SQL Server CHARINDEX() function overview

SQL Server CHARINDEX() function searches for a substring inside a string starting from a specified location. It returns the position of the substring found in the searched string, or zero if the substring is not found. The starting position returned is 1-based, not 0-based.

The following shows the syntax of the CHARINDEX() function:

CHARINDEX(substring, string [, start\_location])

Code language: SQL (Structured Query Language) (sql)

In this syntax:

* substring is the substring to search for. Its length is limited to 8,000 characters.
* string can be a literal string, expression or column. It is a string to search.
* start\_location is the location at which the search starts. The start\_location is an integer, big integer or an expression that evaluates to a value of those [data types](https://www.sqlservertutorial.net/sql-server-basics/sql-server-data-types/).

The start\_location parameter is optional. If it is skipped, zero, or negative value, the search starts at the beginning of the string.

Note that the CHARINDEX() function can perform both case-sensitive and case-insensitive searches based on the specified collation.

SQL Server CHARINDEX() function examples

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

A) Using CHARINDEX() to perform a single search

The following example uses the CHARINDEX() function to perform a simple search of the string 'SQL' in the 'SQL Server CHARINDEX'

SELECT

CHARINDEX('SQL', 'SQL Server CHARINDEX') position;

Code language: SQL (Structured Query Language) (sql)

Here is the output:

position

-----------

1

(1 row affected)

B) Using CHARINDEX() function to perform a case-insensitive search

This statement shows a case-insensitive search for the string 'SERVER' in 'SQL Server CHARINDEX':

SELECT

CHARINDEX(

'SERVER',

'SQL Server CHARINDEX'

) position;

Code language: SQL (Structured Query Language) (sql)

Here is the output:

position

-----------

5

(1 row affected)

C) Using CHARINDEX() function to perform a case-sensitive search

The following example shows a case-sensitive search for the string 'SERVER' in searched string SQL Server CHARINDEX.

SELECT

CHARINDEX(

'SERVER',

'SQL Server CHARINDEX'

COLLATE Latin1\_General\_CS\_AS

) position;

Code language: SQL (Structured Query Language) (sql)

Here is the output:

position

-----------

0

(1 row affected)

D) Using CHARINDEX() function to search for a nonexistent substring

The following example illustrates a search for the substring 'needle' in the string 'This is a haystack'

DECLARE @haystack VARCHAR(100);

SELECT @haystack = 'This is a haystack';

SELECT CHARINDEX('needle', @haystack);

Code language: SQL (Structured Query Language) (sql)

The output is:

position

-----------

0

(1 row affected)

Code language: SQL (Structured Query Language) (sql)

E) Using CHARINDEX() function to searching from a specific position

This example uses the start\_location parameter to start the search for 'is' at the fifth and tenth character of the string 'This is a my sister':

SELECT

CHARINDEX('is','This is a my sister',5) start\_at\_fifth,

CHARINDEX('is','This is a my sister',10) start\_at\_tenth;

Code language: SQL (Structured Query Language) (sql)

Here is the output:

start\_at\_fifth start\_at\_tenth

-------------- --------------

6 15

(1 row affected)

## Overview of SQL Server CONCAT\_WS() function

The SQL Server CONCAT\_WS() function concatenates two or more strings into one string with a separator. CONCAT\_WS() means concatenate with separator.

The following shows the syntax of the CONCAT\_WS() function:

CONCAT\_WS(separator,input\_string1,input\_string2,[...input\_stringN]);

Code language: SQL (Structured Query Language) (sql)

In this syntax:

separator is a character-based expression that evaluates to any character of the type CHAR, NCHAR, VARCHAR, or NVARCHAR.

input\_string1 to input\_stringN are expressions of any type. The CONCAT\_WS() function implicitly converts values of non-character type to character type before concatenation.

The CONCAT\_WS() function joins the input strings into a single string. It separates those concatenated strings with the separator specified in the first argument.

Note that the CONCAT\_WS() requires at least two input strings. It means that if pass zero or one input string argument, the function will raise an error.

The CONCAT\_WS() function treats NULL as an empty string of type VARCHAR(1). It also does not add the separator between NULLs. Therefore, the CONCAT\_WS() function can cleanly join strings that may have blank values.

