If rows from both tables cause the join condition to evaluate to TRUE, the INNER JOIN creates a new row whose columns contain all columns of rows from the tables and includes this new row in the result set. Otherwise, the INNER JOIN just ignores the rows.

If no row between tables causes the join condition to be evaluated to TRUE, the INNER JOIN returns an empty result set.

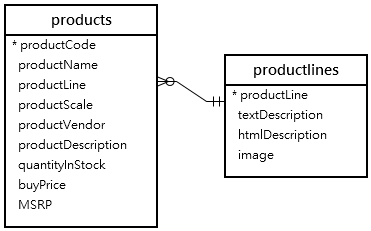
This logic is also applied when you join more than 2 tables.

The following Venn diagram illustrates how the INNER JOIN clause works:



## **MySQL INNER JOIN examples**

Let’s look at the products and productlines tables in the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/).



In this diagram, the table products has the column productLine that references the column  productline of the table productlines . The column productLine in the table products is called the [foreign key](https://www.mysqltutorial.org/mysql-basics/mysql-foreign-key/) column.

Typically, you join tables that have foreign key relationships like the  productlines and products tables.

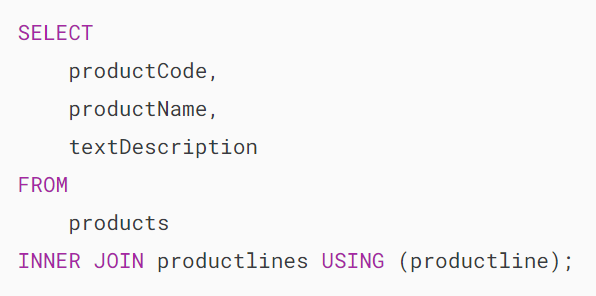
Suppose you want to get:

* The productCode and productName from the products table.
* The textDescription of product lines from the productlines table.

To do this, you need to select data from both tables by matching rows based on values in the productline column using the INNER JOIN clause as follows:

### **MySQL INNER JOIN - Products Data Example**

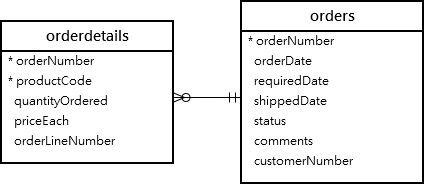
Because the joined columns of both tables have the same name  productline, you can use the USING syntax:



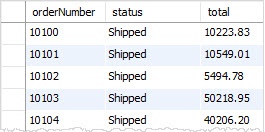
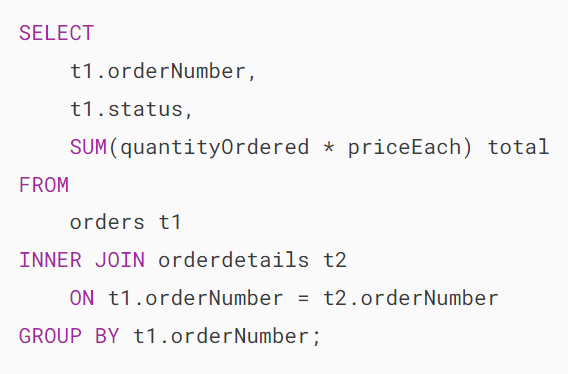
The query returns the same result set. However, the USING syntax is much shorter and cleaner.

### **MySQL INNER JOIN with GROUP BY clause example**

See the following orders and orderdetails tables:



This query returns the order number, order status, and total sales from the orders and orderdetails tables using the INNER JOIN clause with the [GROUP BY](https://www.mysqltutorial.org/mysql-basics/mysql-group-by/)clause:

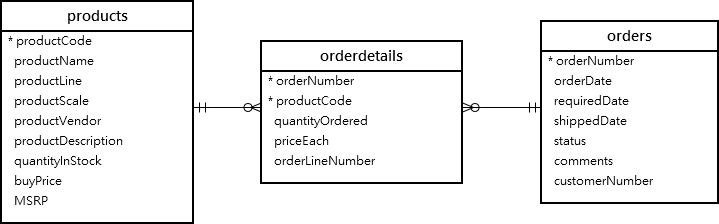


Similarly, the following query uses the INNER JOIN with the USING syntax:

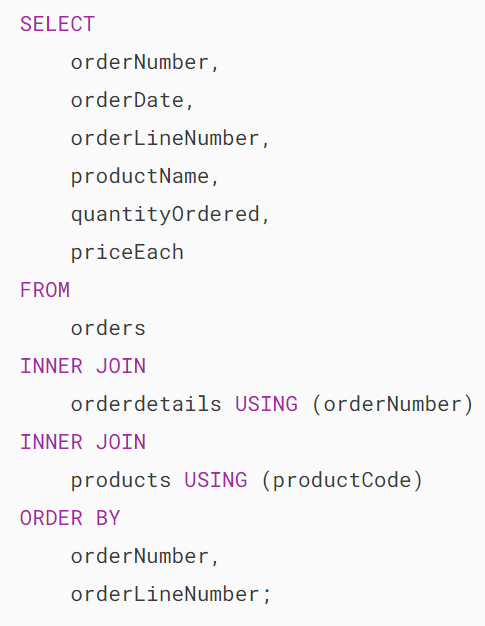
### 

### **MySQL INNER JOIN – join three tables example**

See the following products, orders and orderdetails tables:



This query uses two INNER JOIN clauses to join three tables: orders, orderdetails, and products:

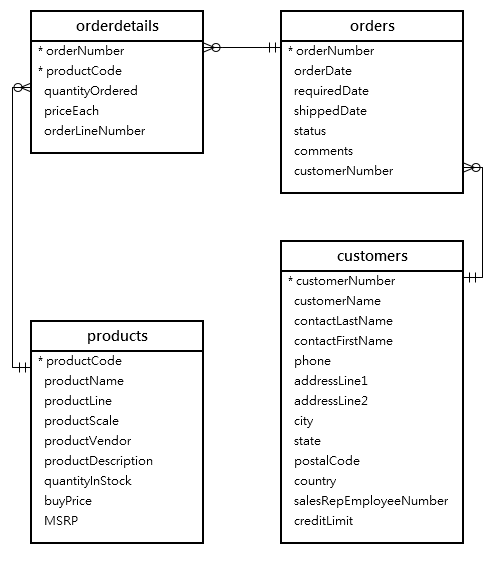


This picture shows the partial output:

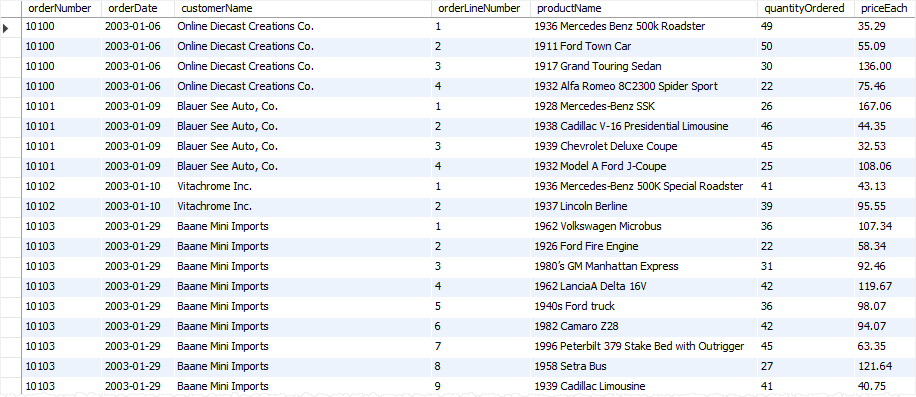
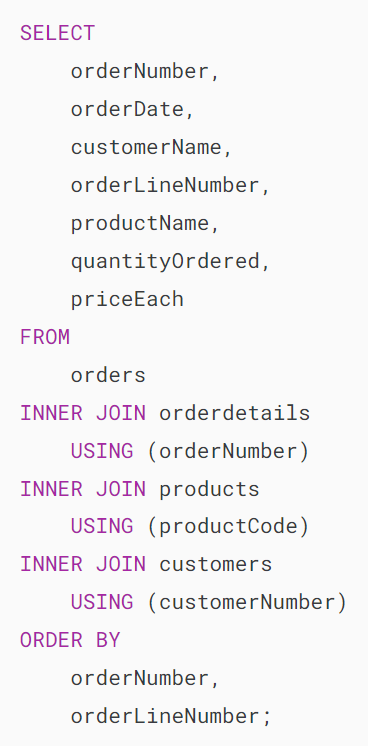


### **MySQL INNER JOIN – join four tables example**

See the following orders, orderdetails, customers and products tables:



This example uses three INNER JOIN clauses to query data from the four tables above:

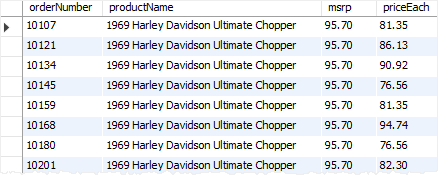
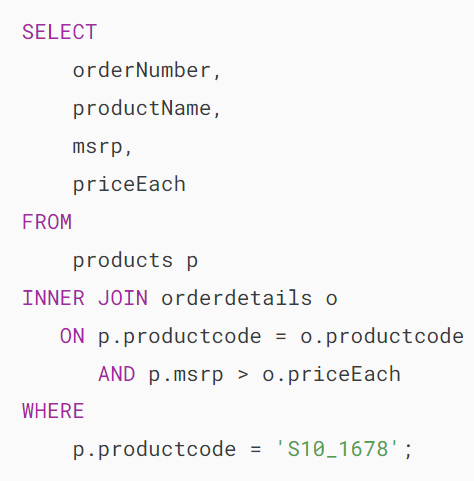


## **MySQL INNER JOIN using other operators**

So far, you have seen that the join condition used the equal operator (=) for matching rows.

In addition to the equal operator (=), you can use other operators such as greater than ( >), less than ( <), and not-equal ( <>) operator to form the join condition.

The following query uses a less-than ( <) join to find the sales price of the product whose code is S10\_1678 that is less than the manufacturer’s suggested retail price (MSRP) for that product.



In this tutorial, you have learned how to use the MySQL INNER JOIN to query data from multiple tables.

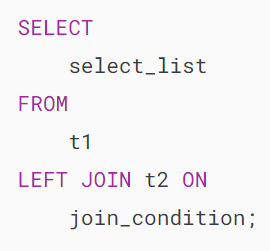
# **MySQL LEFT JOIN**

**Summary**: in this tutorial, you will learn about MySQL LEFT JOIN clause and how to use it to retrieve data from two or more related tables.

## **Introduction to MySQL LEFT JOIN clause**

The LEFT JOIN allows you to retrieve data from two or more tables. Like the [INNER JOIN](https://www.mysqltutorial.org/mysql-basics/mysql-inner-join/) clause, the LEFT JOIN is an optional clause of the [SELECT](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) statement, which appears immediately after the FROM clause.

The following statement illustrates how to use the LEFT JOIN clause to join the two tables, t1 and t2:



When you use the LEFT JOIN clause, the concepts of the left table (t1) and the right table (t2) come into play within the syntax.

The LEFT JOIN clause selects data starting from the left table (t1), matching each row from the left table (t1) with every corresponding row from the right table(t2) based on the join\_condition.

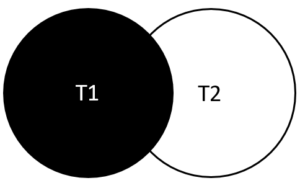
If the rows from both tables satisfy the join condition, the left join combines columns from both tables into a new row and includes this new row in the result rows.

If a row from the left table (t1) does not match with any row from the right table(t2), the left join still combines columns of rows from both tables into a new row and includes the new row in the result set. However, it fills in the columns of the row from the right table with the NULL values.

