# **MySQL String Functions**

This page shows you the most commonly used MySQL string functions that allow you to manipulate character string data effectively.

| **Name** | **Description** |
| --- | --- |
| [CONCAT](https://www.mysqltutorial.org/mysql-string-functions/mysql-concat/) | Concatenate two or more strings into a single string. |
| [CONCAT\_WS](https://www.mysqltutorial.org/mysql-string-functions/mysql-concat_ws/) | Return a single string by concatenating multiple strings separated by a specified separator. |
| [INSTR](https://www.mysqltutorial.org/mysql-string-functions/mysql-instr/) | Return the position of the first occurrence of a substring in a string. |
| [LENGTH](https://www.mysqltutorial.org/mysql-string-functions/mysql-string-length/) | Get the length of a string in bytes. |
| [CHAR\_LENGTH](https://www.mysqltutorial.org/mysql-string-functions/mysql-char_length/) | Return the length of a string measured in characters. |
| [LEFT](https://www.mysqltutorial.org/mysql-string-functions/mysql-left-function/) | Get a specified number of leftmost characters from a string. |
| [LOWER](https://www.mysqltutorial.org/mysql-string-functions/mysql-lower/) | Return a string converted to lowercase. |
| [LOCATE](https://www.mysqltutorial.org/mysql-string-functions/mysql-locate-function/) | Return the position of a substring within a given string starting at a specified position. |
| [LTRIM](https://www.mysqltutorial.org/mysql-string-functions/mysql-ltrim-function/) | Remove all leading spaces from a string. |
| [REPLACE](https://www.mysqltutorial.org/mysql-string-functions/mysql-replace-function/) | Replace all occurrences of a substring in a string with a new one. |
| [REPEAT](https://www.mysqltutorial.org/mysql-string-functions/mysql-repeat-function/) | Repeat a string a specified number of time. |
| [REVERSE](https://www.mysqltutorial.org/mysql-string-functions/mysql-reverse-function/) | Reverse a string. |
| [RIGHT](https://www.mysqltutorial.org/mysql-string-functions/mysql-right-function/) | Get a specified number of rightmost characters from a string. |
| [RTRIM](https://www.mysqltutorial.org/mysql-string-functions/mysql-rtrim-function/) | Remove all trailing spaces from a string. |
| [SUBSTRING](https://www.mysqltutorial.org/mysql-string-functions/mysql-substring/) | Extract a substring starting from a position with a specific length. |
| [SUBSTRING\_INDEX](https://www.mysqltutorial.org/mysql-string-functions/mysql-substring_index-function/) | Return a substring from a string before a specified number of occurrences of a delimiter. |
| [TRIM](https://www.mysqltutorial.org/mysql-string-functions/mysql-trim-function/) | Remove unwanted characters from a string. |
| [FIND\_IN\_SET](https://www.mysqltutorial.org/mysql-string-functions/mysql-find_in_set/) | Find a string within a comma-separated list of strings. |
| [FORMAT](https://www.mysqltutorial.org/mysql-string-functions/mysql-format-function/) | Format a number with a specific locale, rounded to the number of decimals. |
| [UPPER](https://www.mysqltutorial.org/mysql-string-functions/mysql-upper/) | Convert a string to uppercase. |

## **Concatenation Functions**

* [CONCAT()](https://www.mysqltutorial.org/mysql-string-functions/mysql-concat/): Combines two or more strings into a single string.
* [CONCAT\_WS()](https://www.mysqltutorial.org/mysql-string-functions/mysql-concat_ws/): Combines multiple strings into a single string with a specified separator.

## **Substring Functions**

* [SUBSTRING()](https://www.mysqltutorial.org/mysql-string-functions/mysql-substring/): Extracts a substring from a given string.
* [SUBSTRING\_INDEX()](https://www.mysqltutorial.org/mysql-string-functions/mysql-substring_index-function/): Extracts a substring from a string using a delimiter.
* [LEFT()](https://www.mysqltutorial.org/mysql-string-functions/mysql-left-function/): Returns a specified number of characters from the beginning of a string.
* [RIGHT()](https://www.mysqltutorial.org/mysql-string-functions/mysql-right-function/): Returns a specified number of characters from the end of a string.
* [MID()](https://www.mysqltutorial.org/mysql-string-functions/mysql-substring/): Extracts a substring from the middle of a string. The MID() function is a synonym for SUBSTRING().

## **Searching and Locating Functions**

* [LOCATE()](https://www.mysqltutorial.org/mysql-string-functions/mysql-locate-function/): Finds the position of a substring within a string.
* [POSITION()](https://www.mysqltutorial.org/mysql-string-functions/mysql-locate-function/): Finds the position of a substring. The POSITION() is a synonym for the LOCATE() function.
* [INSTR()](https://www.mysqltutorial.org/mysql-string-functions/mysql-instr/): Another function for finding the position of a substring.

## **Case Conversion Functions**

* [UPPER()](https://www.mysqltutorial.org/mysql-string-functions/mysql-upper/): Converts a string to uppercase.
* [LOWER()](https://www.mysqltutorial.org/mysql-string-functions/mysql-lower/): Converts a string to lowercase.

## **Character Manipulation Functions**

* [REPLACE()](https://www.mysqltutorial.org/mysql-string-functions/mysql-replace-function/): Replaces all occurrences of a substring in a string.
* [TRIM()](https://www.mysqltutorial.org/mysql-string-functions/mysql-trim-function/): Removes leading and trailing spaces from a string.
* [LTRIM()](https://www.mysqltutorial.org/mysql-string-functions/mysql-ltrim-function/): Removes leading spaces from a string.
* [RTRIM()](https://www.mysqltutorial.org/mysql-string-functions/mysql-rtrim-function/): Removes trailing spaces from a string.
* [REPEAT()](https://www.mysqltutorial.org/mysql-string-functions/mysql-repeat-function/): Repeats a string a specified number of times.
* [REVERSE()](https://www.mysqltutorial.org/mysql-string-functions/mysql-reverse-function/): Reverses the characters in a string.
* [INSERT()](https://www.mysqltutorial.org/mysql-string-functions/mysql-insert-function/): Replaces a substring within a string with a new substring.

