**STRING FUNCTIONS**

**ASCII() Function**: The ASCII() function returns the ASCII value for the specific character.

## **Syntax:** ASCII(character)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Character | Required. The character to return the ASCII value for. If more than one character is entered, it will only return the value for the first character |

### **Example**

Return the ASCII value of the first character in "CustomerName":

SELECT ASCII(CustomerName) AS NumCodeOfFirstChar  
FROM Customers;

**CHAR() Function:** The CHAR() function returns the character based on the ASCII code.

## **Syntax:** CHAR(code)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Code | Required. The ASCII number code to return the character for |

### **Example**

Return the character based on the number code 65:

SELECT CHAR(65) AS CodeToCharacter;

# CHARINDEX() Function: The CHARINDEX() function searches for a substring in a string, and returns the position. If the substring is not found, this function returns 0.

**Note:** This function performs a case-insensitive search.

## **Syntax:** CHARINDEX(substring, string, start)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Substring | Required. The substring to search for |
| String | Required. The string to be searched |
| Start | Optional. The position where the search will start (if you do not want to start at the beginning of string). The first position in string is 1 |

### **Example**

Search for "t" in string "Customer", and return position:

SELECT CHARINDEX('t', 'Customer') AS MatchPosition;

Search for "OM" in string "Customer", and return position:

SELECT CHARINDEX('OM', 'Customer') AS MatchPosition;

Search for "mer" in string "Customer", and return position (start in position 3):

SELECT CHARINDEX('mer', 'Customer', 3) AS MatchPosition;

# CONCAT() Function: The CONCAT() function adds two or more strings together.

## **Syntax:** CONCAT(string1, string2, ...., string\_n)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string1, string2, string\_n | Required. The strings to add together |

### **Example**

Add 3 strings together:

SELECT CONCAT('SQL', ' is', ' fun!');

Add strings together (separate each string with a space character):

SELECT CONCAT('SQL', ' ', 'is', ' ', 'fun!');

# Concat With +: The + operator allows you to add two or more strings together.

## **Syntax:** string1 + string2 + string\_n

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| string1, string2, string\_n | Required. The strings to add together |

### **Example**

Add 3 strings together:

SELECT 'SQL' + ' is' + ' fun!';

Add strings together (separate each string with a space character):

SELECT 'SQL' + ' ' + 'is' + ' ' + 'fun!';

# CONCAT\_WS() Function: The CONCAT\_WS() function adds two or more strings together with a separator.

## **Syntax:** CONCAT\_WS(separator, string1, string2, ...., string\_n)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Separator | Required. The separator to use |
| string1, string2, string\_n | Required. The strings to add together |

### **Example**

Add strings together. Use '.' to separate the concatenated string values:

SELECT CONCAT\_WS('.', 'www', 'W3Schools', 'com');

Add strings together. Use '-' to separate the concatenated string values:

SELECT CONCAT\_WS('-', 'SQL', ' is', ' fun!');

# DATALENGTH() Function: The DATALENGTH() function returns the number of bytes used to represent an expression.

**Note:** The DATALENGTH() function counts both leading and trailing spaces when calculating the length of the expression.

## **Syntax:** DATALENGTH(expression)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Expression | Required. The data type to return the length for. If expression is NULL, it returns NULL |

### **Example**

Return the length of an expression (in bytes) (counts both leading and trailing spaces):

SELECT DATALENGTH('   W3Schools.com   ');

Return the length of an expression (in bytes):

SELECT DATALENGTH('2017-08');

# LEFT() Function: The LEFT() function extracts a number of characters from a string (starting from left).

## **Syntax:** LEFT(string, number\_of\_chars)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| String | Required. The string to extract from |
| number\_of\_chars | Required. The number of characters to extract. If the number exceeds the number of characters in string, it returns string |

### **Example**

Extract 3 characters from a string (starting from left):

SELECT LEFT('SQL Tutorial', 3) AS ExtractString;

Extract 5 characters from the text in the "CustomerName" column (starting from left):

SELECT LEFT(CustomerName, 5) AS ExtractString  
FROM Customers;

# LEN() Function: The LEN() function returns the length of a string.

**Note:** Trailing spaces at the end of the string is not included when calculating the length. However, leading spaces at the start of the string is included when calculating the length.

