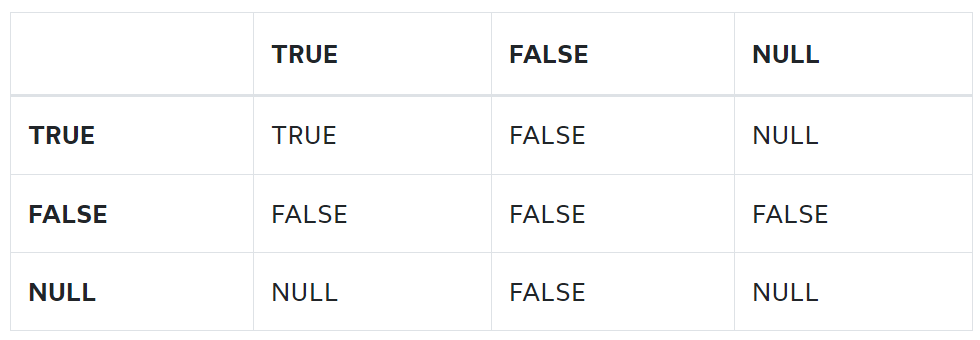
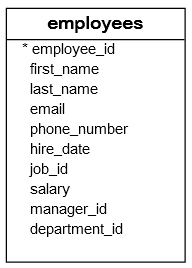
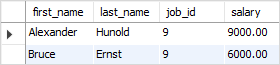
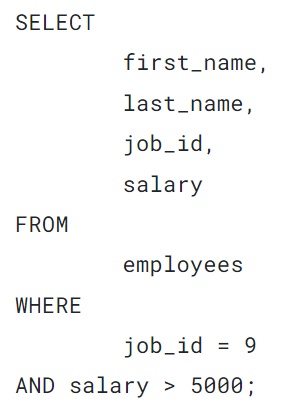
# **SQL AND**

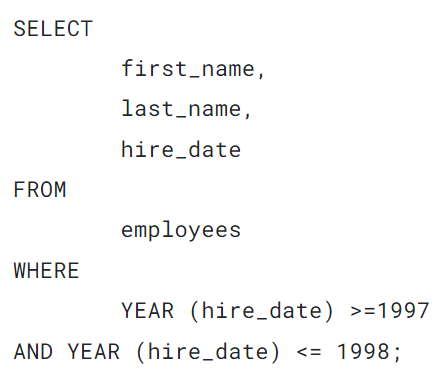




The following statement finds all employees who have both job id 9 and salary greater than 5,000:



To find all the employees who joined the company between 1997 and 1998, you use the AND operator as follows:

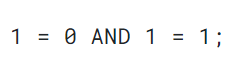


## **SQL AND operator and short-circuit evaluation**

The short-circuit feature allows the database system to stop evaluating the remaining parts of a logical expression as soon as it can determine the result.

Let’s take a look at an example to get a better understanding of how the to short-circuit evaluation feature works.

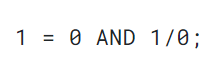
See the following condition:



The database system processes the two comparisons first and uses the AND operator to evaluate the two results.

However, with the short-circuit evaluation feature, the database system just has to evaluate the left part of the expression because the left part (1 = 0) returns false that causes the whole condition returns false regardless of the result of the right part of the condition.

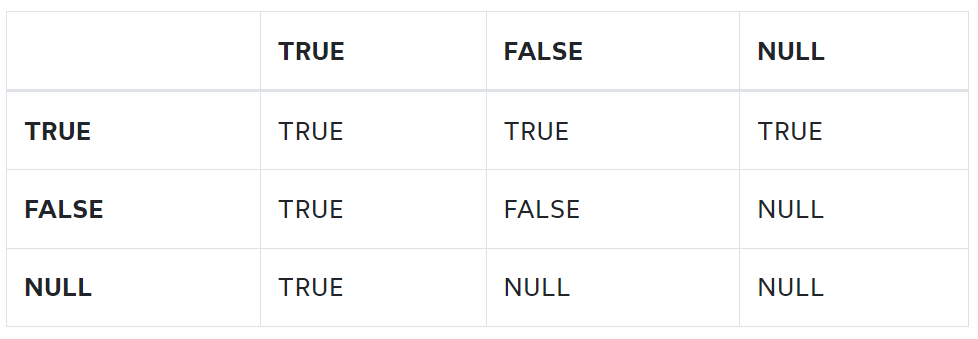
The short-circuit feature, therefore, decreases the CPU computation time and in some cases helps prevent runtime-error. Consider the following condition:



If the database system supports the short-circuit feature, it will not evaluate the right part of the expression (1/0) that causes the division by zero error.

Now, you should understand how the SQL AND operator works and how to apply it to form a complex condition in your queries.

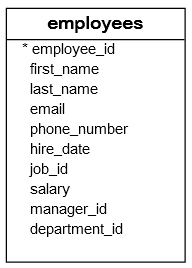
# **SQL OR**



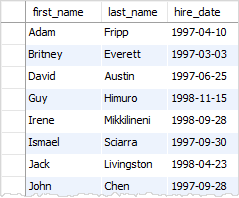
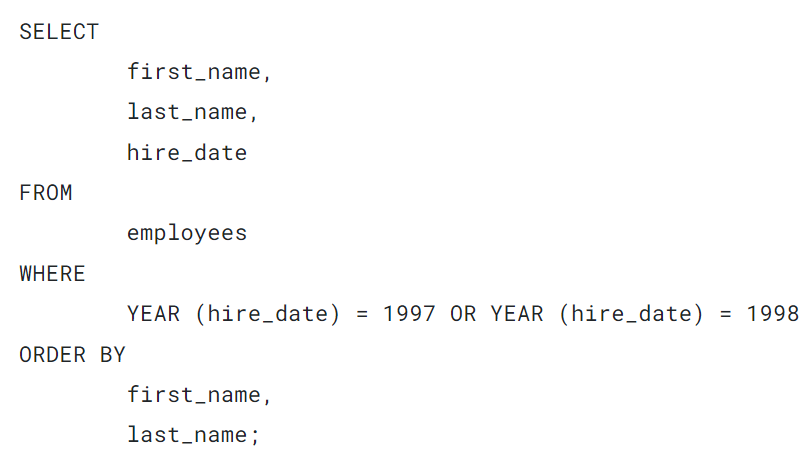
If the database system supports the short-circuit feature, the OR operator stops evaluating the remaining parts of the condition as soon as one expression is true.

Note that the short-circuit feature helps the database system save CPU computation by aborting processing the remaining part of a logical expression as soon as it can determine the result. For more information on the short-circuit feature, check it out the [SQL AND operator tutorial](https://www.sqltutorial.org/sql-and/).

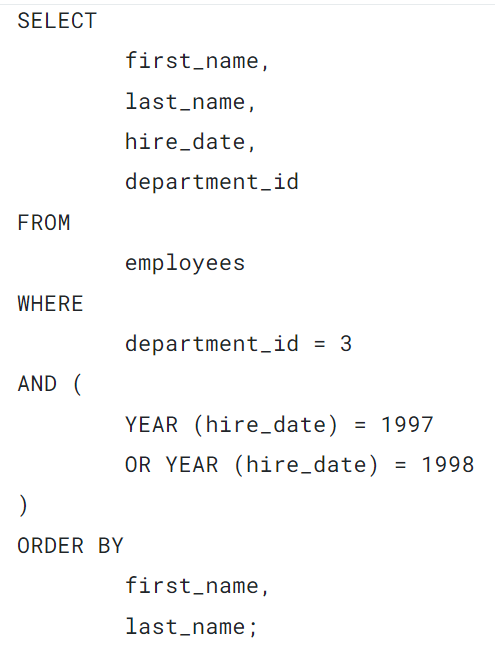
When you use the OR operator with the AND operator, the database system evaluates the OR operator after the AND operator. This is known as the rule of precedence. However, you can use parentheses to change the order of evaluation.



The following statement finds all employees who joined the company in 1997 or 1998.