## SQL Server CONCAT\_WS() function examples

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

### A) Using CONCAT\_WS() to join literal strings with a separator

The following example uses the CONCAT\_WS() function to join two literal strings into one. It separates two concatenated value using a space:

SELECT

CONCAT\_WS(' ', 'John', 'Doe') full\_name

Code language: SQL (Structured Query Language) (sql)

Here is the output:

full\_name

---------

John Doe

(1 row affected)

### B) Using CONCAT\_WS() with table columns

The following statement uses the CONCAT\_WS() function to join values in the last\_name and first\_name columns of the sales.customers table. It separates last name and first name by a comma (,) and space:

SELECT

first\_name,

last\_name,

CONCAT\_WS(', ', last\_name, first\_name) full\_name

FROM

sales.customers

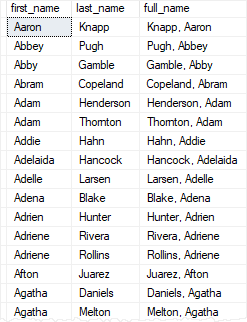
ORDER BY

first\_name,

last\_name;

Code language: SQL (Structured Query Language) (sql)

The following picture shows the partial output:



### C) Using CONCAT\_WS() with NULL

The following statement demonstrates how the CONCAT\_WS() function handles input strings that have NULL values:

SELECT

CONCAT\_WS(',', 1, 2, NULL, NULL, 3);

Code language: SQL (Structured Query Language) (sql)

The output is as follows:

result

----------------------------------------

1,2,3

(1 row affected)

As can be seen clearly from the output, the CONCAT\_WS() function ignores NULL and don’t add the separator between NULL values.

The following example concatenates customer data to format customer’s addresses. If a customer does not have a phone number, the function just ignores it:

SELECT

CONCAT\_WS

(

CHAR(13),

CONCAT(first\_name, ' ', last\_name),

phone,

CONCAT(city, ' ', state),

zip\_code,

'---'

) customer\_address

FROM

sales.customers

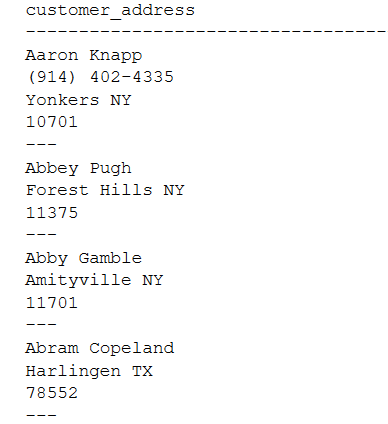
ORDER BY

first\_name,

last\_name;

Code language: SQL (Structured Query Language) (sql)

This picture illustrates the partial output:



Note that you must change the result of the query from the grid to text to see the output in the above format:

SQL Server CONCAT_WS result to text

### D) Using CONCAT\_WS() to generate CSV file

This statement uses a comma (,) as the separator and concatenates values in first\_name, last\_name, and email column to generate a CSV file:

SELECT

CONCAT\_WS(',', first\_name, last\_name, email)

FROM

sales.customers

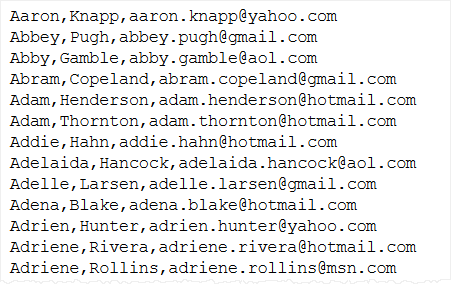
ORDER BY

first\_name,

last\_name;

Code language: SQL (Structured Query Language) (sql)

The partial output is as follows:



SQL Server LEFT() function overview

The LEFT() function extracts a given number of characters from the left side of a supplied string. For example, LEFT('SQL Server', 3) returns SQL.

The syntax of the LEFT() function is as follows:

LEFT ( input\_string , number\_of\_characters )

Code language: SQL (Structured Query Language) (sql)

In this syntax:

* The input\_string can be a literal string, variable, or column. The data type of the result of the input\_string can be any data type, except for TEXT or NTEXT, that is implicitly converted to [VARCHAR](https://www.sqlservertutorial.net/sql-server-basics/sql-server-varchar/) or [NVARCHAR](https://www.sqlservertutorial.net/sql-server-basics/sql-server-nvarchar/).
* The number\_of\_characters is a positive integer that specifies the number of characters of the input\_string will be returned.