## **Whitespace Functions**

* SPACE(): Returns a string consisting of spaces.
* [ASCII()](https://www.mysqltutorial.org/mysql-string-functions/mysql-ascii/): Returns the ASCII value of the leftmost character of a string.
* CHAR(): Converts an ASCII value to a character.

## **Length and Count Functions**

* [LENGTH()](https://www.mysqltutorial.org/mysql-string-functions/mysql-string-length/): Returns the length of a string in bytes.
* [CHAR\_LENGTH()](https://www.mysqltutorial.org/mysql-string-functions/mysql-char_length/): Returns the length of a string in characters.
* OCTET\_LENGTH(): Returns the length of a string in bytes.
* BIT\_LENGTH(): Returns the length of a string in bits.
* [CHARACTER\_LENGTH()](https://www.mysqltutorial.org/mysql-string-functions/mysql-char_length/): Returns the length of a string in characters.
* BIT\_COUNT(): Counts the number of bits in a binary string.
* STRCMP(): Compares two strings and returns their relative order.

## **Padding string functions**

* [LPAD()](https://www.mysqltutorial.org/mysql-string-functions/mysql-lpad/) – Left-pads a string with a set of characters to a specified length.
* [RPAD()](https://www.mysqltutorial.org/mysql-string-functions/mysql-rpad/) – Right-pads a string with a set of characters to a specified length.

<https://www.mysqltutorial.org/mysql-string-functions/mysql-ascii/>

<https://www.mysqltutorial.org/mysql-string-functions/mysql-bin-function/>

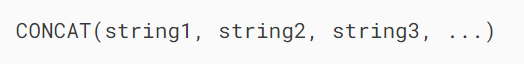
# **MySQL CONCAT() Function**

**Summary**: in this tutorial, you will learn how to use the MySQL CONCAT() function to concatenate multiple strings into a single string.

## **Introduction to the MySQL CONCAT function**

The CONCAT() function allows you to concatenate multiple strings into a single string.

Here’s the basic syntax of the CONCAT() function:



The CONCAT() function accepts a variable number of input strings: string1, string2, string3, …

It returns a single string that combines the string arguments string1, string2, and string3 …

If any string is NULL, the CONCAT() function returns NULL.

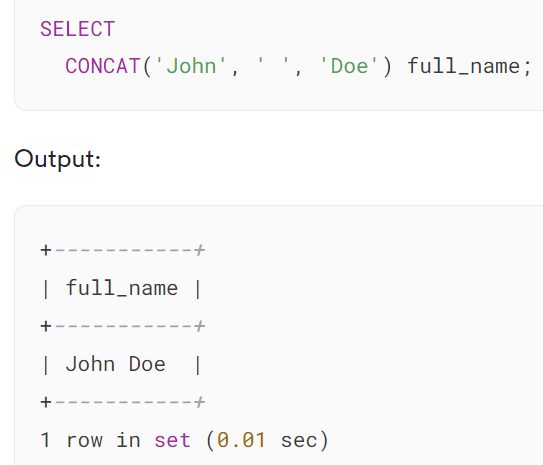
In addition, if you pass numbers to the CONCAT() function, it’ll convert these numbers to their equivalent strings before concatenation.

## **MySQL CONCAT function examples**

Let’s take some examples of using the CONCAT() function.

### **1) Simple CONCAT function example**

The following example uses the CONCAT() function to concatenate the quoted string into a single string:



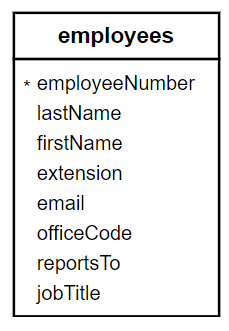
In this example, we use the CONCAT() function to combine three quoted strings ‘John’, ‘ ‘, and ‘Doe’ into the full name.

If you only concatenate the quoted strings, you can concatenate them by placing the strings next to each other like this:

### 

### **2) Using MySQL CONCAT with table data**

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

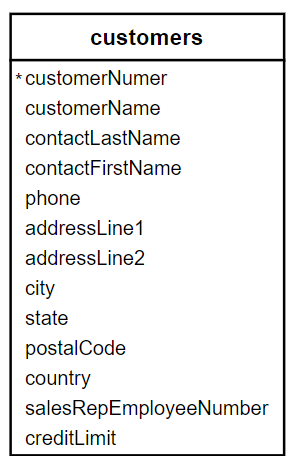


The following example uses the CONCAT function to concatenate the first name, space, and last name into the full name:

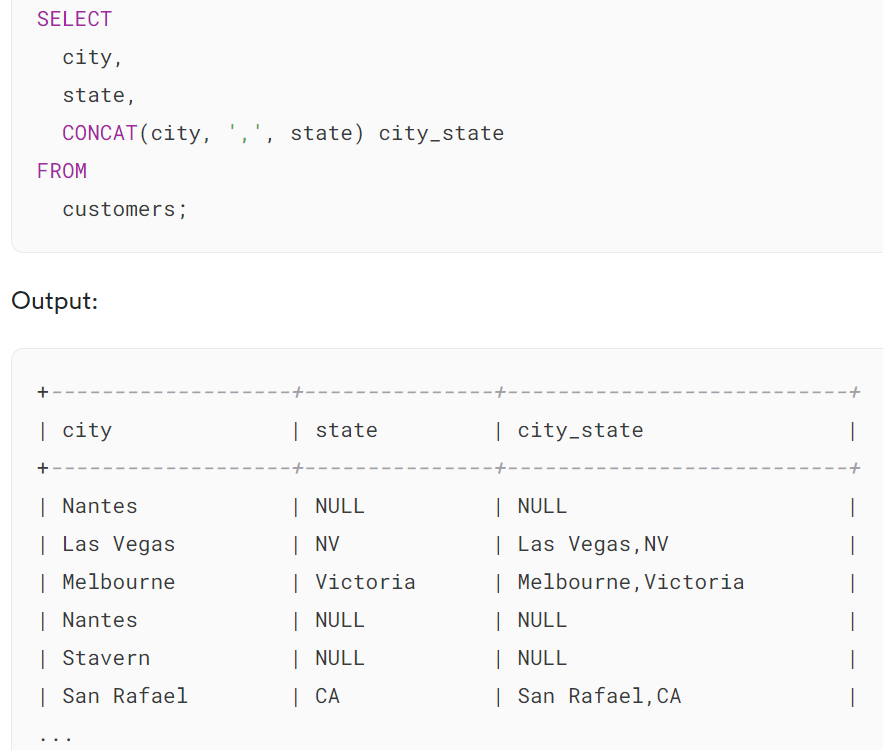
### 

### **3) Using MySQL CONCAT with NULL values**

We’ll take the customers table for the demonstration:

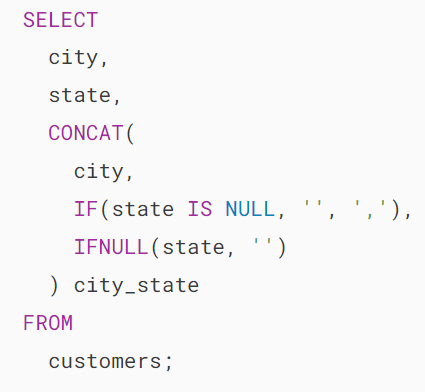


The following example uses the CONCAT function to concatenate the values in the city and state columns into a single string:

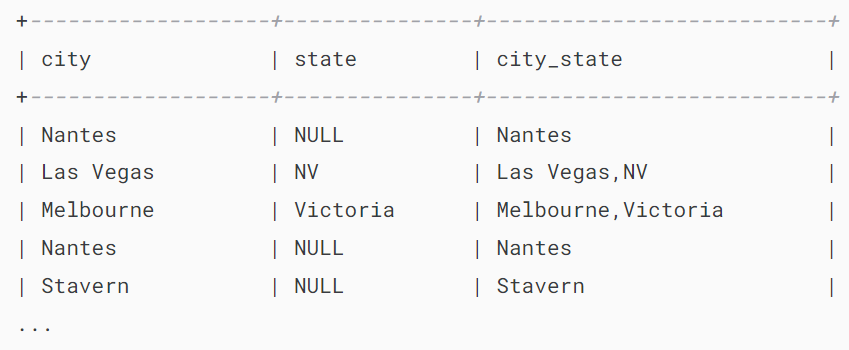


The output indicates that if the state is NULL, the result string will be NULL. To handle the NULL values gracefully, you can use the CONCAT\_WS function.

Alternatively, you can use null-related functions such as [IF](https://www.mysqltutorial.org/mysql-control-flow-functions/mysql-if-function/) and [IFNULL](https://www.mysqltutorial.org/mysql-control-flow-functions/mysql-ifnull/). For example:



Output:

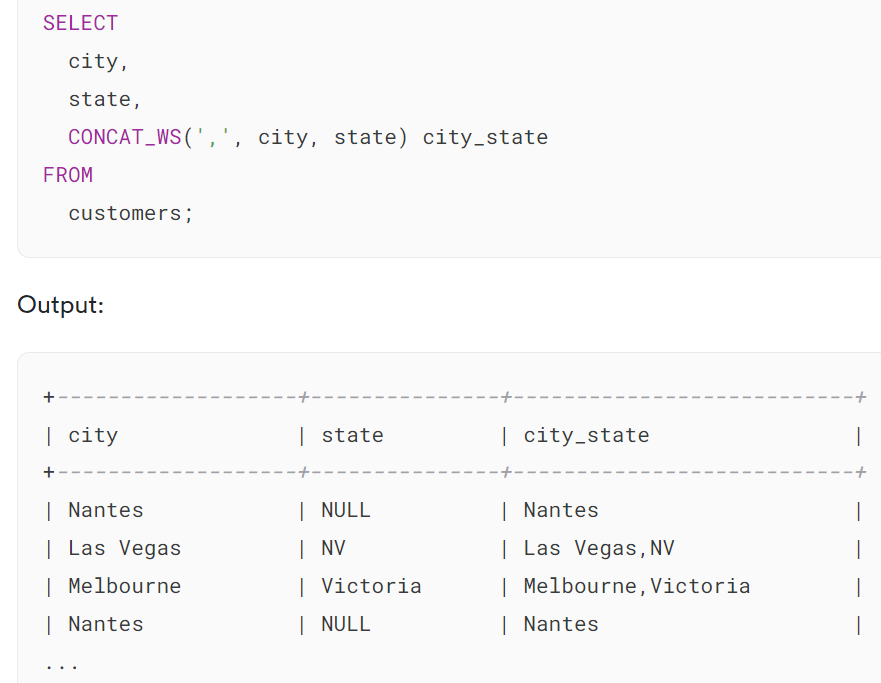


In this example, the CONCAT() takes three arguments:

* city: The first argument.
* IF(state IS NULL, '', ',') : The second argument returns space if the state is NULL or a comma (,) otherwise.
* IFNULL(state, ''): The third argument returns a space if the state is NULL or state otherwise.