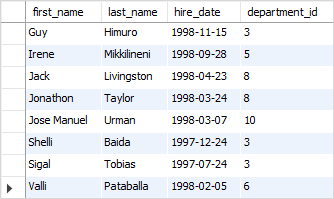
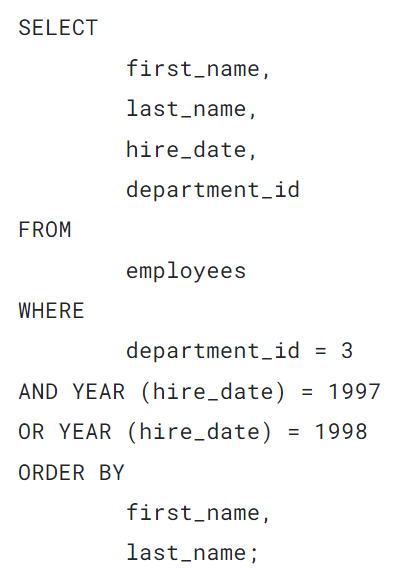


To find all employees who joined the company  in 1997 or 1997 and worked in the department id 3, you use both AND and OR operators as follows:



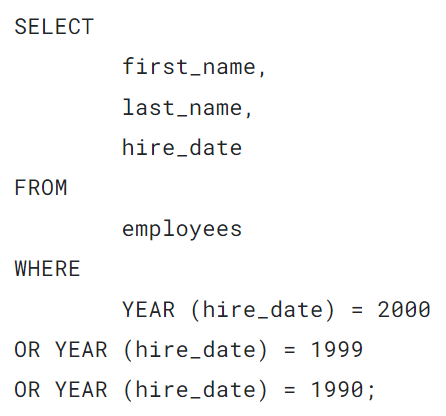
If you don’t use the parentheses, the query will retrieve employees who joined the company in 1997 and worked in department id 3 or employees who joined the company in 1998 regardless of departments.

This is because the database system evaluates the OR operator after the AND operator.

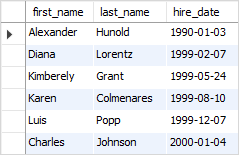
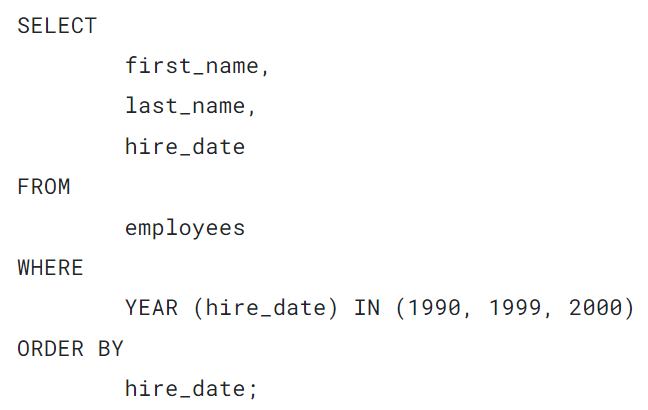


If a query uses many OR operators, it will become difficult to read. To make the query more readable, you can use the [IN](https://www.sqltutorial.org/sql-in/) operator instead.

For example, the following query finds all employees who joined the company in 1990 or 1999 or 2000.



You can replace the OR operators by the [IN](https://www.sqltutorial.org/sql-in/) operator as follows:



# **SQL BETWEEN**

## **Introduction to SQL BETWEEN operator**

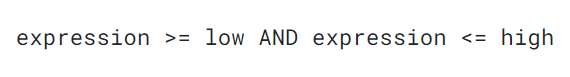
The BETWEEN operator is one of the [logical operators in SQL](https://www.sqltutorial.org/sql-logical-operators/). The BETWEEN operator checks if a value is within a range of values.

The syntax of the BETWEEN operator is as follows:



The BETWEEN operator returns true if the expression is greater than or equal to ( >=) the low value and less than or equal to ( <=) the high value.

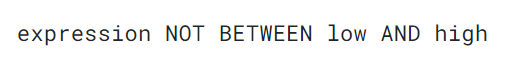
Technically, the BETWEEN is the equivalent to the following expression that uses the greater than or equal to (>=) and less than or equal to (<=) operators:



To compare a value with an exclusive range, you need to use the [comparison operators](https://www.sqltutorial.org/sql-comparison-operators/) less than (<) and greater than ( >).

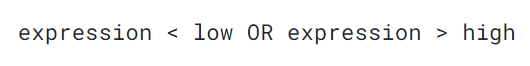
### **NOT BETWEEN**

To negate the result of the BETWEEN operator, you use the [NOT](https://www.sqltutorial.org/sql-not/) operator:

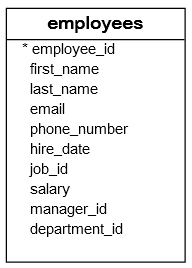


The NOT BETWEEN returns true if the expression is less than low or greater than (>) high; otherwise, it returns false.

Like the BETWEEN operator, you can rewrite the NOT BETWEEN operator using the less than (<) and greater than (>) operators with the OR operator as follows:

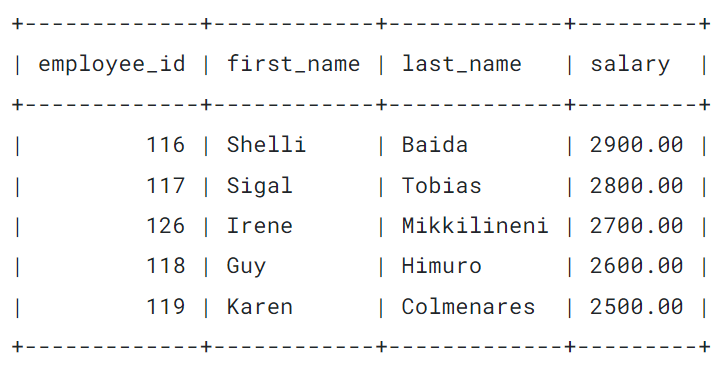
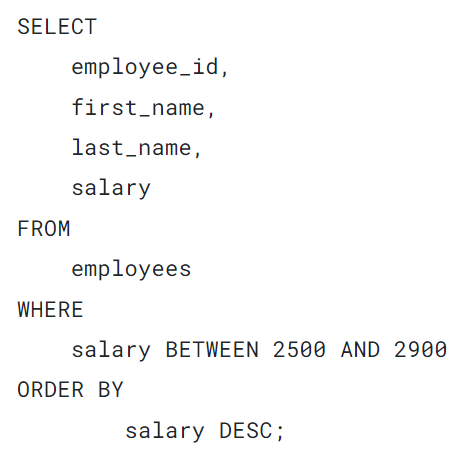


In practice, you often use the BETWEEN and NOT BETWEEN operator in the [WHERE](https://www.sqltutorial.org/sql-where/) clause of the [SELECT](https://www.sqltutorial.org/sql-select/) to select rows whose value of a column is within a specific range.



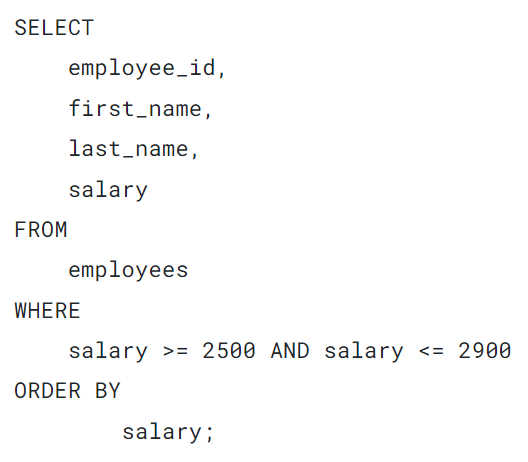
### **1) Using the SQL BETWEEN opeator with numbers example**

The following statement uses the BETWEEN operator to find all employees whose salaries are between 2,500 and 2,900:



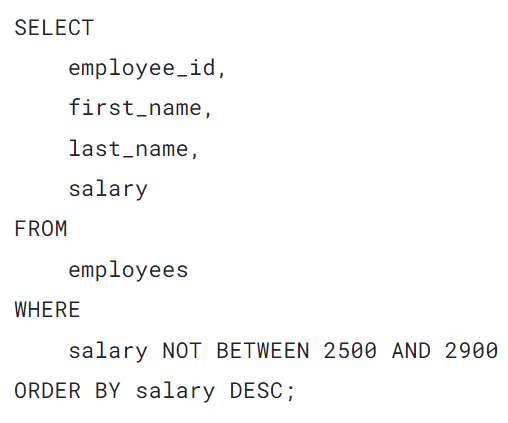
Notice that the result set includes the employees whose salaries are 2,500 and 2,900.