In essence, the LEFT JOIN returns all rows from the left table, irrespective of whether a matching row from the right table exists or not.

In the absence of a match, the columns of the row from the right table will be filled with NULL values.

The following Venn diagram helps you visualize how the LEFT JOIN clause works:



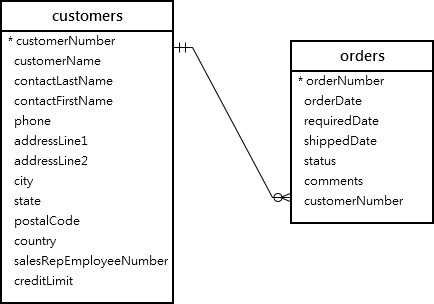
MySQL LEFT JOIN – Venn Diagram

## **MySQL LEFT JOIN clause examples**

Let’s take some examples of using the LEFT JOIN clause.

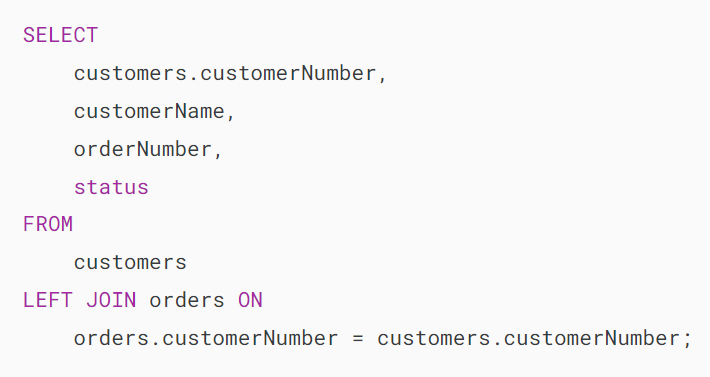
### **1) Using MySQL LEFT JOIN clause to join two tables**

See the following tables customers and orders in the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/).

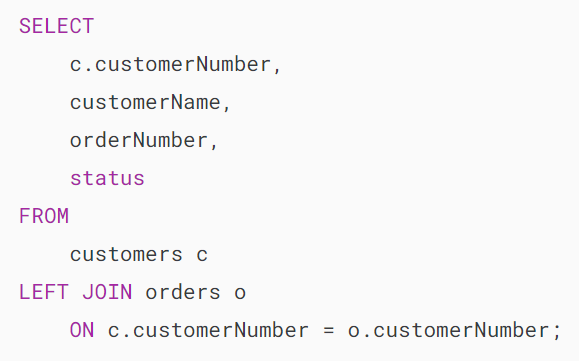


Each customer can have zero or more orders, whereas each order must belong to one customer.

The following query uses the LEFT JOIN clause to find all customers and their corresponding orders:



Alternatively, you can make the query shorter using [table aliases](https://www.mysqltutorial.org/mysql-basics/mysql-alias/):

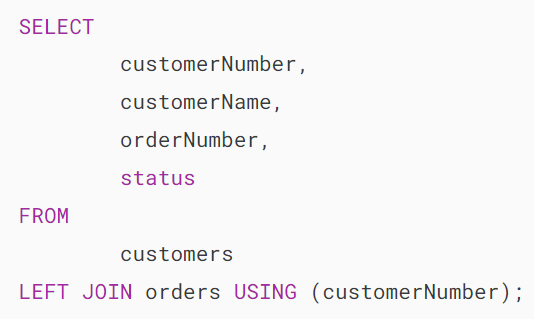




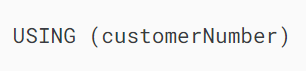
In this example:

* The customers is the left table and orders is the right table.
* The LEFT JOIN clause returns all customers including the customers who have no order. If a customer has no order, the values in the column orderNumber and status are NULL.

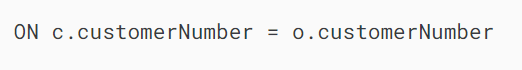
Since both the customers and orders tables share the same column name ( customerNumber) in the join condition using the equal operator, you can utilize the USING syntax as follows:



The following clauses are equivalent:



And

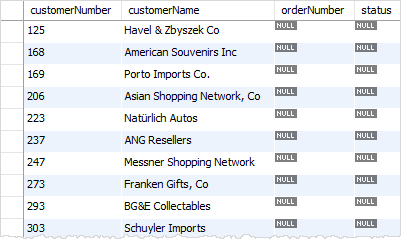
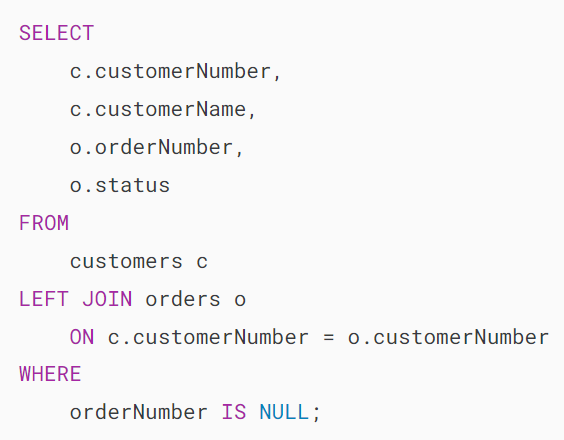


If you replace the LEFT JOIN clause by the [INNER JOIN](https://www.mysqltutorial.org/mysql-basics/mysql-inner-join/) clause, you will get the only customers who have at least one order.

### **2) Using MySQL LEFT JOIN clause to find unmatched rows**

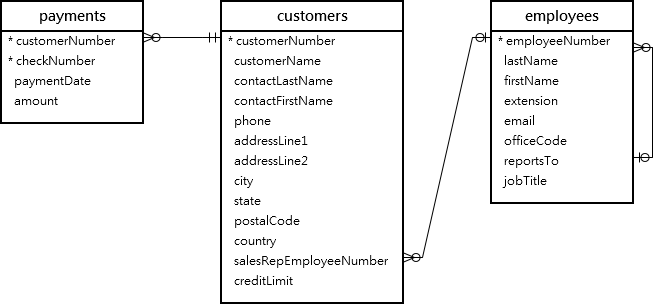
The LEFT JOIN clause is very useful when you need to identify rows in a table that doesn’t have a matching row from another table.