The result string will be city only if the state is NULL or city, state if the state is not NULL.

The CONCAT\_WS function concatenates multiple strings into a string delimited by a specified space. It skips the NULL values. For example:



## **Summary**

* Use the MySQL CONCAT() function to concatenate multiple strings into a single string.
* The CONCAT() function returns NULL if any string argument is NULL.

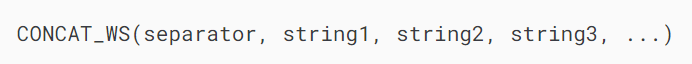
# **MySQL CONCAT\_WS() Function**

**Summary**: in this tutorial, you will learn how to use the MySQL CONCAT\_WS function to concatenate strings into a single string, separated by a specified delimiter.

## **Introduction to MySQL CONCAT\_WS function**

CONCAT\_WS stands for **Concat**enate **W**ith **S**eparator. The CONCAT\_WS function concatenates multiple strings into a single string separated by a specified separator.

Here’s the syntax of the CONCAT\_WS function:



In this syntax:

* separator: This is a separator that you use to separate the strings.
* string1, string2, string3, ..: The strings that you want to concatenate.

The CONCAT\_WS returns a single string that combines the string1, string2, string3… separated by the separator.

If the separator is NULL, the CONCAT\_WS will return NULL. The CONCAT\_WS function does not skip empty strings. But if does skip any NULL strings (string1, string2, string3…).

In practice, you use the CONCAT\_WS function to combine values from different columns with a custom separator.

## **MySQL CONCAT\_WS function examples**

Let’s take some examples of using the CONCAT\_WS() function.

### **1) Simple CONCAT\_WS function example**

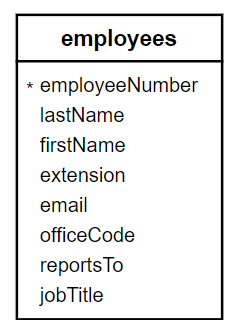
The following example uses the CONCAT\_WS() function to concatenate two strings with a comma:



In this example, we use the CONCAT\_WS function to combine the strings 'John' and 'Doe' with a comma separator. The result is the string 'John,Doe'.

### **2) Using the CONCAT\_WS with the table data**

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



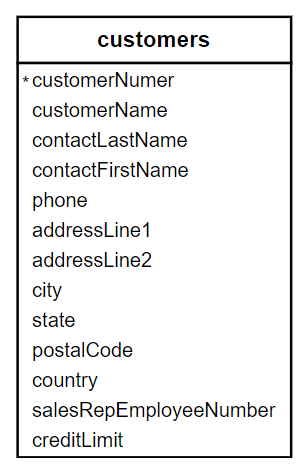
The following example uses the CONCAT\_WS to concatenate values from the firstName and lastName columns of the employees table using a space as a separator:



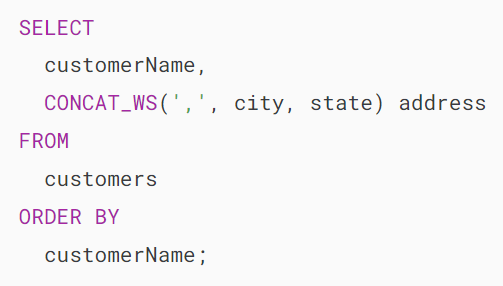
The query returns a result set with a single column full\_name containing the full names of all employees.

### **3) Using CONCAT\_WS function with NULL values**

Consider the following customers table in the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/):



The following query uses the CONCAT\_WS function to concatenate the city and state of the customers into a single string with the comma as a separator:



Output:



In this example, when the state is NULL, the CONCAT\_WS skips it in the result string.

## **Summary**

* Use the CONCAT\_WS function to concatenate multiple strings into a single string separated by a specified separator.
* The CONCAT\_WS function skips NULL values.

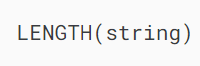
# **MySQL LENGTH() Function**

**Summary**: in this tutorial, you will learn about the MySQL LENGTH() function to get the length of strings in bytes.

## **Introduction to MySQL LENGTH() function**

The LENGTH() function returns the length of a string measured in bytes.

Here’s the basic syntax of the LENGTH() function:



The LENGTH() function takes a string argument and returns its length measured in bytes. Therefore, the result of the LENGTH() function is based on the current [character set](https://www.mysqltutorial.org/mysql-basics/mysql-character-set/) of the string.

Note that to get the number of characters of a string, you use the CHAR\_LENGTH() function instead.

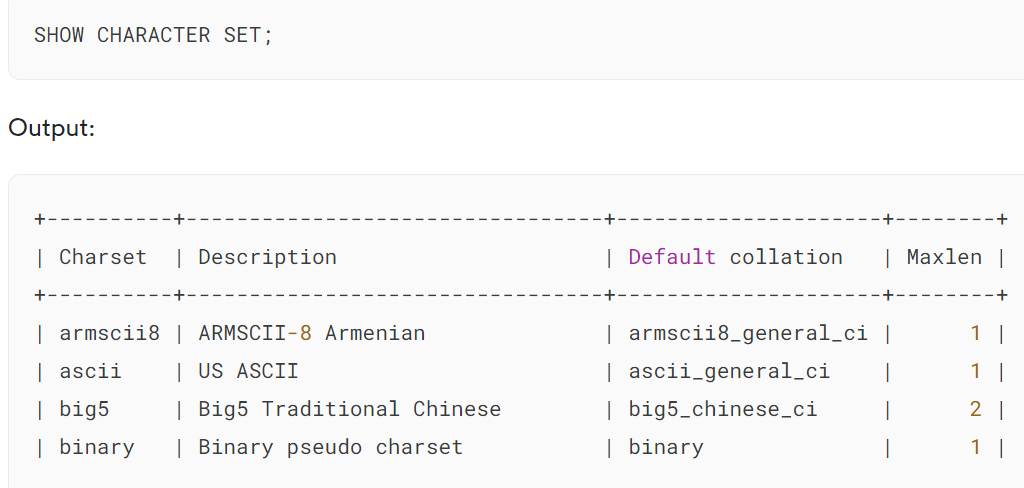
### **Understanding character sets**

Character sets define how MySQL store and represent characters in a database. MySQL supports various character sets, including single-byte and multi-byte character sets.