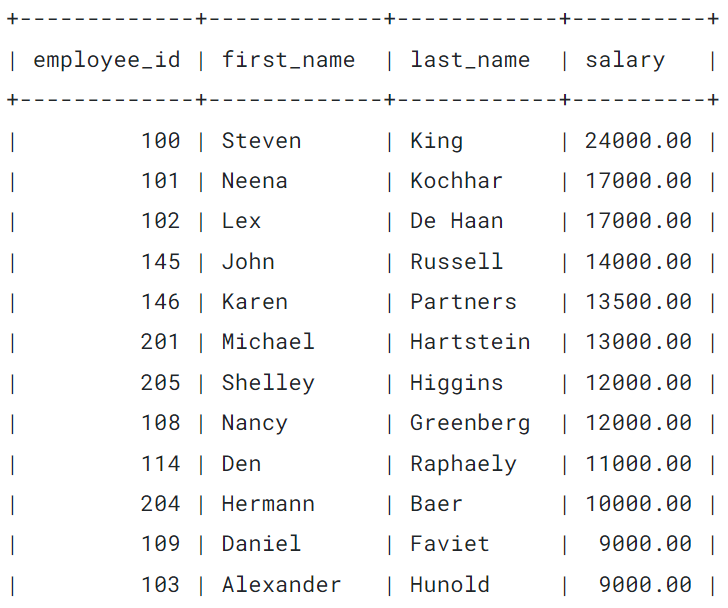
The following query returns the same result set as the above query. However, it uses comparison operators greater than or equal to (>=) and less than or equal to (<=) instead:



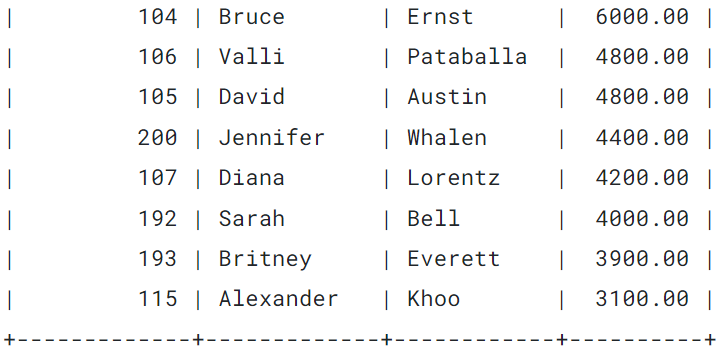
### **2) Using SQL NOT BETWEEN example**

The following example uses the NOT BETWEEN operator to find all employees whose salaries are not in the range of 2,500 and 2,900:



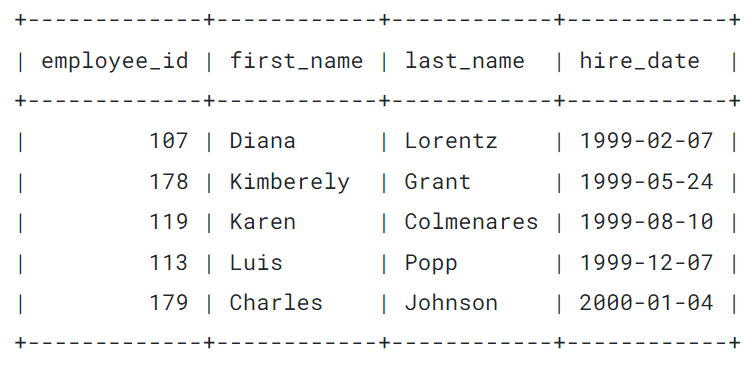
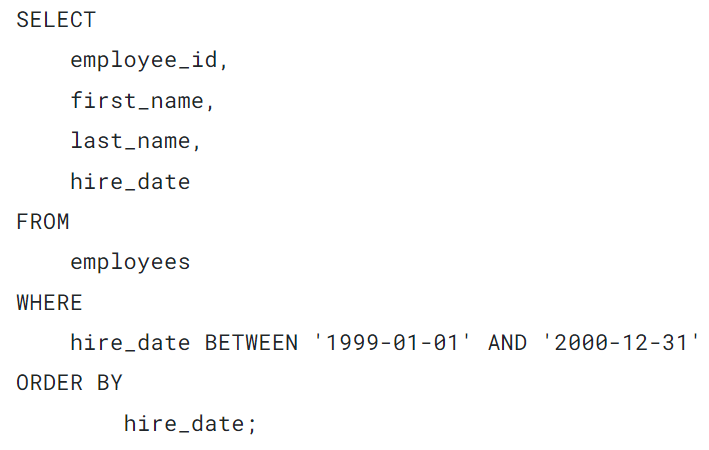


…..

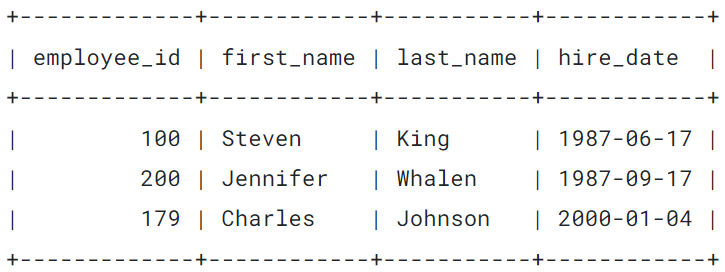
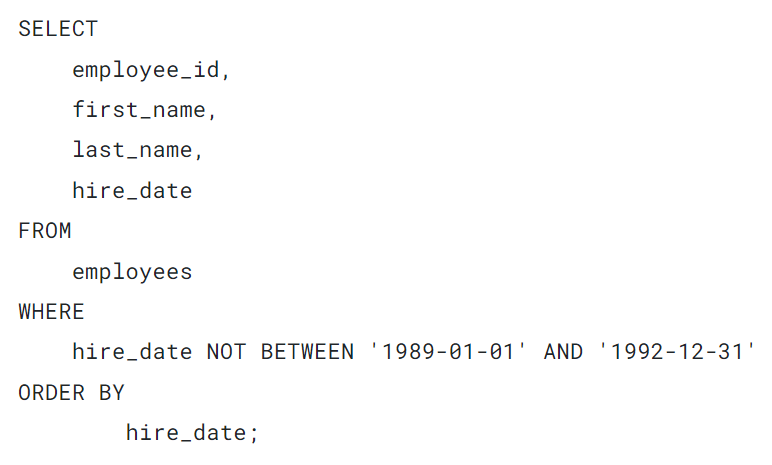


### **3) Using SQL BETWEEN operator with a date ranges**

The following example uses the BETWEEN operator to find all employees who joined the company between January 1, 1999, and December 31, 2000:

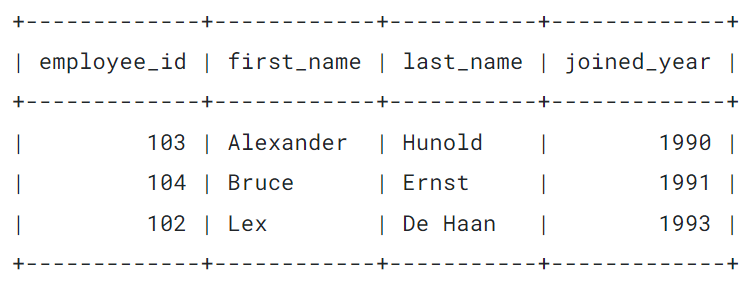
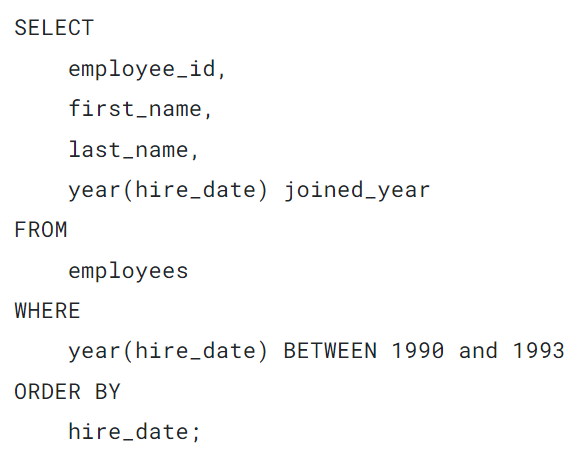