The following example uses the LEFT JOIN to find customers without any orders:

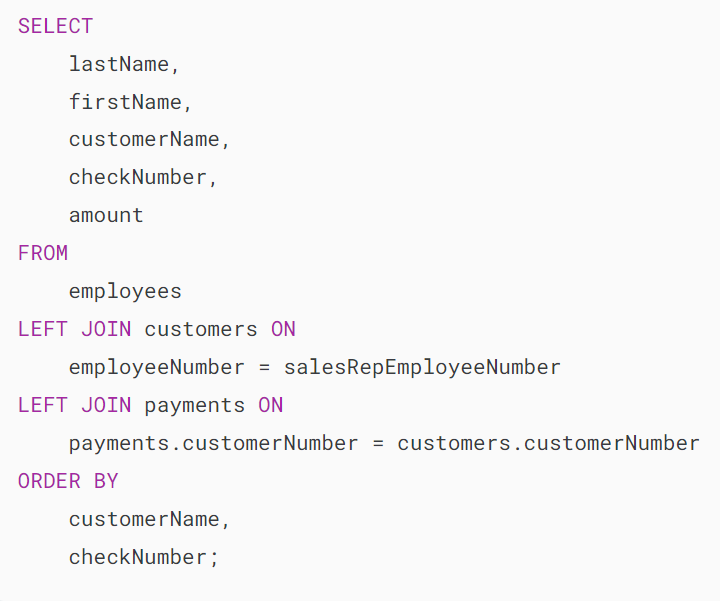


### **3) Using MySQL LEFT JOIN to join three tables**

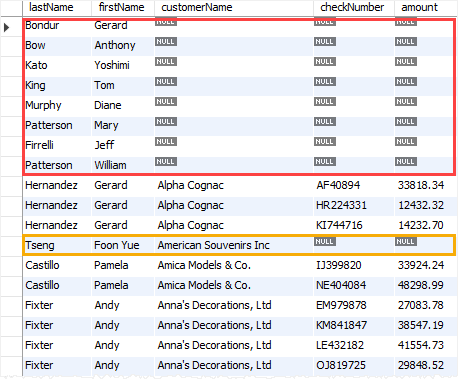
See the following three tables employees, customers, and payments:



This example uses two LEFT JOIN clauses to join the three tables: employees, customers, and payments.



This picture shows the partial output:

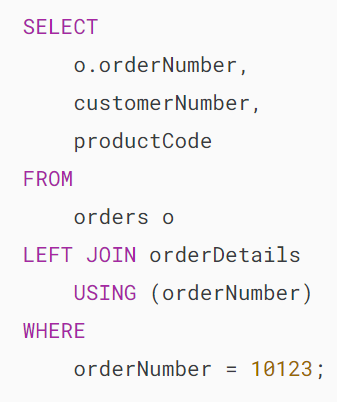


How it works.

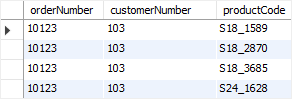
* The first LEFT JOIN returns all employees and customers who represented each employee or NULL if the employee is not in charge of any customer.
* The second LEFT JOIN retrieve payments for each customer represented by an employee or returns NULL if the customer has no payments.

## **Condition in WHERE clause vs. ON clause**

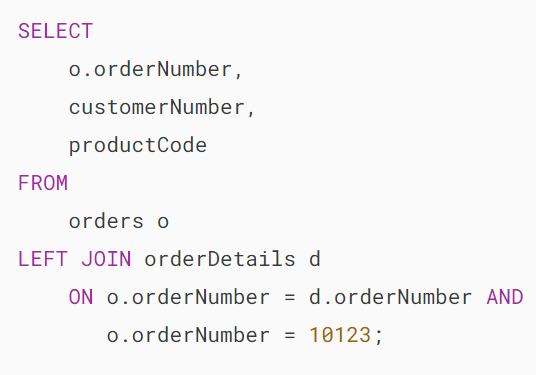
The following example uses the LEFT JOIN clause to query data from the orders and  orderDetails tables:



The query returns the order and its line items of the order number 10123.

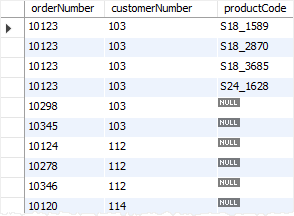


However, if you move the condition from the [WHERE](https://www.mysqltutorial.org/mysql-basics/mysql-where/) clause to the ON clause:



It will have a different meaning.

In this case, the query returns all orders; However, only the order 10123 will have associated line items as shown in the query result:



Notice that for [INNER JOIN](https://www.mysqltutorial.org/mysql-basics/mysql-inner-join/) clause, the condition in the ON clause is equivalent to the condition in the WHERE clause.

In this tutorial, you have learned how to use the MySQL LEFT JOIN clause to join data from two or more tables.

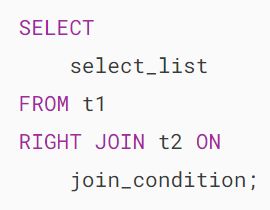
# **MySQL RIGHT JOIN**

**Summary**: in this tutorial, you will learn how to use the MySQL RIGHT JOIN to query data from two tables.

## **Introduction to MySQL RIGHT JOIN clause**

MySQL RIGHT JOIN is similar to [LEFT JOIN](https://www.mysqltutorial.org/mysql-basics/mysql-left-join/), except that the treatment of the joined tables is reversed.