In a single-byte character set, MySQL represents each character using a single byte. For these character sets, the length of a string in characters is equal to its length in bytes.

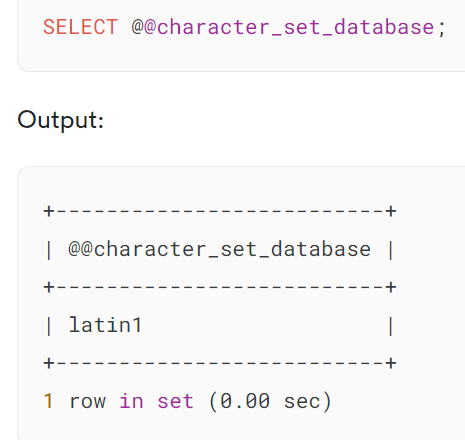
In multi-byte character sets, such as UTF-8, MySQL represents characters using multiple bytes. In such cases, the length of a string in character may be different from its length in bytes.

To find available character sets in the current database, you use the following statement:



In the output, the Maxlen column shows the maximum number of bytes for each character.

To find the character set that the current database uses, you use the @@character\_set\_database variable:



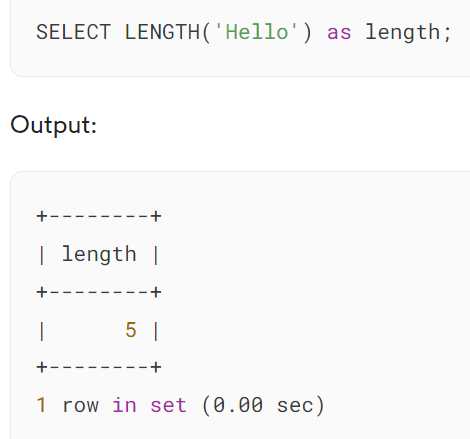
For the rest of the tutorial, we assume that you use the latin1 as the default character set.

## **MySQL LENGTH function examples**

Let’s take some examples of using the LENGTH() function.

### **1) Single-byte character set example**

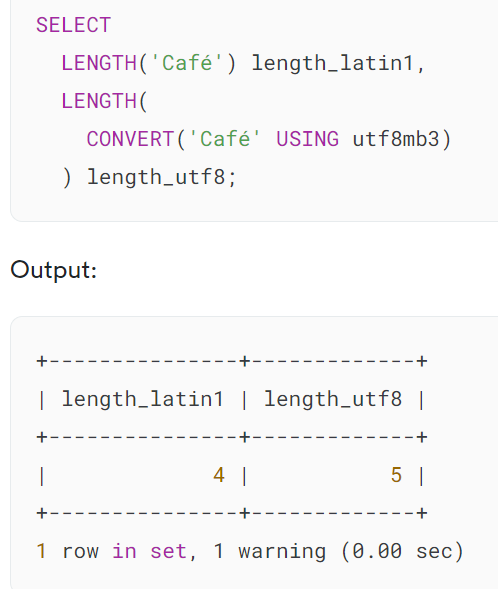
The following example uses the LENGTH() function to return the length of a string in the Latin1 character set:



The query returns 5 because the string 'Hello' contains five characters and each character is represented using a single byte in the latin1 character set.

### **2) Multi-byte character set example**

The following example uses the LENGTH() to get the length of the string Café in latin1 and utf8mb3 character sets:



In this example:

* The first LENGTH() function returns 4 because latin1 represents each character using 1 byte.
* The second LENGTH() function returns 5 because utf8mb3 uses one byte to represent the first three characters (caf) two bytes to represent the last character (é).

## **Summary**

* Use the MySQL LENGTH() function to get the length of a string in bytes.

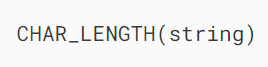
# **MySQL CHAR\_LENGTH() Function**

**Summary**: in this tutorial, you will learn how to use the MySQL CHAR\_LENGTH() function to get the number of characters in a string.

## **Introduction to the MySQL CHAR\_LENGTH function**

The CHAR\_LENGTH() function returns the number of characters in a string regardless of character set used.

Here’s the basic syntax of the CHAR\_LENGTH() function:



In this syntax, the string is the input string which you want to calculate the character length.

If the string is NULL, the CHAR\_LENGTH() function returns NULL.

Unlike the [LENGTH()](https://www.mysqltutorial.org/mysql-string-functions/mysql-string-length/) function that returns the length of a string **in bytes**, the CHAR\_LENGTH() returns the length of the string **in characters**.

Therefore, the CHAR\_LENGTH() function can be useful when you work with multibyte character sets like UTF-8.

## **MySQL CHAR\_LENGTH function examples**

Let’s take some examples of using the CHAR\_LENGTH() function.