The following example uses the NOT BETWEEN operator to find employees who have not joined the company from January 1, 1989 to December 31, 1999:



### **4) Using SQL BETWEEN operator with a function example**

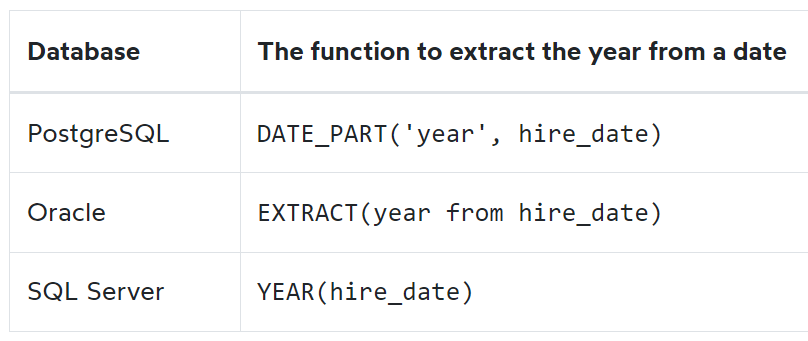
The following example uses the BETWEEN operator with the YEAR function to find employees who joined the company between 1990 and 1993:



In this example:

* First, the YEAR() function returns the year from the hire date.
* Second, the BETWEEN operator uses the result of the YEAR() function and check if it is within the range 1990 and 1993.

If your database doesn’t support the YEAR() function, you need to use a similar function:



## **Summary**

* The BETWEEN operator returns true if a value is within a specific range.
* Use the NOT operator to negate the BETWEEN opeator.

# **SQL IN**

The IN is a [logical operator](https://www.sqltutorial.org/sql-logical-operators/) in SQL. The IN operator returns true if a value is in a set of values or false otherwise.

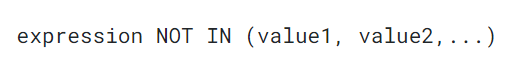
The following illustrates the syntax of the IN operator:



Technically, you can substitute the IN operator with the = and [OR](https://www.sqltutorial.org/sql-or/) operators The condition that uses the IN operator can be rewritten using one or more [OR](https://www.sqltutorial.org/sql-or/) operators as follows:



To negate the IN operator, you use the NOT operator:



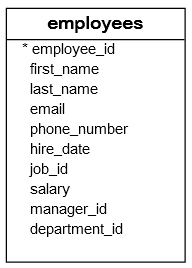
The NOT IN operator returns true if the expression does not equal any values in the list or false otherwise.

To substitute the IN operator, you can use the != and AND operators as follows:

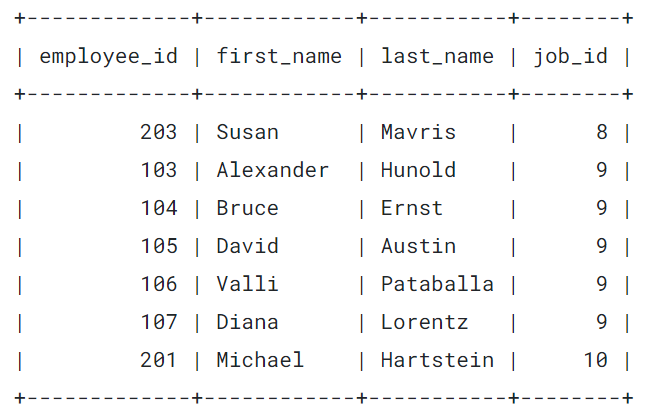
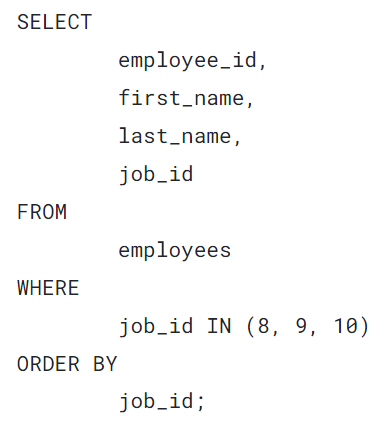


Notice that if any value in the list (value1,value2,...) is null, the IN operator returns no rows.

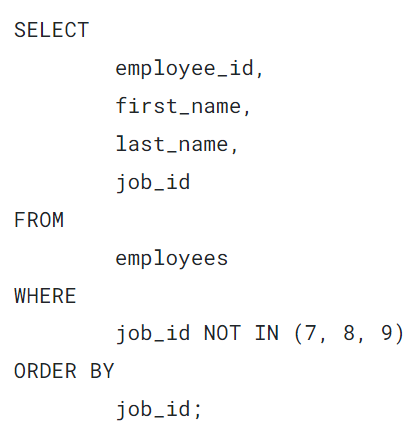
In practice, you often use the IN and NOT IN operators in the  [WHERE](https://www.sqltutorial.org/sql-where/) clause of the [SELECT](https://www.sqltutorial.org/sql-select/) statement to select rows with a value in a set of values. Also, you’ll use the IN operator in [subqueries](https://www.sqltutorial.org/sql-subquery/).

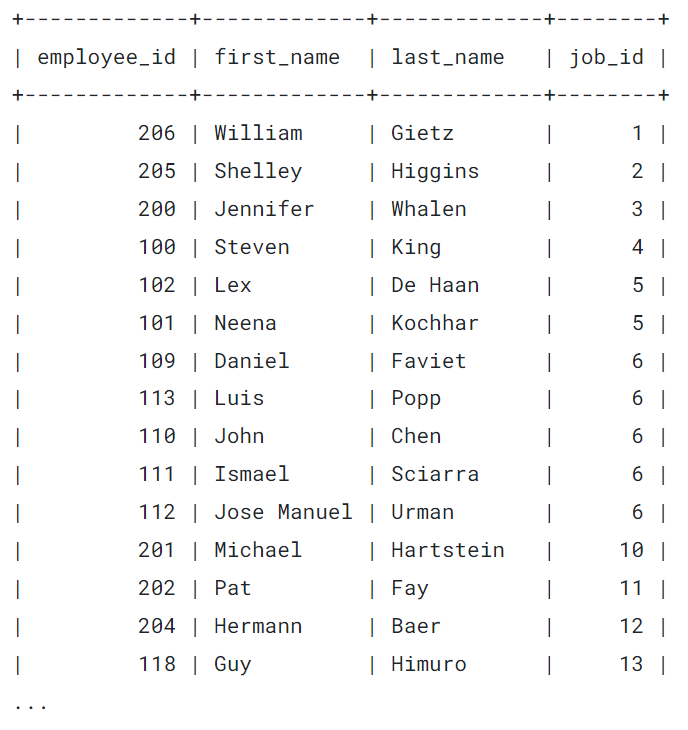


The following example uses the IN operator to find employees with the job id is 8, 9, or 10:



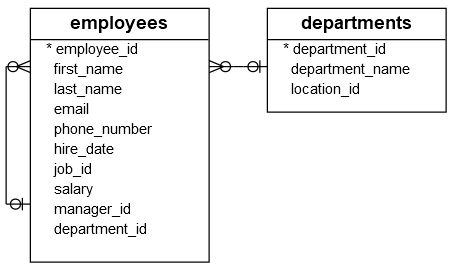
The following example uses the NOT IN operator to find employees whose job’s id is neither 7, 8, nor 9:



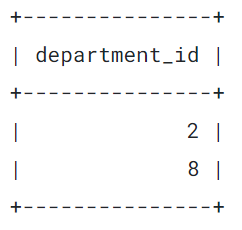
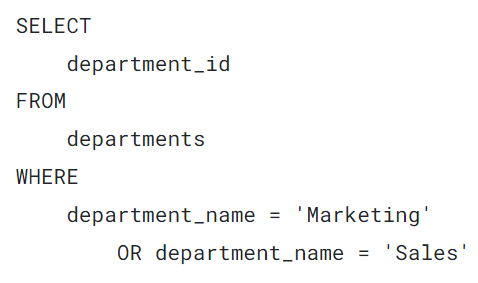