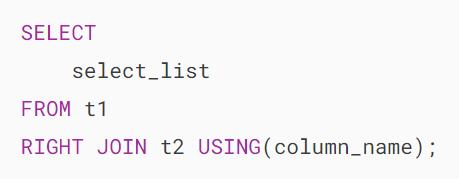
Here’s the syntax of the RIGHT JOIN of two tables t1 and t2:



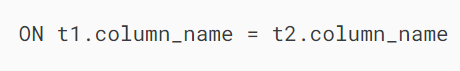
In this syntax:

* The t1 is the left table and t2 is the right table.
* The join\_condition specifies the rule for matching rows from both tables.

If the join\_condition uses the equal operator (=) and the joined columns of both tables have the same name, and you can use the USING syntax like this:



Therefore, the following join conditions are equivalent:



and



How the RIGHT JOIN works.

The RIGHT JOIN starts selecting data from the right table (t2). It matches each row from the right table with every row from the left table. If both rows cause the join condition to evaluate to TRUE, the RIGHT JOIN combines columns of these rows into a new row and includes this new row in the result set.

If a row from the right table does not have a matching row from the left table, the RIGHT JOIN combines columns of rows from the right table with NULL values for all columns of the right table into a new row and include this row in the result set.

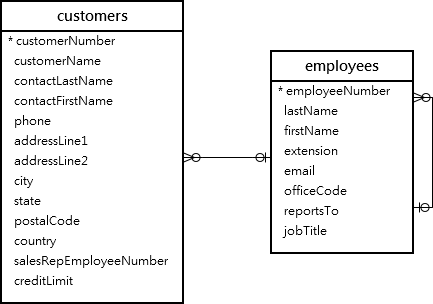
In other words, the RIGHT JOIN returns all rows from the right table regardless of having matching rows from the left table or not.

It’s important to emphasize that RIGHT JOIN and LEFT JOIN clauses are functionally equivalent, and they can replace each other as long as the table order is reversed.

Notice that the RIGHT OUTER JOIN is a synonym for RIGHT JOIN. Therefore, you can use them interchangeably.

## **MySQL RIGHT JOIN clause examples**

We’ll use the tables employees and customers from the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/) for the demonstration:



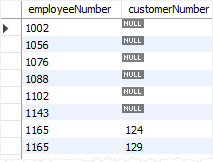
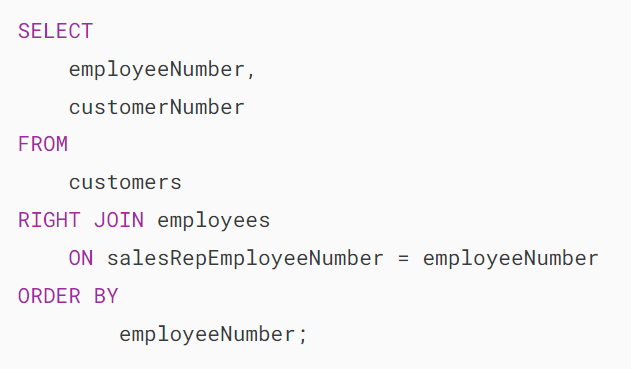
The column salesRepEmployeeNumber in the table customers links to the column employeeNumber in the employees table.

A sales representative, or an employee, may be in charge of zero or more customers. And each customer is taken care of by zero or one sales representative.

If the value in the column salesRepEmployeeNumber is NULL, which means the customer does not have any sales representative.

### **1) Simple MySQL RIGHT JOIN example**

This statement uses the RIGHT JOIN clause join the table customers with the table employees.

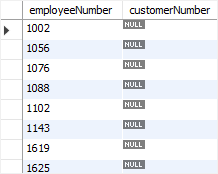
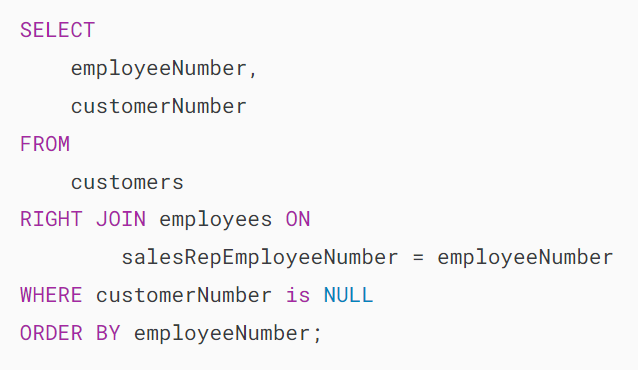


In this example:

* The RIGHT JOIN returns all rows from the table employees whether rows in the table employees have matching values in the column salesRepEmployeeNumber of the table customers.
* If a row from the table employees has no matching row from the table customers , the RIGHT JOIN uses NULL for the customerNumber column.

### **2) Using MySQL RIGHT JOIN to find unmatching rows**

The following statement uses the RIGHT JOIN clause to find employees who are not in charge of any customers:



## **Summary**

* MySQL RIGHT JOIN allows you to query data from two or more related tables.
* The RIGHT JOIN starts selecting rows from the right table. It always returns rows from the right table whether or not there are matching rows in the left table.
* The RIGHT OUTER JOIN is the synonym of the RIGHT JOIN.

# **MySQL Self Join**

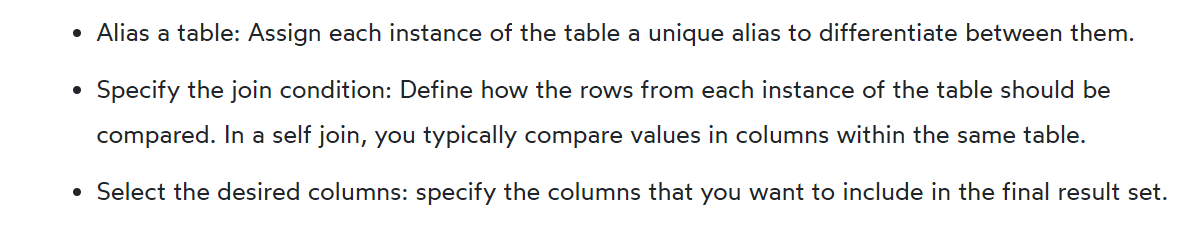
**Summary:**in this tutorial, you will learn how to use **MySQL self join** which joins a table to itself using the inner join or left join.