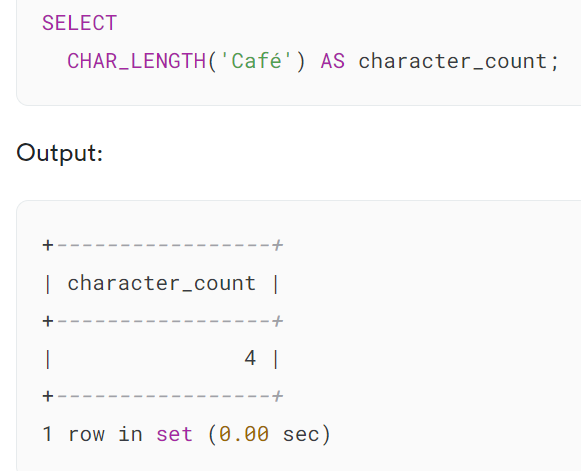
### **1) Using CHAR\_LENGTH with single-byte character set string example**

A single-byte character set represents each character by a single byte. Therefore, the CHAR\_LENGTH() function will return the same result as the LENGTH() function, as each byte represents one character:

### 

### **2) Using CHAR\_LENGTH with multibyte character set string example**

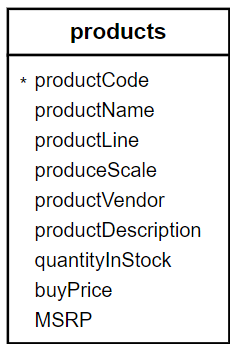
A multibyte character set may use multiple bytes to represent each character. However, the CHAR\_LENGTH() will count the number characters accurately in such situations. For example:



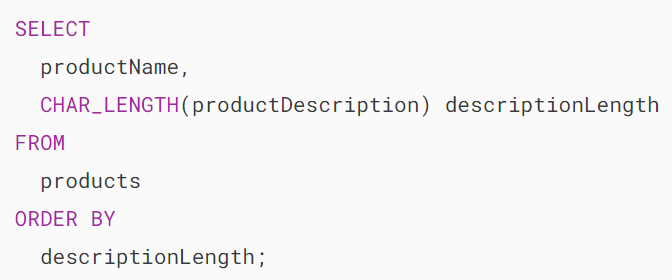
The query returns 4 because the string Café contains four characters. Even though the 'é' character is represented by two bytes in UTF-8, the CHAR\_LENGTH() function counts it as a single character.

### **3) Using CHAR\_LENGTH function with table data example**

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



The following example uses the CHAR\_LENGTH() function to return the character counts of the product descriptions:



Output:

## 

## **Summary**

* Use MySQL CHAR\_LENGTH() function to count the number of characters in a string, regardless of the character set is currently being used.

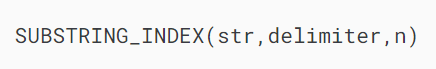
# **MySQL SUBSTRING\_INDEX Function**

**Summary**: in this tutorial, you will learn how to use the MySQL SUBSTRING\_INDEX() function to get a substring from a string before a specified number of occurrences of the delimiter.

## **MySQL SUBSTRING\_INDEX() function overview**

The SUBSTRING\_INDEX() function returns a substring from a string before a specified number of occurrences of the delimiter.

Here is the syntax of the SUBSTRING\_INDEX() function:



In this syntax:

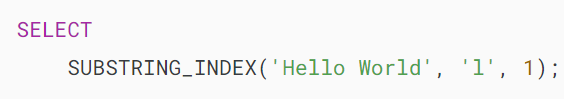
* str is the string from which you want to extract a substring.
* delimiter is a string that acts as a delimiter. The function performs a case-sensitive match when searching for the delimiter.
* n is an integer that specifies the number of occurrences of the delimiter. The n can be negative or positive. If n is positive, the function returns every character from the left of the string up to n number of occurrences of the delimiter. If n is negative, the function returns every character from right up to n number of occurrences of the delimiter.

## **MySQL SUBSTRING\_INDEX() function examples**

Let’s take some examples of using the SUBSTRING\_INDEX() function.

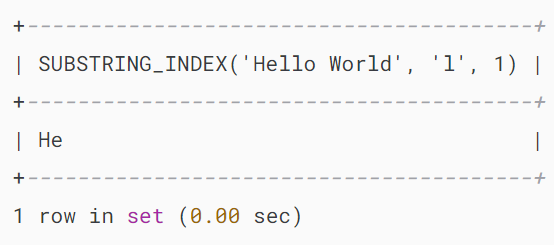
### **1) Using MySQL SUBSTRING\_INDEX() function with a positive number of occurrences of a delimiter**

See the following example:

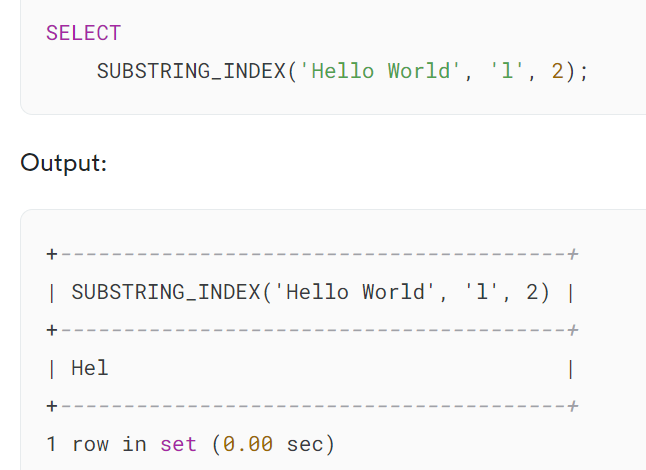


In this example, the delimiter is l and the n is 1, therefore, the function returns every character up to the 1st occurrence of the delimiterl.