### **2) Using SQL IN operator with a subquery example**

A [subquery](https://www.sqltutorial.org/sql-subquery/) is a query nested inside another query. Let’s take a look at an example:

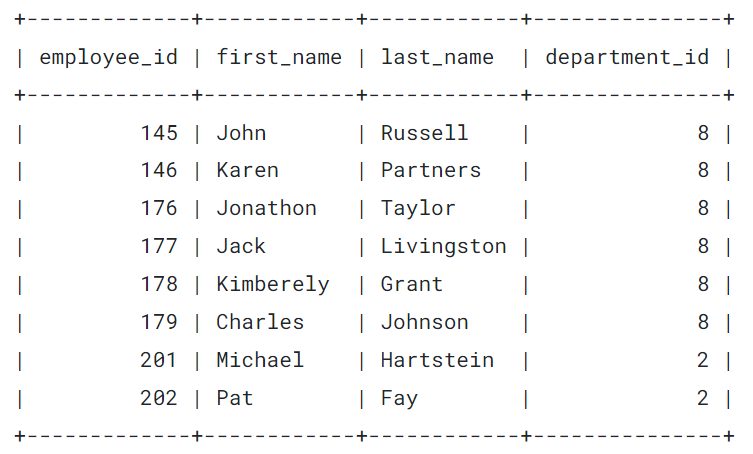
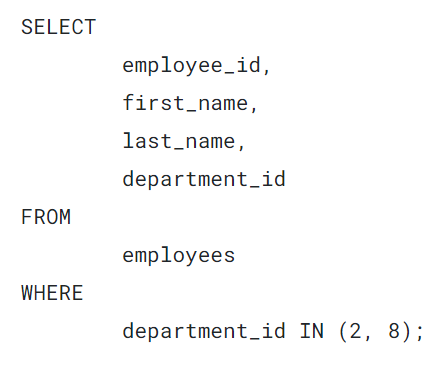


The following query returns the department id of the Marketing and Sales departments:

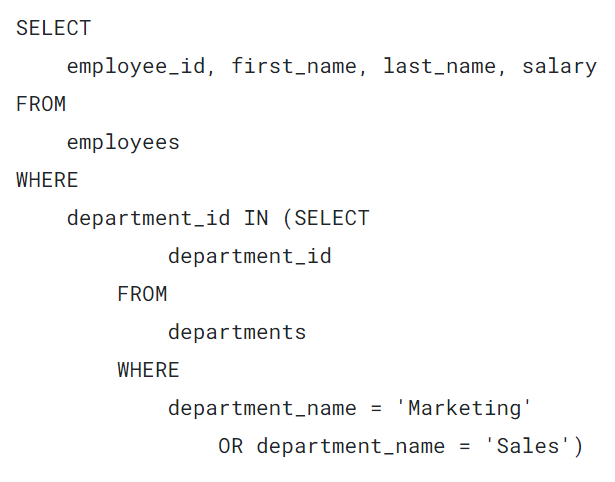


The query returns a list of two department ids.

And you can pass the id list to the IN operator to find employees who work in the Marketing and Sales departments like this:



To combine two above queries into a single query, you can use the first query in place of the list inside parentheses followed the IN operator:



## **Summary**

* Use the SQL IN operator to check if a value is in a set of values.
* Use the NOT opeator to negate the IN opeator, NOT IN.
* Use the `IN` with a subquery to combine two queries into a single query.

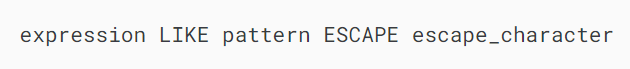
# **MySQL LIKE**

**Summary**: in this tutorial, you will learn how to use the MySQL LIKE operator to query data based on a specified pattern.

## **Introduction to MySQL LIKE operator**

The LIKE operator is a logical operator that tests whether a string contains a specified pattern or not.

Here’s the syntax of the LIKE operator:



In this syntax, if the expression matches the pattern, the LIKE operator returns 1. Otherwise, it returns 0.

MySQL provides two wildcard characters for constructing patterns: Percentage % and underscore \_ .

* The percentage ( % ) wildcard matches any string of zero or more characters.
* The underscore ( \_ ) wildcard matches any single character.

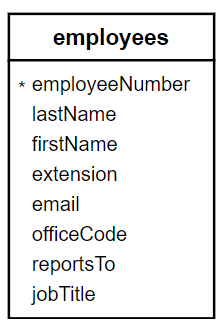
For example, s% matches any string starting with the character s such as sun and six. The se\_ matches any string starting with  se and is followed by any character such as see and sea.

When the pattern contains the wildcard character and you want to treat it as a regular character, you can use the ESCAPE clause.

Typically, you’ll use the LIKE operator in the [WHERE](https://www.mysqltutorial.org/mysql-basics/mysql-where/) clause of the [SELECT](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) , [DELETE](https://www.mysqltutorial.org/mysql-basics/mysql-delete/), and [UPDATE](https://www.mysqltutorial.org/mysql-basics/mysql-update/) statement.

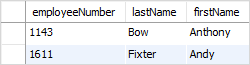
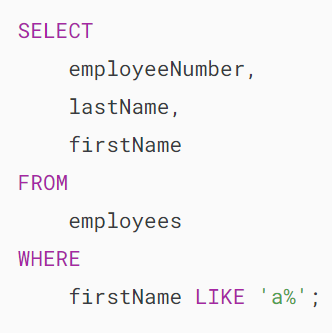
## **MySQL LIKE operator examples**

Let’s practice with some examples of using the LIKE operator. We will use the following employees table from the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/) for the demonstration:

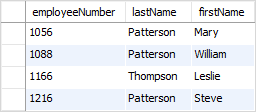
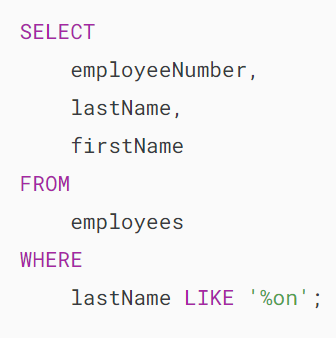


### **1) Using MySQL LIKE operator with the percentage (%) wildcard examples**

This example uses the LIKE operator to find employees whose first names start with the letter a:

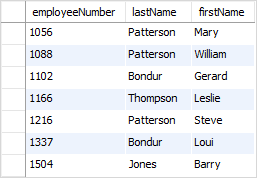
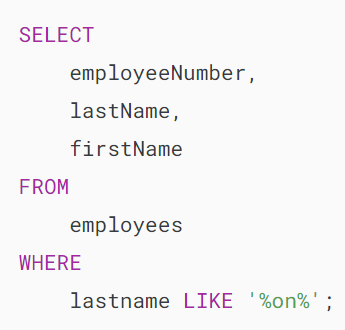
  
In this example, MySQL scans the whole employees table to find employees whose first names start with the letter a and are followed by any number of characters.

This example uses the LIKE operator to find employees whose last names end with the literal string on e.g., Patterson, Thompson:



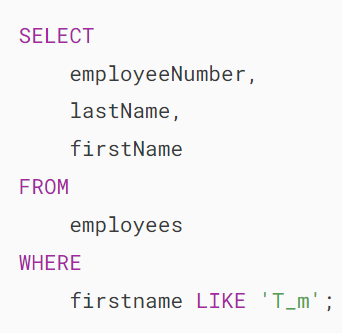
To check if a string contains a substring, you can use the percentage ( % ) wildcard at the beginning and the end of the substring.