## **Introduction to MySQL Self Join**

A self join allows you to join a table to itself. Since MySQL does not have specific self join syntax, you need to perform a self join via a regular join such as [left join](https://www.mysqltutorial.org/mysql-basics/mysql-left-join/) or [inner join](https://www.mysqltutorial.org/mysql-basics/mysql-inner-join/).

Since you reference the same table within a single query, you need to use [table aliases](https://www.mysqltutorial.org/mysql-basics/mysql-alias/) to assign the table a temporary name when you reference it for the second time.

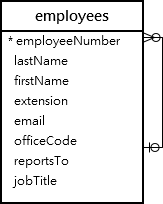
To perform a self join, you follow these steps:



In practice, you use a self join to query hierarchical data such as displaying organization structure or comparing rows within the same table.

## **MySQL self join examples**

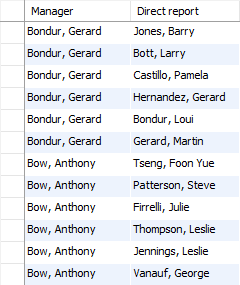
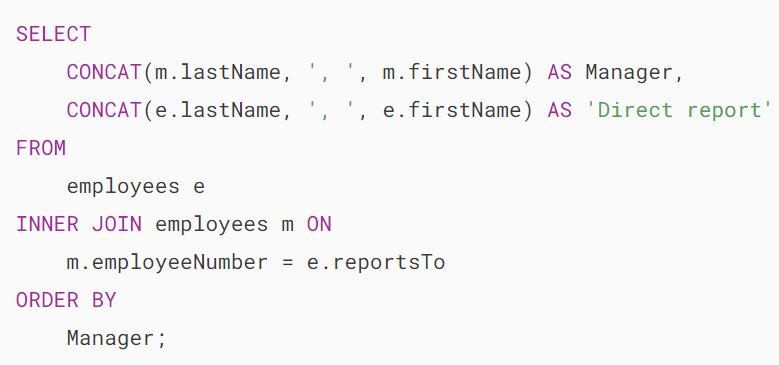
We’ll use the employees table from the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/).



The table employees stores not only employees’ data but also the organization’s data. It uses the reportsto column to determine the manager id of an employee.

### **1) Performing a self join using an inner join**

To obtain the entire organization structure, you can perform a self join on the employees table using the employeeNumber and reportsTo columns:

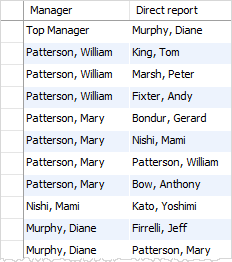


The output displays only the employees who have a manager. However, you don’t see the President because his name is filtered out due to the INNER JOIN clause.

### **2) Performing a self join using a left join**

The President is the employee who does not have any manager or value in the reportsTo column is NULL .

The following statement uses the LEFT JOIN clause instead of INNER JOIN to include the President:



### **3) Using a self join to compare successive rows within the same table**

By using the MySQL self join, you can display a list of customers who are located in the same city by joining the customers table to itself.



In this example, the table customers is joined to itself using the following join conditions:

* c1.city = c2.city  makes sure that both customers have the same city.
* c1.customerName > c2.customerName ensures that the result does not include the same customer.

## **Summary**

* The MySQL self-join is a technique that joins a table to itself.
* Use table aliases and inner join or left join to perform a self join in MySQL.

# **MySQL CROSS JOIN**

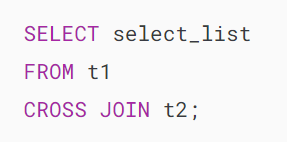
**Summary**: in this tutorial, you will learn about the MySQL CROSS JOIN clause and how to use it more effectively.

## **Introduction to MySQL CROSS JOIN clause**

Suppose you join two tables using the CROSS JOIN clause. The result set will include all rows from both tables, where each row is the combination of the row in the first table with the row in the second table. In general, if each table has n and m rows respectively, the result set will have nxm rows.

In other words, the CROSS JOIN clause returns a **Cartesian product** of rows from the joined tables.

The following illustrates the syntax of the CROSS JOIN clause that joins two tables t1 and t2:



Note that different from the [INNER JOIN](https://www.mysqltutorial.org/mysql-basics/mysql-inner-join/), [LEFT JOIN](https://www.mysqltutorial.org/mysql-basics/mysql-left-join/) , and [RIGHT JOIN](https://www.mysqltutorial.org/mysql-basics/mysql-right-join/) clauses, the CROSS JOIN clause does not have a join predicate. In other words, it does not have the ON or USING clause.

If you add a WHERE clause, in case table t1 and t2 has a relationship, the CROSS JOIN works like the INNER JOIN clause as shown in the following query:

## 

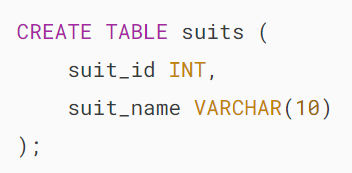
## **MySQL CROSS JOIN clause examples**

Let’s take some examples to understand how the cross join works.

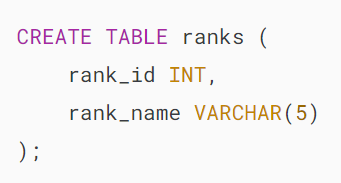
### **1) Simple cross join example**

We’ll use a cross join to create a deck of 52 playing cards.

