**SQL Functions**

[**SQL**](https://www.youtube.com/playlist?list=PLyGqUe6Oa_5Hcb_5QqXIsMZhvsm7Bo_-Z) has many built-in functions for performing processing on string or numeric data. Following is the list of all useful [**SQL**](http://www.gcreddy.com/2013/03/sql-guide.html) built-in functions:

**String Functions:**

**1.ASCII:**

It Returns the ASCII code value of the leftmost character of a character expression.

**Syntax:**ASCII ( character\_expression )

Arguments

character\_expression: Is an expression of the type char or varchar.

**2.CHAR:**

It Converts an int ASCII code to a character.

Syntax:CHAR ( integer\_expression )

**Arguments:**

integer\_expression:Is an integer from 0 through 255. NULL is returned if the integer expression is not in this range.

**3.CONCAT:**

It Returns a string that is the result of concatenating two or more string values.

Syntax:CONCAT ( string\_value1, string\_value2 [, string\_valueN ] )

**Arguments:**

string\_value:A string value to concatenate to the other values.

**4.CHARINDEX:**

It Searches an expression for another expression and returns its starting position if found.

Syntax:CHARINDEX ( expressionToFind ,expressionToSearch [ , start\_location ] )

**Arguments**

expressionToFind:Is a character expression that contains the sequence to be found. expressionToFind is limited to 8000 characters.  
expressionToSearch:Is a character expression to be searched.  
start\_location: Is an integer or bigint expression at which the search starts. If start\_location is not specified, is a negative number, or is 0, the search starts at the beginning of expressionToSearch.

**5.DIFFERENCE:**

It Returns an integer value that indicates the difference between the SOUNDEX values of two character expressions.

Syntax:DIFFERENCE ( character\_expression , character\_expression )

**Arguments:**

character\_expression:Is an alphanumeric expression of character data. character\_expression can be a constant, variable, or column.

**6.LEFT:**

It Returns the left part of a character string with the specified number of characters.

**Syntax**:LEFT ( character\_expression , integer\_expression )

**Arguments:**

**character\_expression:**  
 Is an expression of character or binary data. character\_expression can be a constant, variable, or column. character\_expression can be of any data type, except text or ntext, that can be implicitly converted to varchar or nvarchar. Otherwise, use the CAST function to explicitly convert character\_expression.

**integer\_expression:**  
 Is a positive integer that specifies how many characters of the character\_expression will be returned. If integer\_expression is negative, an error is returned. If integer\_expression is type bigint and contains a large value, character\_expression must be of a large data type such as varchar(max).The integer\_expression parameter counts a UTF-16 surrogate character as one character.

**7.LOWER:**

Returns a character expression after converting uppercase character data to lowercase.

**Syntax:**LOWER ( character\_expression )

**Arguments:**

**character\_expression:**  
 Is an expression of character or binary data. character\_expression can be a constant, variable, or column. character\_expression must be of a data type that is implicitly convertible to varchar. Otherwise, use CAST to explicitly convert character\_expression.

**8.LTRIM:**

Returns a character expression after it removes leading blanks.

**Syntax:**LTRIM ( character\_expression )

**Arguments:**

**character\_expression:**

Is an expression of character or binary data. character\_expression can be a constant, variable, or column. character\_expression must be of a data type, except text, ntext, and image, that is implicitly convertible to varchar. Otherwise, use CAST to explicitly convert character\_expression.

**9.NCHAR:**

Returns the Unicode character with the specified integer code, as defined by the Unicode standard.

**Syntax:**NCHAR ( integer\_expression )

**Arguments:**

**integer\_expression:**

When the collation of the database does not contain the supplementary character (SC) flag, this is a positive whole number from 0 through 65535 (0 through 0xFFFF). If a value outside this range is specified, NULL is returned. For more information about supplementary characters, see Collation and Unicode Support.

When the collation of the database supports the supplementary character (SC) flag, this is a positive whole number from 0 through 1114111 (0 through 0x10FFFF). If a value outside this range is specified, NULL is returned.

**10.PATINDEX:**

Returns the starting position of the first occurrence of a pattern in a specified expression, or zeros if the pattern is not found, on all valid text and character data types.