For example, the following query uses the LIKE operator to find all employees whose last names contain the substring on:



### **2) Using MySQL LIKE operator with an underscore( \_ ) wildcard examples**

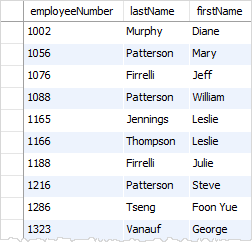
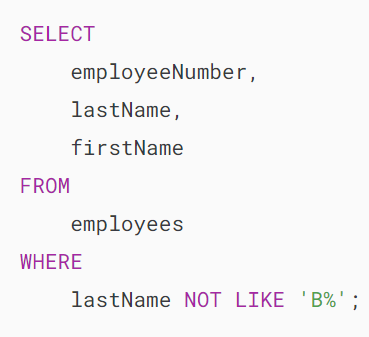
To find employees whose first names start with the letter T , end with the letter m, and contain any single character between e.g., Tom , Tim, you use the underscore (\_) wildcard to construct the pattern as follows:

mysql-like-with-_-pattern

### **3) Using MySQL NOT LIKE operator example**

The MySQL allows you to combine the NOT operator with the LIKE operator to find a string that does not match a specific pattern.

Suppose you want to search for employees whose last names don’t start with the letter B, you can use the NOT LIKE operator as follows:



Note that the pattern is not case-sensitive. Therefore, the b% and B% patterns return the same result.

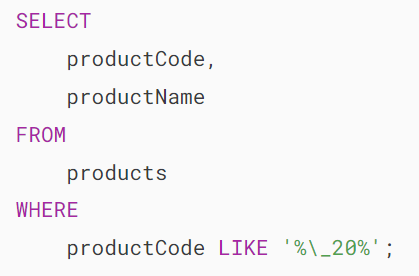
## **MySQL LIKE operator with the ESCAPE clause**

Sometimes the pattern may contain the wildcard characters e.g., 10%, \_20, etc.

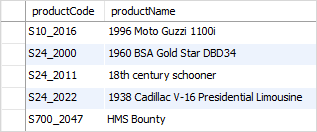
In this case, you can use the ESCAPE clause to specify the escape character so that the LIKE operator interprets the wildcard character as a literal character.

If you don’t specify the escape character explicitly, the backslash character (\) is the default escape character.

For example, if you want to find products whose product codes contain the string \_20 , you can use the pattern %\\_20% with the default escape character:



Alternatively, you can specify a different escape character e.g., $ using the ESCAPE clause:



The pattern %$\_20% matches any string that contains the \_20 string.

## **Summary**

* Use the LIKE operator to test if a value matches a pattern.
* The % wildcard matches zero or more characters.
* The \_ wildcard matches a single character.
* Use ESCAPE clause specifies an escape character other than the default escape character (\).
* Use the NOT operator to negate the LIKE operator.

# **MySQL LIMIT**

**Summary**: in this tutorial, you will learn how to use the MySQL LIMIT clause to constrain the number of rows returned by a query.

## **Introduction to MySQL LIMIT clause**

The LIMIT clause is used in the [SELECT](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) statement to constrain the number of rows to return. The LIMIT clause accepts one or two arguments. The values of both arguments must be zero or positive [integers](https://www.mysqltutorial.org/mysql-basics/mysql-int/).

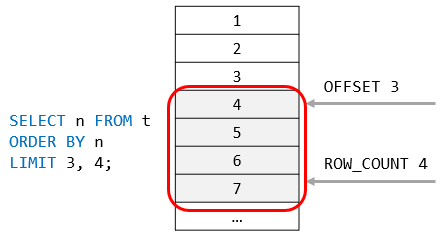
The following illustrates the LIMIT clause syntax with two arguments:



In this syntax:

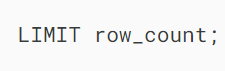
* The offset specifies the offset of the first row to return. The offset of the first row is 0, not 1.
* The row\_count specifies the maximum number of rows to return.

The following picture illustrates the LIMIT clause:

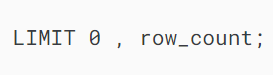


When you use the LIMIT clause with one argument, MySQL will use this argument to determine the maximum number of rows to return from the first row of the result set.

Therefore, these two clauses are equivalent:



And



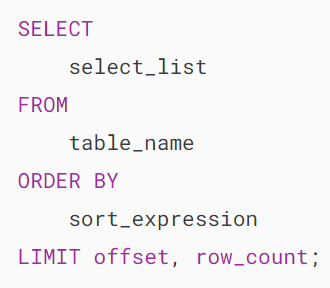
In addition to the above syntax, MySQL provides the following alternative LIMIT clause syntax:

### 

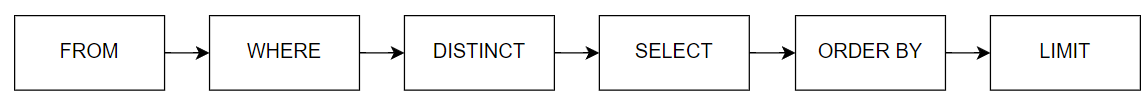
### **The LIMIT and ORDER BY clauses**

By default, the SELECT statement returns rows in an unspecified order. When you add the LIMIT clause to the SELECT statement, the returned rows are unpredictable.

Therefore, to ensure the LIMIT clause returns an expected output, you should always use it with an [ORDER BY](https://www.mysqltutorial.org/mysql-basics/mysql-order-by/) clause like this:

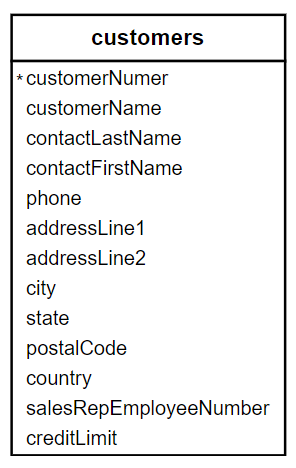


The following picture illustrates the evaluation order of the LIMIT clause in the SELECT statement:



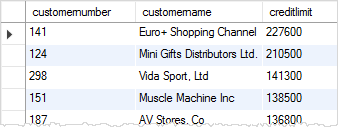
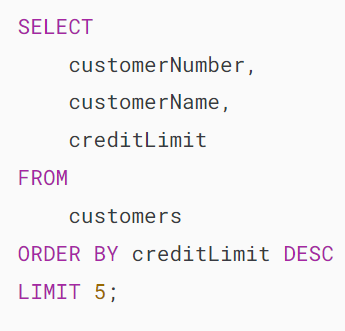
## **MySQL LIMIT clause examples**

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



### **1) Using MySQL LIMIT to get the highest or lowest rows**

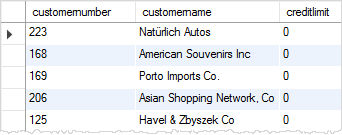
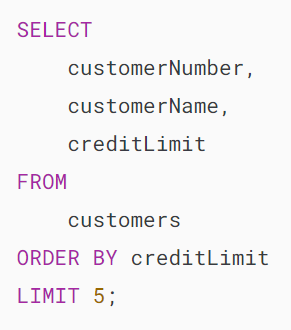
This statement uses the LIMIT clause to get the top five customers who have the highest credit:



In this example:

* First, the ORDER BY clause sorts the customers by credits from high to low.
* Then, the LIMIT clause returns the first 5 rows.

Similarly, this example uses the LIMIT clause to find five customers who have the lowest credits:

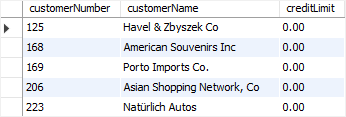
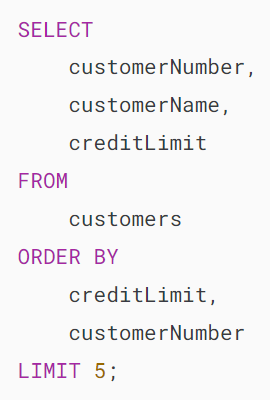


In this example:

* First, the ORDER BY clause sorts the customers by credits from low to high.
* Then, the LIMIT clause returns the first 5 rows.

Because there are more than 5 customers that have credits zero, the result of the query above may lead to an inconsistent result.

To fix this issue, you need to add more columns to the ORDER BY clause to constrain the row in unique order:

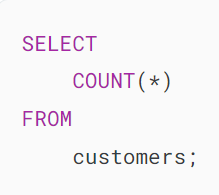
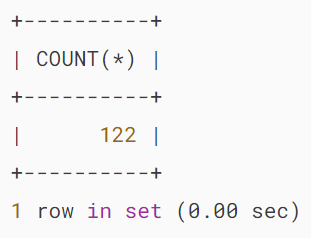


### **2) Using MySQL LIMIT clause for pagination**

When you display data on the screen, you often want to divide rows into pages, where each page contains a limited number of rows like 10 or 20.