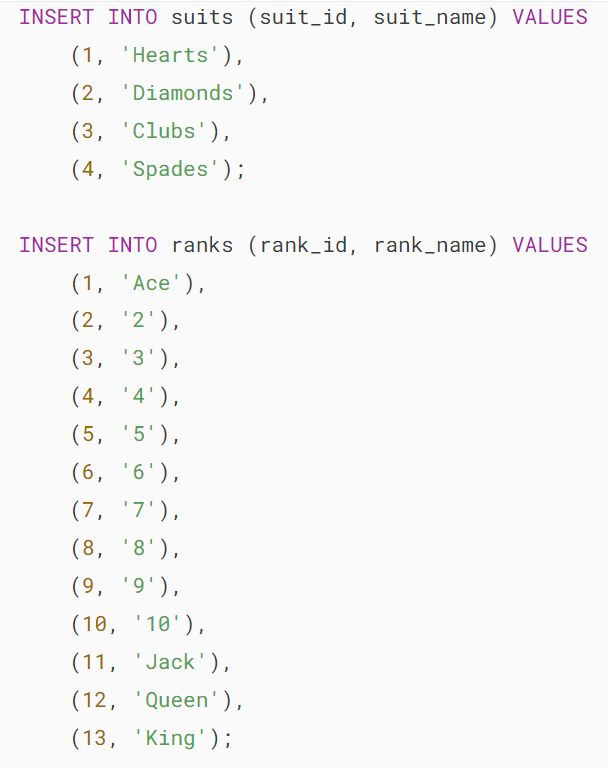
First, create a table that stores suits:



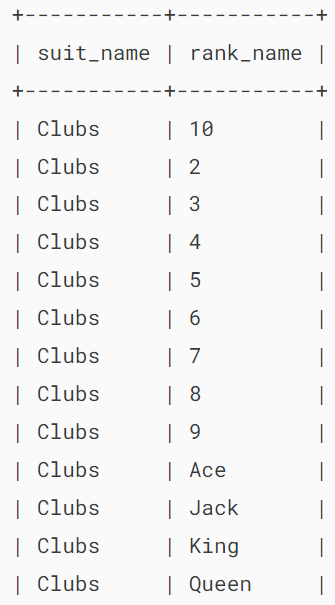
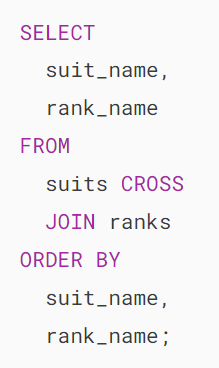
Second, create a table to store ranks:



Third, insert data into the suits and ranks table:



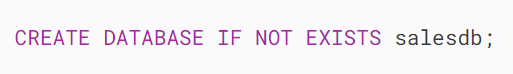
Finally, use a cross to combine the suits and ranks to create a deck of 52 playing cards:



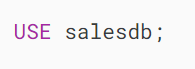
In this query, we use a cross join to combine each suit from the suits table with each rank from the ranks table, resulting in a Cartesian product that pairs every suit with every rank.

### **2) A complex cross join example**

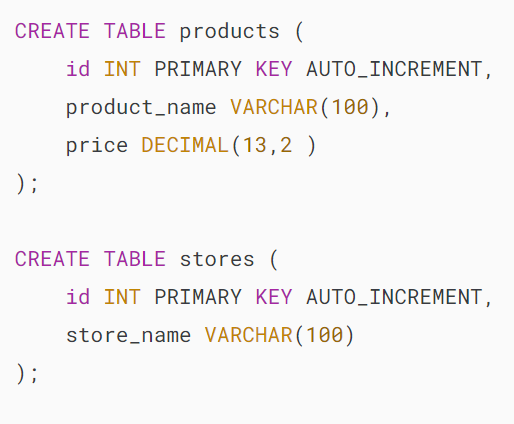
First, [create a new database](https://www.mysqltutorial.org/mysql-basics/mysql-create-database/) salesdb:

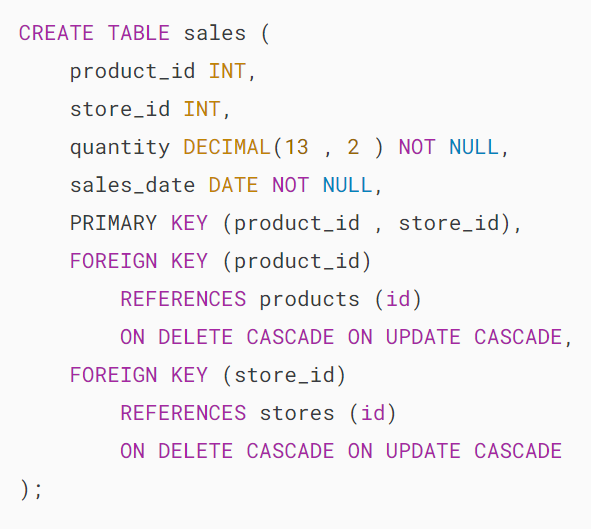


Second, switch the current data to the new database salesdb:



Third, [create new tables](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) in the salesdb database:



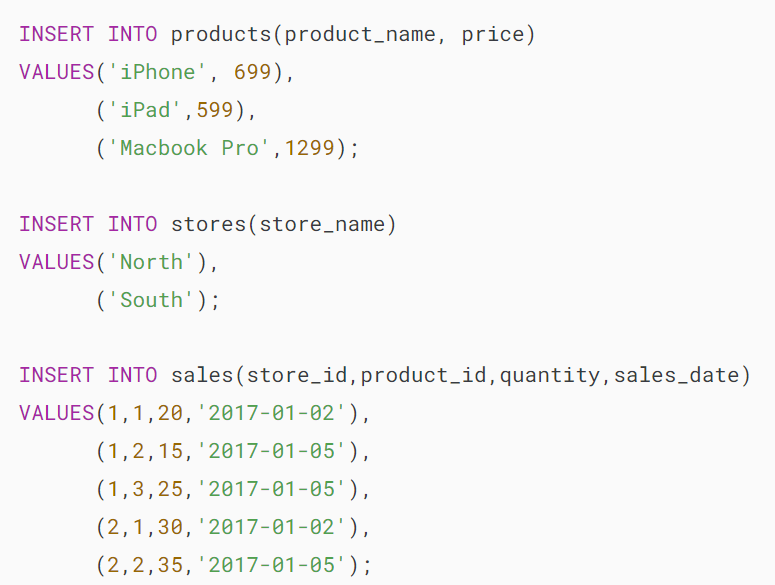


Here are the descriptions of the tables:

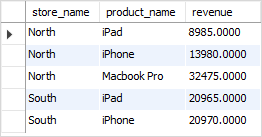
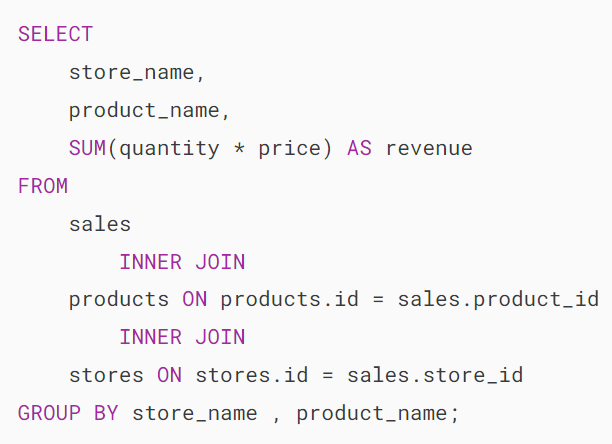
* The table products contains the product master data that includes product id, product name, and sales price.
* The table stores contains the stores where the products are sold.
* The table sales contains the products that are sold in a particular store by quantity and date.

Fourth, insert data into the three tables.

Suppose that we have three products iPhone, iPad and Macbook Pro which are sold in two stores North and South.



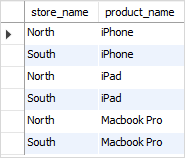
Fifth, the following statement returns the total sales for each product in each store:



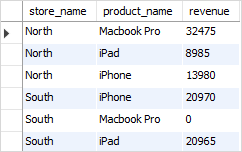
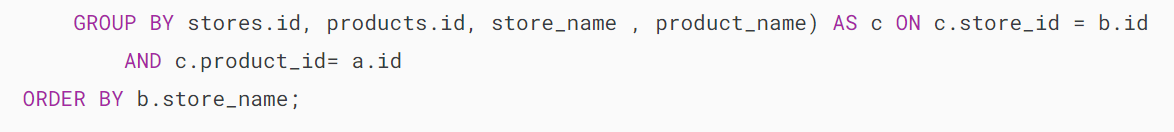
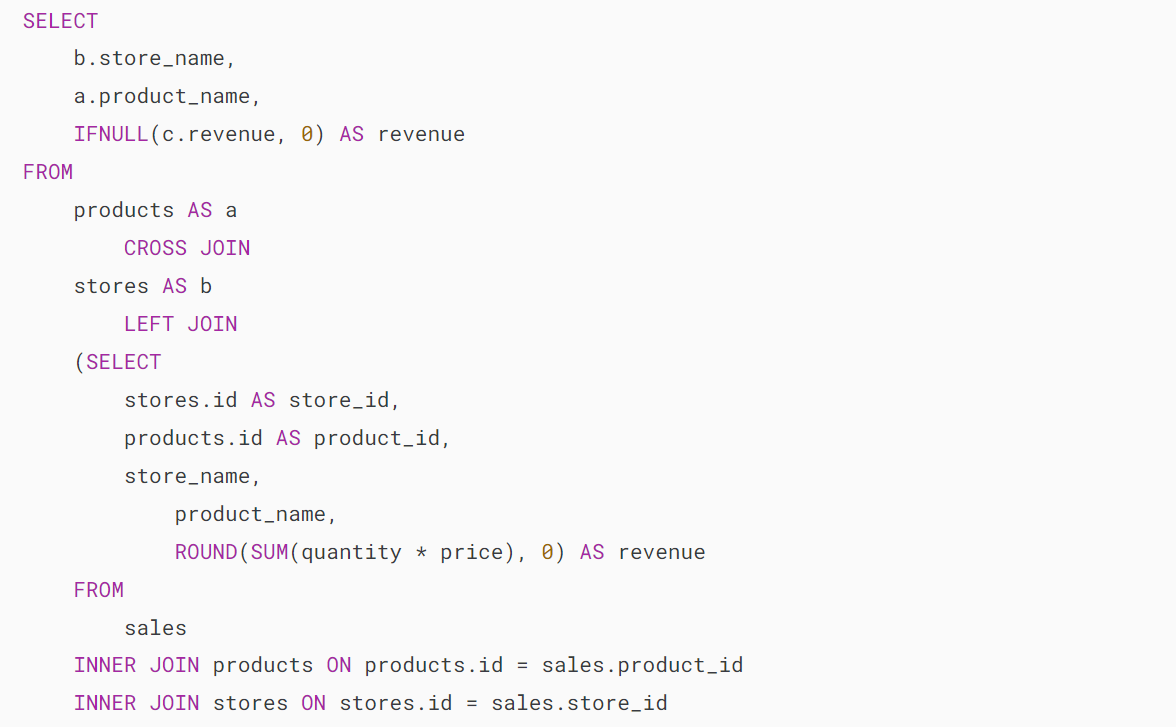
Now, what if you wish to determine which store had no sales of a particular product? The previously mentioned statement is unable to address this query.

To solve the problem, you can use the CROSS JOIN clause.

Sixth, use the CROSS JOIN clause to get the combination of all stores and products:



Next, join the result of the query above with a query that returns the total of sales by store and product:



Note that the query used the [IFNULL](https://www.mysqltutorial.org/mysql-control-flow-functions/mysql-ifnull/) function to return 0 if the revenue is [NULL](https://www.mysqltutorial.org/mysql-basics/mysql-null/) (in case the store had no sales).

By using the CROSS JOIN clause this way, you can answer a wide range of questions e.g., find the sales revenue by salesman, month even if the salesman has no sales in a particular month.

## **Summary**

* A cross join combines each row from a table with each row from another table, resulting in a Cartesian product.
* Use the CROSS JOIN clause to perform a cross join.