**Syntax:**PATINDEX ( '%pattern%' , expression )

Arguments:

**pattern:**  
 Is a character expression that contains the sequence to be found. Wildcard characters can be used; however, the % character must come before and follow pattern (except when you search for first or last characters). pattern is an expression of the character string data type category. pattern is limited to 8000 characters.  
**expression:**

Is an expression, typically a column that is searched for the specified pattern. expression is of the character string data type category.

**11.REPLACE:**

Replaces all occurrences of a specified string value with another string value.

**Syntax:**

REPLACE ( string\_expression , string\_pattern , string\_replacement )

**Arguments:**

**string\_expression:**

Is the string expression to be searched. string\_expression can be of a character or binary data type.

**string\_pattern:**

Is the substring to be found. string\_pattern can be of a character or binary data type. string\_pattern cannot be an empty string (''), and must not exceed the maximum number of bytes that fits on a page.

**string\_replacement:**

Is the replacement string. string\_replacement can be of a character or binary data type.

**12.REPLICATE:**

Repeats a string value a specified number of times.

**Syntax:**REPLICATE ( string\_expression ,integer\_expression )

**Arguments:**

**string\_expression:**  
Is an expression of a character string or binary data type. string\_expression can be either character or binary data.

**Note:If string\_expression is not of type varchar(max) or nvarchar(max), REPLICATE truncates the return value at 8,000 bytes. To return values greater than 8,000 bytes, string\_expression must be explicitly cast to the appropriate large-value data type.**

**13.REVERSE:**

Returns the reverse order of a string value.

**Syntax:**REVERSE ( string\_expression )

**Arguments:**

**string\_expression:**

string\_expression is an expression of a string or binary data type. string\_expression can be a constant, variable, or column of either character or binary data.

**14.RIGHT:**

Returns the right part of a character string with the specified number of characters.

**Syntax:**RIGHT ( character\_expression , integer\_expression )

**Arguments:**

**character\_expression:**

Is an expression of character or binary data. character\_expression can be a constant, variable, or column. character\_expression can be of any data type, except text or ntext, that can be implicitly converted to varchar or nvarchar. Otherwise, use the CAST function to explicitly convert character\_expression.

**integer\_expression:**

Is a positive integer that specifies how many characters of character\_expression will be returned. If integer\_expression is negative, an error is returned. If integer\_expression is type bigint and contains a large value, character\_expression must be of a large data type such as varchar(max).

**15.RTRIM**

Returns a character string after truncating all trailing blanks.

**Syntax:**

RTRIM ( character\_expression )

**Arguments:**

**character\_expression:**

Is an expression of character data. character\_expression can be a constant, variable, or column of either character or binary data.character\_expression must be of a data type that is implicitly convertible to varchar. Otherwise, use CAST to explicitly convert character\_expressio

**16.SUBSTRING**

Returns part of a character, binary, text, or image expression in SQL Server 2012.

Syntax:SUBSTRING ( expression ,start , length )

**Arguments:**

**expression:**Is a character, binary, text, ntext, or image expression.

**start:**  
 Is an integer or bigint expression that specifies where the returned characters start. If start is less than 1, the returned expression will begin at the first character that is specified in expression. In this case, the number of characters that are returned is the largest value of either the sum of start + length- 1 or 0. If start is greater than the number of characters in the value expression, a zero-length expression is returned.

**length:**  
 Is a positive integer or bigint expression that specifies how many characters of the expression will be returned. If length is negative, an error is generated and the statement is terminated. If the sum of start and length is greater than the number of characters in expression, the whole value expression beginning at start is returned.

**17.SPACE**

Returns a string of repeated spaces.

**Syntax:**SPACE ( integer\_expression )

**Arguments:**

**integer\_expression:**

Is a positive integer that indicates the number of spaces. If integer\_expression is negative, a null string is returned.

**18.STR**

Returns character data converted from numeric data.

**Syntax**

STR ( float\_expression [ , length [ , decimal ] ] )

**Arguments:**

**float\_expression:**  
 Is an expression of approximate numeric (float) data type with a decimal point.

**length:**  
 Is the total length. This includes decimal point, sign, digits, and spaces. The default is 10.

**decimal:**  
 Is the number of places to the right of the decimal point. decimal must be less than or equal to 16. If decimal is more than 16 then the result is truncated to sixteen places to the right of the decimal point.

**19.UNICODE**

Returns the integer value, as defined by the Unicode standard, for the first character of the input expression.