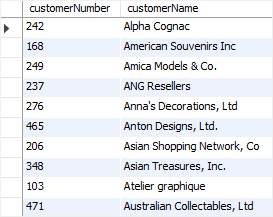
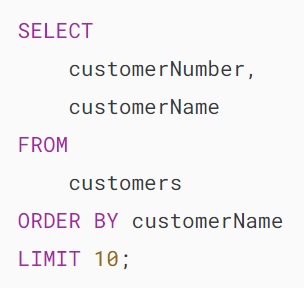
To calculate the number of pages, you take the total rows divided by the number of rows per page. For fetching rows of a specific page, you can use the LIMIT clause.

This query uses the COUNT(\*) [aggregate function](https://www.mysqltutorial.org/mysql-aggregate-functions/) to get the total rows from the customers table:

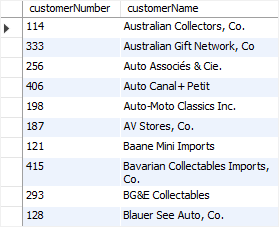
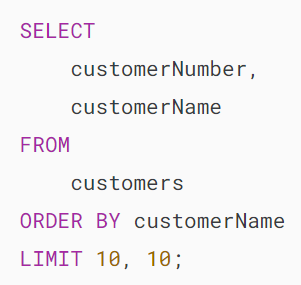
 

Suppose that each page has 10 rows; to display 122 customers, you have 13 pages. The last 13th page contains two rows only.

This query uses the LIMIT clause to get rows of page 1 which contains the first 10 customers sorted by the customer name:



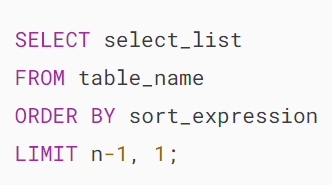
This query uses the LIMIT clause to get the rows of the second page that include rows 11 – 20:



In this example, the clause LIMIT 10, 10 returns 10 rows for the rows 11 – 20.

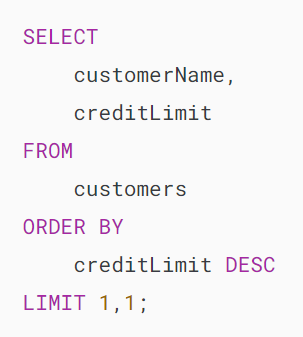
### **3) Using MySQL LIMIT to get the nth highest or lowest value**

To get the nth highest or lowest value, you use the following LIMIT clause:

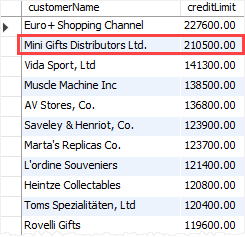
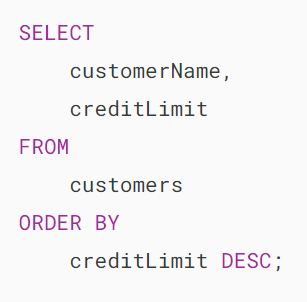


The clause LIMIT n-1, 1 returns 1 row starting at the row n.

For example, the following finds the customer who has the second-highest credit:

MySQL LIMIT find nth highest row example

Let’s double-check the result. This query returns all customers sorted by credits from high to low:



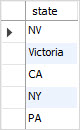
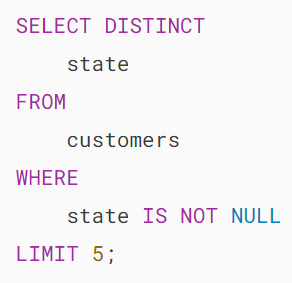
As you can see clearly from the output, the result was correct as expected.

Note that this technique works when there are no two customers who have the same credit limits. To get a more accurate result, you should use the [DENSE\_RANK()](https://www.mysqltutorial.org/mysql-window-functions/mysql-dense_rank-function/) [window function](https://www.mysqltutorial.org/mysql-window-functions/).

## **MySQL LIMIT & DISTINCT clauses**

If you use the LIMIT clause with the [DISTINCT](https://www.mysqltutorial.org/mysql-basics/mysql-distinct/) clause, MySQL immediately stops searching when it finds the number of unique rows specified in the LIMIT clause.

The example uses the LIMIT clause with the DISTINCT clause to return the first five unique states in the customers table:



## **Summary**

* Use the MySQL LIMIT clause to constrain the number of rows returned by the SELECT statement.

# **SQL IS NULL**

## **What is NULL**

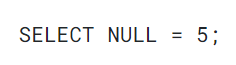
NULL is special in SQL. NULL indicates that the data is unknown, inapplicable, or even does not exist. In other words, NULL represents the missing data in the database.

For example, if employees do not have phone numbers, you can store their phone numbers as empty strings.

However, if you don’t know their phone numbers when you save the employee records, you need to use the NULL for the unknown phone numbers.

The NULL is special because any comparisons with a NULL can never result in true or false, but in a third logical result, unknown.

The following statement returns NULL:

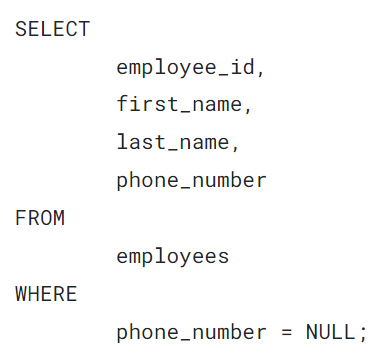


The NULL value is not even equal to itself, as shown in the following statement:



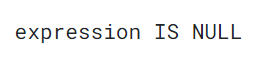
In this example, the result is NULL.

You cannot use the [comparison operator](https://www.sqltutorial.org/sql-comparison-operators/) equal to (=) to compare a value to a NULL value. For example, the following statement will not return the correct result:



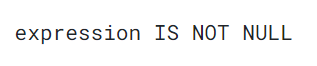
## **The IS NULL and IS NOT NULL operators**

To determine whether an expression or column is NULL or not, you use the IS NULL operator as follows:



If the result of the expression is NULL, IS NULL operator returns true; otherwise, it returns false.

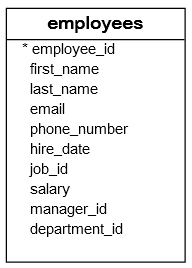
To check if an expression or column is not NULL, you use the IS NOT operator:



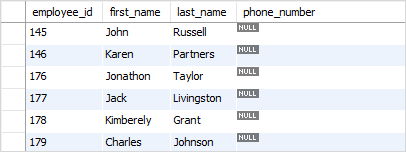
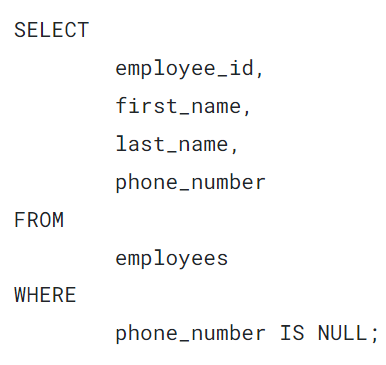
The IS NOT NULL returns false if the value of the expression is NULl; otherwise, it returns true;

## **SQL IS NULL and IS NOT NULL examples**

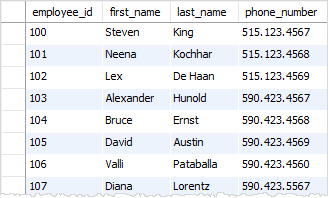
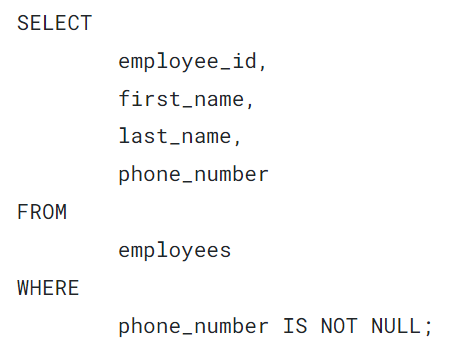
In these examples, we’ll use the  employees table from the [sample database](https://www.sqltutorial.org/sql-sample-database/) for the demonstration.