## **Syntax:** LEN(string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| String | Required. The string to return the length for. If string is NULL, it returns NULL |

### **Example**

Return the length of a string:

SELECT LEN('W3Schools.com');

Return the length of a string (counts leading spaces, but not trailing spaces):

SELECT LEN(' W3Schools.com ');

# LOWER() Function: The LOWER() function converts a string to lower-case.

## **Syntax:** LOWER(text)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Text | Required. The string to convert |

### **Example**

Convert the text to lower-case:

SELECT LOWER('SQL Tutorial is FUN!');

Convert the text in "CustomerName" to lower-case:

SELECT LOWER(CustomerName) AS LowercaseCustomerName  
FROM Customers;

# LTRIM() Function: The LTRIM() function removes leading spaces from a string.

### **Example**

Remove leading spaces from a string:

SELECT LTRIM('     SQL Tutorial') AS LeftTrimmedString;

## **Syntax:** LTRIM(string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| String | Required. The string to remove leading spaces from |

# PATINDEX() Function: The PATINDEX() function returns the position of a pattern in a string.

If the pattern is not found, this function returns 0.

**Note:** The search is case-insensitive and the first position in *string* is 1.

### **Example**

Return the position of a pattern in a string:

SELECT PATINDEX('%schools%', 'W3Schools.com');

## **Syntax:** PATINDEX(%pattern%, string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| %pattern% | Required. The pattern to find. It MUST be surrounded by %. Other wildcards can be used in pattern, such as:   * % - Match any string of any length (including 0 length) * \_ - Match one single character * [] - Match any characters in the brackets, e.g. [xyz] * [^] - Match any character not in the brackets, e.g. [^xyz] |
| String | Required. The string to be searched |

### **Example**

Return the position of a pattern in a string:

SELECT PATINDEX('%s%com%', 'W3Schools.com');

Return the position of a pattern in a string:

SELECT PATINDEX('%[ol]%', 'W3Schools.com');

# REPLACE() Function: The REPLACE() function replaces all occurrences of a substring within a string, with a new substring.

## **Syntax:** REPLACE(string, old\_string, new\_string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| String | Required. The original string |
| old\_string | Required. The string to be replaced |
| new\_string | Required. The new replacement string |

### **Example**

Replace "T" with "M":

SELECT REPLACE('SQL Tutorial', 'T', 'M');

Replace "SQL" with "HTML":

SELECT REPLACE('SQL Tutorial', 'SQL', 'HTML');

Replace "a" with "c":

SELECT REPLACE('ABC ABC ABC', 'a', 'c');

# REPLICATE() Function: The REPLICATE() function repeats a string a specified number of times.

## **Syntax:** REPLICATE(string, integer)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| String | Required. The string to repeat |
| Integer | Required. The number of times to repeat the string |

### **Example**

Repeat a string:

SELECT REPLICATE('SQL Tutorial', 5);

Repeat the text in CustomerName two times:

SELECT REPLICATE(CustomerName, 2)  
FROM Customers;

# REVERSE() Function: The REVERSE() function reverses a string and returns the result.

## **Syntax:** REVERSE(string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| String | Required. The string to reverse |

### **Example**

Reverse the text in CustomerName:

SELECT REVERSE(CustomerName)  
FROM Customers;

# RIGHT() Function: The RIGHT() function extracts a number of characters from a string (starting from right).

## **Syntax:** RIGHT(string, number\_of\_chars)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| String | Required. The string to extract from |
| number\_of\_chars | Required. The number of characters to extract. If number\_of\_chars > string, it returns string |

### **Example**

Extract 3 characters from a string (starting from right):

SELECT RIGHT('SQL Tutorial', 3) AS ExtractString;

Extract 5 characters from the text in the "CustomerName" column (starting from right):

SELECT RIGHT(CustomerName, 5) AS ExtractString  
FROM Customers;

# RTRIM() Function: The RTRIM() function removes trailing spaces from a string.

## **Syntax:** RTRIM(string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| String | Required. The string to remove trailing spaces from |

### **Example**

Remove trailing spaces from a string:

SELECT RTRIM('SQL Tutorial     ') AS RightTrimmedString;

# SPACE() Function : The SPACE() function returns a string of the specified number of space characters.

**Syntax:** SPACE(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Number | Required. The number of spaces to be returned |

### **Example**

Return a string with 10 spaces:

SELECT SPACE(10);

# SUBSTRING() Function: The SUBSTRING() function extracts some characters from a string.

## **Syntax:** SUBSTRING(string, start, length)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| String | Required. The string to extract from |
| Start | Required. The start position. The first position in string is 1 |
| Length | Required. The number of characters to extract. Must be a positive number |

### **Example**

Extract 3 characters from a string, starting in position 1:

SELECT SUBSTRING('SQL Tutorial', 1, 3) AS ExtractString;

Extract 5 characters from the "CustomerName" column, starting in position 1:

SELECT SUBSTRING(CustomerName, 1, 5) AS ExtractString  
FROM Customers;

# TRIM() Function: The TRIM() function removes the space character OR other specified characters from the start or end of a string.