**Syntax:**UNICODE ( 'ncharacter\_expression' )

**Arguments:**

' ncharacter\_expression ': Is an nchar or nvarchar expression.

**20.UPPER**

Returns a character expression with lowercase character data converted to uppercase.

**Syntax:**

UPPER ( character\_expression )

**Arguments:**

**character\_expression:**

Is an expression of character data. character\_expression can be a constant, variable, or column of either character or binary data.character\_expression must be of a data type that is implicitly convertible to varchar. Otherwise, use CAST to explicitly convert character\_expression

**Aggregate Functions :**  
Aggregate functions perform a calculation on a set of values and return a single value. With the exception of COUNT, aggregate functions ignore null values. Aggregate functions are often used with the GROUP BY clause of the SELECT statement.

All aggregate functions are deterministic; they return the same value any time they are called with a given set of input values. For more information about function determinism, see Deterministic and Nondeterministic Functions.

**1.AVG:**

Returns the average of the values in a group. Null values are ignored.

**Syntax:**

AVG ( [ ALL | DISTINCT ] expression )

Ex: SELECT AVG(MAX(salary)) FROM 'emp1';

**Arguments:**

**ALL:Applies the aggregate function to all values. ALL is the default.**

**DISTINCT:**Specifies that AVG be performed only on each unique instance of a value, regardless of how many times the value occurs.

**expression:**Is an expression of the exact numeric or approximate numeric data type category, except for the bit data type. Aggregate functions and subqueries are not permitted.

**2.COUNT:**

Returns the number of items in a group.

**Syntax:**

COUNT ( { [ ALL | DISTINCT ] expression ] | \* } )

Ex:SELECT COUNT(\*) "Total" FROM employees;

**Arguments:**

**ALL:**Applies the aggregate function to all values. ALL is the default.

**DISTINCT:**Specifies that COUNT returns the number of unique nonnull values.

**expression:**Is an expression of any type except uniqueidentifier, text, image, or ntext. Aggregate functions and subqueries are not permitted.

\*:Specifies that all rows should be counted to return the total number of rows in a table. COUNT(\*) takes no parameters and cannot be used with DISTINCT. COUNT(\*) does not require an expression parameter because, by definition, it does not use information about any particular column. COUNT(\*) returns the number of rows in a specified table without eliminating duplicates. It counts each row separately, including rows that contain null values.

Important Distinct aggregates, for example AVG(DISTINCT column\_name), COUNT(DISTINCT column\_name), MAX(DISTINCT column\_name), MIN(DISTINCT column\_name), and SUM(DISTINCT column\_name), are not supported when using CUBE or ROLLUP. If used, Microsoft® SQL Server™ returns an error message and cancels the query.

**3.MIN:**

Returns the minimum value in the expression.

**Syntax:**

MIN ( [ ALL | DISTINCT ] expression )

Ex:SELECT MIN(hire\_date) "Earliest" FROM employees;

**4.MAX:**

Returns the maximum value in the expression.

**Syntax**

MAX ( [ ALL | DISTINCT ] expression )

Ex:SELECT MAX(salary) "Maximum" FROM employees;

**5.SUM:**

Returns the sum of all the values, or only the DISTINCT values, in the expression. SUM can be used with numeric columns only. Null values are ignored.

**Syntax:**

SUM ( [ ALL | DISTINCT ] expression )

Ex:SELECT SUM(salary) "Total" FROM employees;

**Date Functions:**  
**1.DATEADD:**

Returns a new datetime value based on adding an interval to the specified date.

Syntax:DATEADD ( datepart , number, date )

Arguments

datepart:Is the parameter that specifies on which part of the date to return a new value.

**2.DATEDIFF:**

Returns the number of date and time boundaries crossed between two specified dates.

**Syntax:**

DATEDIFF ( datepart , startdate , enddate )

**3.DATENAME:**

Returns a character string representing the specified datepart of the specified date.

**Syntax:**

DATENAME ( datepart , date )

**4.DATEPART:**

Returns an integer representing the specified datepart of the specified date.

**Syntax:**DATEPART ( datepart , date )

**5.GETDATE:**

Returns the current system date and time in the Microsoft® SQL Server™ standard internal format for datetime values.

**Syntax**:GETDATE ( )