Here is the output:



The following shows another example of using the SUBSTRING\_INDEX() function:



And

### 

### **2) Using SUBSTRING\_INDEX() function with a negative number of occurrences of a delimiter**

See the following example:



In this example, the delimiter is l and the n is -1, therefore, the function returns every character from the right of the string up to the 1st occurrence of the character l, (counting from the right)

Here is the output:

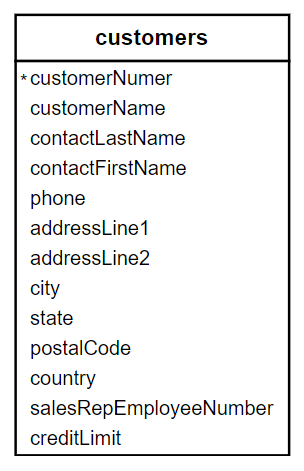


Here is another example:

### 

### **3) Using SUBSTRING\_INDEX() function with the table data example**

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



This example uses the SUBSTRING\_INDEX() function to extract the house numbers from the addresses of all customers in the USA:



The following picture shows the partial output:



In this tutorial, you have learned how to use the MySQL SUBSTRING\_INDEX() function to get a substring from a string before a specified number of occurrences of the delimiter.

<https://www.mysqltutorial.org/mysql-string-functions/mysql-replace-function/>

<https://www.mysqltutorial.org/mysql-string-functions/mysql-right-function/>

<https://www.mysqltutorial.org/mysql-string-functions/mysql-rtrim-function/>

<https://www.mysqltutorial.org/mysql-string-functions/mysql-reverse-function/>

<https://www.mysqltutorial.org/mysql-string-functions/mysql-repeat-function/>

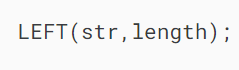
# **MySQL LEFT() Function**

**Summary**: in this tutorial, you will learn how to use the MySQL LEFT() function to return the left part of a string with a specified length.

## **Introduction to MySQL LEFT function**

In MySQL, the LEFT function allows you to extract the left part of a string with a specified length.

The following shows the syntax of the LEFT function.



The LEFT function accepts two arguments:

1. The str is the string that you want to extract the substring.
2. The length is a positive [integer](https://www.mysqltutorial.org/mysql-basics/mysql-int/) that specifies the number of characters that will be returned.

The LEFT function returns the leftmost length characters from the str string. It returns a NULL value if either str or length argument is NULL.

If the length is zero or negative, the LEFT function returns an empty string. If the length is greater than the length of the str string, the LEFT function returns the entire str string.

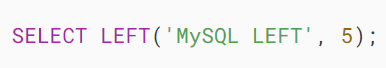
Notice that the [SUBSTRING](https://www.mysqltutorial.org/mysql-string-functions/mysql-substring/) (or [SUBSTR](https://www.mysqltutorial.org/mysql-string-functions/mysql-substring/)) function also provides the same functionality as the LEFT function.

## **MySQL LEFT() function examples**

Let’s take some examples of using the LEFT() function

### **1) Using MySQL LEFT() function with literal strings example**

The following statement uses the LEFT function to return the 5 leftmost characters of the string MySQL LEFT.



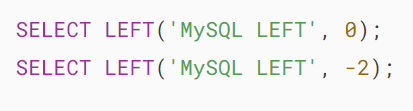
MySQL LEFT example

The following statement returns the entire string because the length exceeds the length of the string.



MySQL LEFT Function example

The following statements return an empty string because the length is zero or negative.



MySQL LEFT function returns empty string

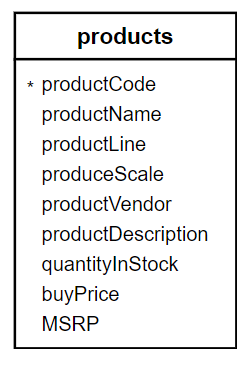
The following statement returns a NULL value because the length is NULL:



MySQL LEFT function returns NULL

### **2) Using MySQL LEFT() function with table data**

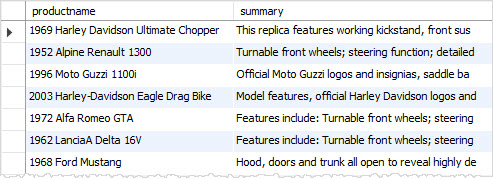
Let’s take a look at the products table in the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/):



Suppose you want to display the product name and product description in a catalog. The product description is long, therefore, you want to take just the first 50 characters for displaying.

The following statement uses the LEFT() function to return the first 50 characters of the product description.





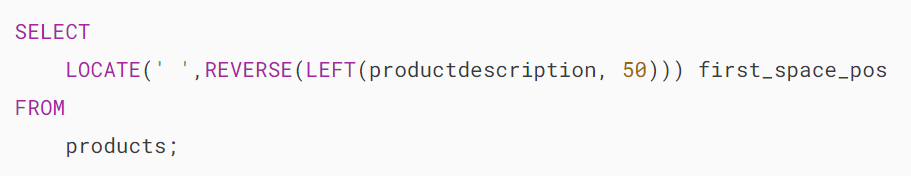
The LEFT function returns the first 50 characters. It does not care about words.

You want to get the first 50 characters without cutting the word in the middle. To achieve this, you use the following steps:

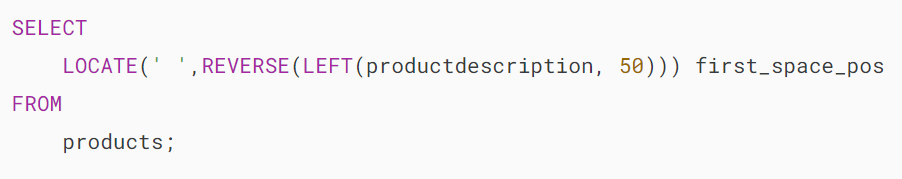
1) Take the leftmost 50 characters of the productDescription column.