By default, the TRIM() function removes leading and trailing spaces from a string.

## **Syntax:** TRIM([characters FROM ]string)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| characters FROM | Optional. Specific characters to remove |
| String | Required. The string to remove spaces or characters from |

### **Example**

Remove leading and trailing spaces from a string:

SELECT TRIM('     SQL Tutorial!     ') AS TrimmedString;

Remove characters and spaces from a string:

SELECT TRIM('#! ' FROM '    #SQL Tutorial!    ') AS TrimmedString;

# UPPER() Function: The UPPER() function converts a string to upper-case.

## **Syntax:** UPPER(text)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Text | Required. The string to convert |

### **Example**

Convert the text to upper-case:

SELECT UPPER('SQL Tutorial is FUN!');

Convert the text in "CustomerName" to upper-case:

SELECT UPPER(CustomerName) AS UppercaseCustomerName  
FROM Customers;

**NUMERIC FUNCTIONS**

# ABS() Function: The ABS() function returns the absolute value of a number.

## **Syntax:** ABS(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Number | Required. A numeric value |

### **Example**

Return the absolute value of a number:

SELECT Abs(-243.5) AS AbsNum;

# ACOS() Function, ASIN() Function, ATAN() Function:

# For ACOS() and ASIN() The specified number must be between -1 to 1, otherwise this function returns NULL.

### **Example**

Return the arc cosine of a number:

SELECT ACOS(0.25);

Return the arc cosine of a number:

SELECT ACOS(-0.8);

Return the arc sine of a number:

SELECT ASIN(0.25);

Return the arc sine of a number:

SELECT ASIN(-0.8);

Return the arc tangent of a number:

SELECT ATAN(2.5);

Return the arc tangent of a number:

SELECT ATAN(-45.01);

# AVG() Function: The AVG() function returns the average value of an expression.

**Note:** NULL values are ignored.

## **Syntax:** AVG(expression)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Expression | Required. A numeric value (can be a field or a formula) |

### **Example**

Return the average value for the "Price" column in the "Products" table:

SELECT AVG(Price) AS AveragePrice FROM Products;

Select all the products that have a price above the average price:

SELECT \* FROM Products  
WHERE Price > (SELECT AVG(Price) FROM Products);

# CEILING() Function: The CEILING() function returns the smallest integer value that is larger than or equal to a number.

## **Syntax:** CEILING(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Number | Required. A numeric value |

### **Example**

Return the smallest integer value that is greater than or equal to a number:

SELECT CEILING(25.75) AS CeilValue;

Return the smallest integer value that is greater than or equal to a number:

SELECT CEILING(-13.5) AS CeilValue;

# COUNT() Function: The COUNT() function returns the number of records returned by a select query.

**Note:** NULL values are not counted.

## **Syntax:** COUNT(expression)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| expression | Required. A field or a string value |

### **Example**

Return the number of products in the "Products" table:

SELECT COUNT(ProductID) AS NumberOfProducts FROM Products;

# COS() Function, COT() Function, SIN() Function, TAN() Function:

### **Example**

Return the cosine of a number:

SELECT COS(2);

Return the cosine of a number:

SELECT COS(PI());

# DEGREES() Function: The DEGREES() function converts a value in radians to degrees.

## **Syntax:** DEGREES(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Number | Required. A numeric value |

### **Example**

Convert a radian value into degrees:

SELECT DEGREES(1.5);

Convert a radian value into degrees:

SELECT DEGREES(PI()\*2);

# EXP() Function: The EXP() function returns e raised to the power of a specified number.

The constant e (2.718281...), is the base of natural logarithms.

## **Syntax:** EXP(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Number | Required. The power number |

### **Example**

Return e raised to the power of 1:

SELECT EXP(1);

Return *e* raised to the power of 2:

SELECT EXP(2);

# FLOOR() Function: The FLOOR() function returns the largest integer value that is smaller than or equal to a number.

## **Syntax:** FLOOR(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Number | Required. A numeric value |

### **Example**

Return the largest integer value that is equal to or less than 25.75:

SELECT FLOOR(25.75) AS FloorValue;

Return the largest integer value that is equal to or less than -13.5:

SELECT FLOOR(-13.5) AS FloorValue;

# LOG() Function: The LOG() function returns the natural logarithm of a specified *number*, or the logarithm of the *number* to the specified *base*.

From SQL Server 2012, you can also change the base of the logarithm to another value by using the optional base parameter.