The LEFT() function returns a value of [VARCHAR](https://www.sqlservertutorial.net/sql-server-basics/sql-server-varchar/) when the input\_string is a non-Unicode character data type or [NVARCHAR](https://www.sqlservertutorial.net/sql-server-basics/sql-server-nvarchar/) if the input\_string is a Unicode character data type.

SQL Server LEFT() function examples

Let’s take some example of using the LEFT() function to understand it better.

A) using LEFT() function with a literal character string

The following statement uses LEFT() to return the three leftmost characters of the character string SQL Server:

SELECT LEFT('SQL Server',3) Result\_string;

Code language: SQL (Structured Query Language) (sql)

Here is the output:

Result\_string

-------------

SQL

(1 row affected)

B) Using the LEFT() function with a table column

The following example returns the seven leftmost characters of each product name in the production.products table:

SELECT

product\_name,

LEFT(product\_name, 7) first\_7\_characters

FROM

production.products

ORDER BY

product\_name;

Code language: SQL (Structured Query Language) (sql)

The following picture shows the partial output:



C) Using LEFT() function with GROUP BY clause

The following example uses the LEFT() function to return a set of initials of the product name and the number of each product for each initial:

SELECT

LEFT(product\_name, 1) initial,

COUNT(product\_name) product\_count

FROM

production.products

GROUP BY

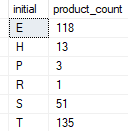
left(product\_name, 1)

ORDER BY

initial;

Code language: SQL (Structured Query Language) (sql)

Here is the output:



This query can be used for alphabetical pagination in applications.

SQL Server PATINDEX() overview

The PATINDEX() function returns the position of the first occurrence of a pattern in a string. The syntax of the PATINDEX() function is as follows:

PATINDEX ( '%pattern%' , input\_string )

Code language: SQL (Structured Query Language) (sql)

The PATINDEX() function accepts two arguments:

* pattern is a character expression to be found. It can contain wildcard characters such as % and '\_' in the pattern. The meanings of the wildcards are the same as they are used with the [LIKE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-like/)operator.
* input\_string is a character string in which the pattern to be searched.

The PATINDEX() returns an integer that specifies the position of the first occurrence of the pattern in the input\_string, or zero of the pattern not found. The PATINDEX() function will return NULL if either pattern or input\_string is NULL.

Note that the PATINDEX() searches for the pattern based on the collation of the input. If you want to use a specific collation, you can use the COLLATE clause explicitly.

SQL Server PATINDEX() function examples

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

A) SQL Server PATINDEX() simple example

This example returns the starting position of the substring 'ern' in the string 'SQL Pattern Index':

SELECT

PATINDEX('%ern%', 'SQL Pattern Index') position;

Code language: SQL (Structured Query Language) (sql)

Here is the output:

position

-----------

9

(1 row affected)

B) Using SQL Server PATINDEX() with multiple wildcards example

This example uses % and \_ wildcards to find the position at which the pattern 'f', followed by any two characters and 'ction' starts in the 'SQL Server String Function' string:

SELECT

PATINDEX('%f\_\_ction%', 'SQL Server String Function') position;

Code language: SQL (Structured Query Language) (sql)

The output is as follows:

Position

-----------

19

(1 row affected)

C) Using SQL Server PATINDEX() function with table column example

This example finds the position of the first occurrence of the pattern 2018  in values of the product\_name column in the production.products table from the [sample database](https://www.sqlservertutorial.net/sql-server-sample-database/).

SELECT

product\_name,

PATINDEX('%2018%', product\_name) position

FROM

production.products

WHERE

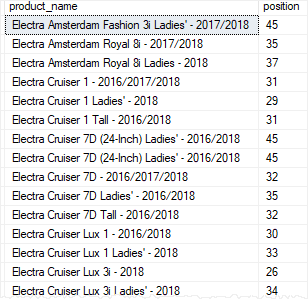
product\_name LIKE '%2018%'

ORDER BY

product\_name;

Code language: SQL (Structured Query Language) (sql)

The following picture shows the partial output:



SQL Server REPLICATE() function overview

The REPLICATE() function repeats a string a specified number of times. Its syntax is straightforward as follows:

REPLICATE(input\_string, count);

Code language: SQL (Structured Query Language) (sql)

In this syntax:

* input\_string is an expression that evaluates to a value of the character or binary type.
* count is the number of times that the input\_string will be repeated in the result string.