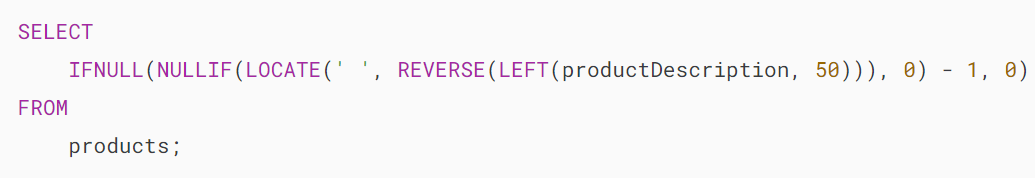
2) Reverse the substring using the REVERSE function.



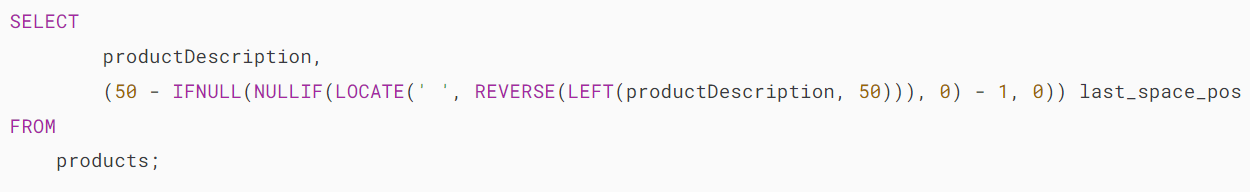
3) Get the first space’s position in the reversed substring using the LOCATE function.



4) Minus 1 from the position. In case you don’t find any space, keep the position zero.

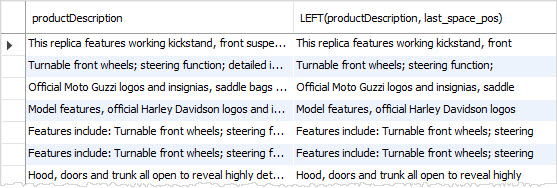


5) Minus the position from the 50, you got the position. Let’s call it as last\_space\_pos.



6) Take the leftmost last\_space\_pos characters of the product description.





## **Summary**

* Use the MySQL LEFT() function to return the left part of a string with a specified length.

# **MySQL SUBSTRING() Function**

**Summary**: in this tutorial, you will learn how to the MySQL SUBSTRING()function that extracts a substring from a string.

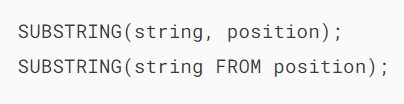
## **Introduction to MySQL SUBSTRING() function**

The SUBSTRING() function allows you to extract a substring from a string. MySQL provides various forms of the substring function.

We will examine each form of the SUBSTRING function in the following sections.

### **1) SUBSTRING() function with position parameter**

The following illustrates the first form the SUBSTRING() function:



In this syntax, the SUBSTRING() function has two parameters:

* string. The string from which you want to extract the substring.
* position. This is an [integer](https://www.mysqltutorial.org/mysql-basics/mysql-int/) that specifies the starting character of the substring. The position can be a positive or negative integer.

If the position is positive, the SUBSTRING() function extracts the substring from the start of the string. For example:

MySQL substring - string demo

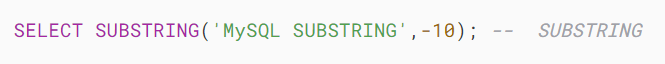
For example, to extract the ” SUBSTRING” from the ” MySQL SUBSTRING” string, the starting position of the substring must be 7 as the following [SELECT](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) statement:



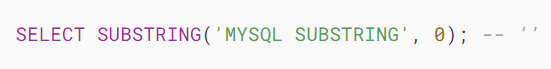
If the position is negative, the SUBSTRING() function extracts the substring from the end of the string. See the following ” MYSQL SUBSTRING” string:

MySQLsubstring function with negative position

To get the ” SUBSTRING” out of the ” MySQL SUBSTRING” using a negative position, you pass -10 to the position argument as follows:



Notice that if the position is zero, the SUBSTRING() function returns an empty string:



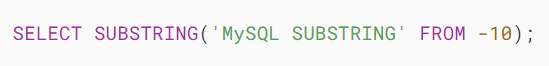
Besides the MySQL-specific syntax, you can use SQL-standard syntax with the FROM keyword to call the SUBSTRING function.

For example, the following statement gets the SUBSTRING from the MySQL SUBSTRING string using the SQL-standard syntax:

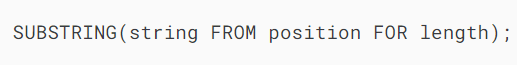
### 

### **2) SUBSTRING function with position and length arguments**

If you want to specify the length of the substring that you want to extract from a string, you can use the following form of the SUBSTRING function:



The following is the SQL-standard version of the above statement, which is longer but more expressive.



Besides the string and position arguments, the SUBSTRING() function has an additional length argument.

The length can be a positive integer that specifies the number of characters of the substring.

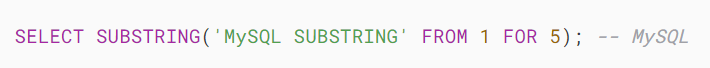
If the sum of position and length is greater than the number of characters of the string, the SUBSTRING() function returns a substring starting from the position to the end of the string.

For example, to get the ” MySQL” from the “MySQL SUBSTRING“, you use the following statement:

MySQL substring - string demo

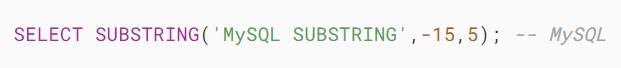


Or



If you want to use the negative position, you use the following statement:

MySQLsubstring function with negative position



Or with the FROM FOR syntax:



The SUBSTR() is the synonym for the SUBSTRING() so you can use both of them interchangeably.

## **Summary**

* Use the MySQL SUBSTRING() function to extract a substring from a string.