## **Syntax:** LOG(number, base) -- Syntax for SQL Server

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Number | Required. The number to return the natural logarithm of. Must be greater than 0 |
| Base | Optional. The base the natural logarithm is to be calculated with. Must be greater than 1 |

### **Example**

Return the natural logarithm of 2:

SELECT LOG(2);

Return the natural logarithm of 2 to a specified base (4):

SELECT LOG(2, 4);

# LOG10() Function: The LOG10() function returns the natural logarithm of a number to base 10.

## **Syntax:** LOG10(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Number | Required. A number greater than 0 |

### **Example**

Return the base-10 logarithm of 2:

SELECT LOG10(2);

Return the base-10 logarithm of 4.5:

SELECT LOG10(4.5);

# MAX() and MIN() Function: The MAX() function returns the maximum value in a set of values.

## **Syntax:** MAX(expression)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Expression | Required. A numeric value (can be a field or a formula) |

### **Example**

Find the price of the most expensive product in the "Products" table:

SELECT MAX(Price) AS LargestPrice FROM Products;

# PI() Function: The PI() function returns the value of PI.

## **Syntax:** PI()

### **Example**

Return the value of PI:

SELECT PI();

# POWER() Function: The POWER() function returns the value of a number raised to the power of another number.

# Syntax: POWER(a, b)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| A | Required. A number (the base) |
| B | Required. A number (the exponent) |

### **Example**

Return 4 raised to the second power:

SELECT POWER(4, 2);

Return 8 raised to the third power:

SELECT POWER(8, 3);

# RADIANS() Function: The RADIANS() function converts a degree value into radians.

## **Syntax:** RADIANS(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Number | Required. A number in degrees |

### **Example**

Convert a degree value into radians:

SELECT RADIANS(180);

Convert a degree value into radians:

SELECT RADIANS(-45);

# RAND() Function: The RAND() function returns a random number between 0 (inclusive) and 1 (exclusive).

## **Syntax:** RAND(seed)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Seed | Optional. If seed is specified, it returns a repeatable sequence of random numbers. If no seed is specified, it returns a completely random number |

### **Example**

Return a random decimal number (no seed value - so it returns a completely random number >= 0 and <1):

SELECT RAND();

Return a random decimal number (with seed value of 6):

SELECT RAND(6);

Return a random decimal number >= 5 and <10:

SELECT RAND()\*(10-5)+5;

Return a random number >= 5 and <=10:

SELECT FLOOR(RAND()\*(10-5+1)+5);

Return a random number >= 5 and <=10:

SELECT FLOOR(RAND()\*(10-5+1)+5);

# ROUND() Function: The ROUND() function rounds a number to a specified number of decimal places.

## **Syntax:** ROUND(number, decimals, operation)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Number | Required. The number to be rounded |
| Decimals | Required. The number of decimal places to round number to |
| Operation | Optional. If 0, it rounds the result to the number of decimal. If another value than 0, it truncates the result to the number of decimals. Default value is 0 |

### **Example**

Round the number to 2 decimal places:

SELECT ROUND(235.415, 2) AS RoundValue;

Round the number to 2 decimal places, and also use the *operation* parameter:

SELECT ROUND(235.415, 2, 1) AS RoundValue;

Round the number to -1 decimal place:

SELECT ROUND(235.415, -1) AS RoundValue;

Round the number to -1 decimal place:

SELECT ROUND(235.415, -1) AS RoundValue;

# SIGN() Function: The SIGN() function returns the sign of a number.

This function will return one of the following:

* If number > 0, it returns 1
* If number = 0, it returns 0
* If number < 0, it returns -1

## **Syntax:** SIGN(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| number | Required. The number to return the sign for |

### **Example**

Return the sign of a number:

SELECT SIGN(255.5);

Return the sign of a number:

SELECT SIGN(-12);

# SQRT() Function: The SQRT() function returns the square root of a number.

## **Syntax:** SQRT(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Number | Required. A positive number to calculate the square root  of |

### **Example**

Return the square root of a number:

SELECT SQRT(64);

Return the square root of a number:

SELECT SQRT(13);

# SQUARE() Function: The SQUARE() function returns the square of a number.

## **Syntax:** SQUARE(number)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Number | Required. A positive number to calculate the square of |

### **Example**

Return the square of a number:

SELECT SQUARE(64);

Return the square of a number:

SELECT SQUARE(13);

# SUM() Function: The SUM() function calculates the sum of a set of values.

**Note:** NULL values are ignored.

## **Syntax:** SUM(expression)

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| Expression | Required. A field or a formula |

### **Example**

Return the sum of the "Quantity" field in the "OrderDetails" table:

SELECT SUM(Quantity) AS TotalItemsOrdered FROM OrderDetails;