To find all employees who do not have the phone numbers, you use the IS NULL operator as follows:



To find all employees who have phone numbers, you use IS NOT NULL as shown in the following statement:



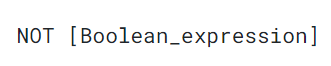
Now you should understand the NULL concept and know how to use the SQL IS operator to check whether a value is NULL or not

# **SQL NOT**

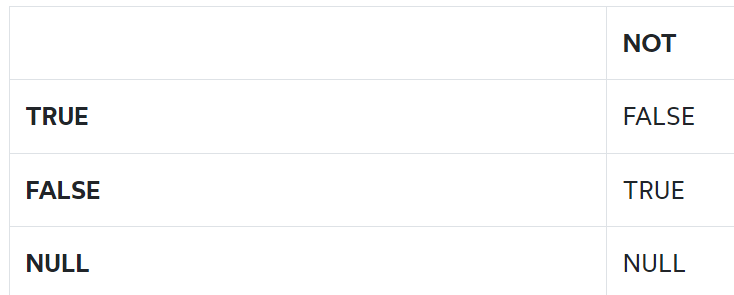
## **Introduction to the SQL NOT operator**

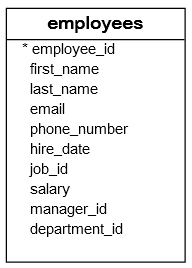
You have learned how to use various logical operators such as [AND](https://www.sqltutorial.org/sql-and/), [OR](https://www.sqltutorial.org/sql-or/), [LIKE](https://www.sqltutorial.org/sql-like/), [BETWEEN](https://www.sqltutorial.org/sql-between/), [IN](https://www.sqltutorial.org/sql-in/), and [EXISTS](https://www.sqltutorial.org/sql-exists/). These operators help you to form flexible conditions in the [WHERE clause](https://www.sqltutorial.org/sql-where/).

To negate the result of any Boolean expression, you use the NOT operator. The following illustrates how to use the NOT operator:

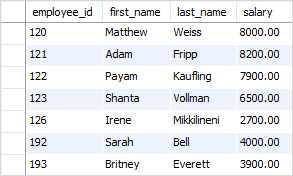
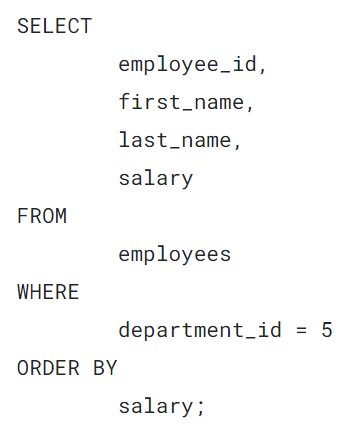


The following table shows the result of the NOT operator.

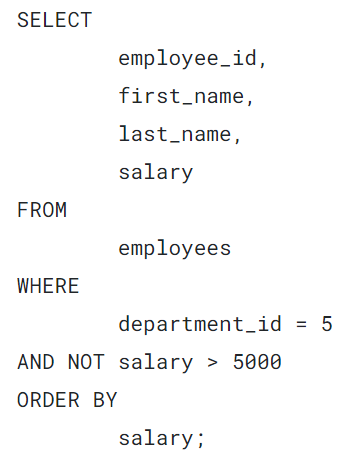




The following statement retrieves all employees who work in the department id 5.

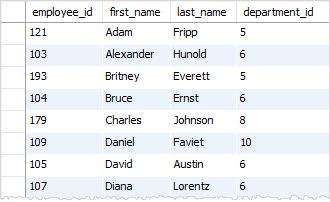
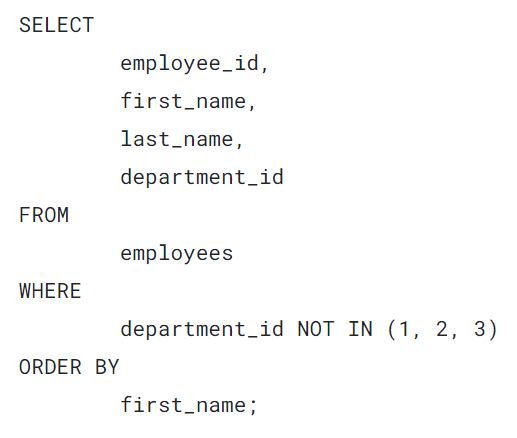


To get the employees who work in the department id 5 and with a salary not greater than 5000.



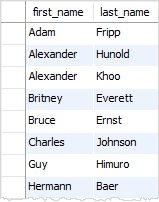
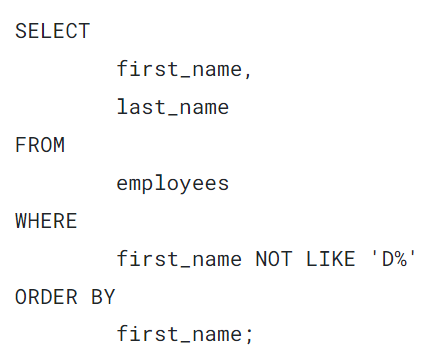
### **SQL NOT with IN operator example**

To negate the [IN](https://www.sqltutorial.org/sql-in/) operator, you use the NOT operator. For example, the following statement gets all the employees who are not working in the departments 1, 2, or 3.



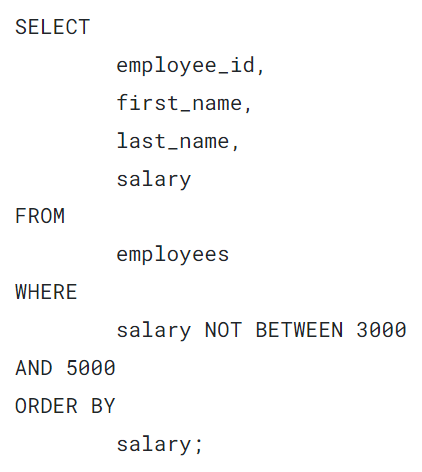
### **SQL NOT LIKE operator example**

You can negate the [LIKE](https://www.sqltutorial.org/sql-like/) operator by using the NOT LIKE. For example, the following statement retrieves all the employees whose first names do not start with the letter D.



### **SQL NOT BETWEEN example**

The following example shows you how to use the NOT to negate the [BETWEEN](https://www.sqltutorial.org/sql-between/) operator to get employees whose salaries are not between 5,000 and 1,000.

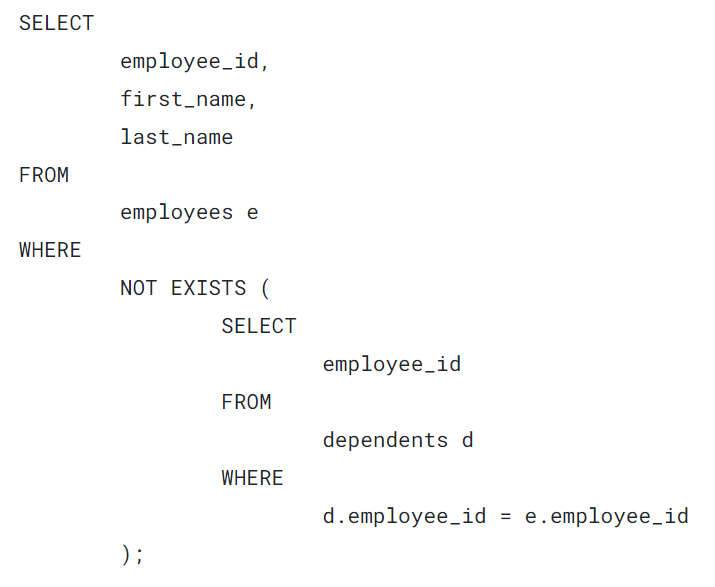


### **SQL NOT EXISTS example**

See the following employees and dependents tables:



The following query uses the NOT EXISTS operator to get the employees who do not have any dependents.



Now you should know how to use the NOT operator to negate a Boolean expression.