SQL Server REPLICATE() function examples

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

A) Using REPLICATE() function to repeat a literal string

This example uses the REPLICATE() function to repeat the character z three times:

SELECT

REPLICATE('z',3) result;

Code language: SQL (Structured Query Language) (sql)

Here is the output:

result

------

zzz

(1 row affected)

B) Using REPLICATE() function making columns with leading zero data

First, [create a new table](https://www.sqlservertutorial.net/sql-server-basics/sql-server-create-table/) named spare\_parts in the production schema:

CREATE TABLE production.spare\_parts (

part\_id INT IDENTITY PRIMARY KEY,

part\_no VARCHAR(10) NOT NULL UNIQUE,

description VARCHAR(50) NOT NULL

);

Code language: SQL (Structured Query Language) (sql)

Suppose the part\_no must always have 10 characters. If a spare part has less than 10 characters, the application needs to pad leading zeros so that the length of the spare part always has 10 characters.

Second, [insert](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert/) some values into the production.spare\_parts table:

INSERT INTO

production.spare\_parts(part\_no, description)

VALUES

('FRMTUBE','Frame Tube'),

('CHNCO','Chain Cover'),

('CRKS','Cranks');

Code language: SQL (Structured Query Language) (sql)

Third, select data from the production.spare\_parts table, left pads zeros using the REPLICATE() function:

SELECT

part\_id,

CONCAT(

REPLICATE('0', 10 - LEN(part\_no)),

part\_no

) part\_no,

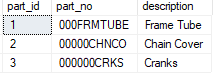
description

FROM

production.spare\_parts;

Code language: SQL (Structured Query Language) (sql)

Here is the output:



In this example:

First, use the [LEN()](https://www.sqlservertutorial.net/sql-server-string-functions/sql-server-len-function/) function to calculate the number of zeros to be padded:

10 - LEN(part\_no)

Code language: SQL (Structured Query Language) (sql)

Second, use the REPLICATE() function to replicate the necessary zeros to be padded:

REPLICATE('0', 10 - LEN(part\_no)

Code language: SQL (Structured Query Language) (sql)

Third, use the [CONCAT()](https://www.sqlservertutorial.net/sql-server-string-functions/sql-server-concat-function/) function to left pad the zeros to the part no:

CONCAT(

REPLICATE('0', 10 - LEN(part\_no)),

part\_no

) part\_no

Code language: SQL (Structured Query Language) (sql)

SQL Server RIGHT() function overview

The RIGHT() function extracts a given number of characters from the right side of a specified character string. For example, RIGHT('SQL Server', 6) returns Server.

The syntax of the RIGHT() function is as follows:

RIGHT ( input\_string , number\_of\_characters )

Code language: SQL (Structured Query Language) (sql)

In this syntax:

* The input\_string can be a literal string, variable, or column. The result of the input\_string can be in any data type, except for TEXT or NTEXT, that is implicitly converted to [VARCHAR](https://www.sqlservertutorial.net/sql-server-basics/sql-server-varchar/) or [NVARCHAR](https://www.sqlservertutorial.net/sql-server-basics/sql-server-nvarchar/).
* The number\_of\_characters is a positive integer that specifies the number of characters of the input\_string will be returned.

Note that the RIGHT() function returns a value of VARCHAR when the input\_string is a non-Unicode character data type or NVARCHAR if the input\_string is a Unicode character data type.

SQL Server RIGHT() function examples

The following statement uses RIGHT() to return the three rightmost characters of the character string SQL Server:

SELECT RIGHT('SQL Server',6) Result\_string;

Code language: SQL (Structured Query Language) (sql)

Here is the output:

Result\_string

-------------

Server

(1 row affected)

The following example returns the four rightmost characters of each product name in the production.products table from the [sample database](https://www.sqlservertutorial.net/sql-server-sample-database/):

SELECT

product\_name,

RIGHT(product\_name, 4) last\_4\_characters

FROM

production.products

ORDER BY

product\_name;

Code language: SQL (Structured Query Language) (sql)

Here is the partial output:

