Category	Function Name	Syntax	Description
Aggregate	APPROX_COUNT_DISTINCT	APPROX_COUNT_DISTINCT (expression)	This function returns the approximate number of unique non-null values in a group.
Aggregate	APPROX_PERCENTILE_CONT	APPROX_PERCENTILE_CONT (numeric_literal) WITHIN GROUP (ORDER BY order_by_expression [ASC|DESC])	This function returns an approximate interpolated value from the set of values in a group based on percentile value and sort specification.
Aggregate	APPROX_PERCENTILE_DISC	APPROX_PERCENTILE_DISC (numeric_literal) WITHIN GROUP (ORDER BY order_by_expression [ASC|DESC])	This function returns the value from the set of values in a group based on the provided percentile and sort specification.
Aggregate	AVG	AVG ([ALL | DISTINCT] expression) [OVER ([partition_by_clause] order_by_clause)]	This function returns the average of the values in a group. It ignores null values.
Aggregate	CHECKSUM_AGG	CHECKSUM_AGG ([ALL | DISTINCT] expression)	This function returns the checksum of the values in a group. CHECKSUM_AGG ignores null values. The OVER clause can follow CHECKSUM_AGG.
Aggregate	COUNT	COUNT ({ [[ALL | DISTINCT] expression] | * })	This function returns the number of items found in a group. COUNT always returns an int data type value.
Aggregate	COUNT_BIG	COUNT_BIG ([ALL] { expression | * }) [OVER ([<partition_by_clause>])]	This function returns the number of items found in a group. COUNT_BIG always returns a bigint data type value.
Aggregate	GROUPING	GROUPING (<column_expression>)	Indicates whether a specified column expression in a GROUP BY list is aggregated or not. GROUPING returns 1 for aggregated or 0 for not aggregated in the result set. GROUPING can be used only in the SELECT <select> list, HAVING, and ORDER BY clauses when GROUP BY is specified.
Aggregate	MAX	MAX ([ALL | DISTINCT] expression) [OVER (<partition_by_clause> [<order_by_clause>])]	Returns the maximum value in the expression.
Aggregate	MIN	MIN ([ALL | DISTINCT] expression) [OVER (<partition_by_clause> [<order_by_clause>])]	Returns the minimum value in the expression. May be followed by the OVER clause.

Category	Function Name	Syntax	Description
Aggregate	STDEV	STDEV ([ALL | DISTINCT] expression) [OVER (<partition_by_clause> [<order_by_clause>])]	Returns the statistical standard deviation of all values in the specified expression.
Aggregate	STDEVP	STDEVP ([ALL | DISTINCT] expression) [OVER (<partition_by_clause> [<order_by_clause>])]	Returns the statistical standard deviation for the population for all values in the specified expression.
Aggregate	SUM	SUM ([ALL | DISTINCT] expression) [OVER (<partition_by_clause> [<order_by_clause>])]	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.
Aggregate	VAR	VAR ([ALL | DISTINCT] expression) [OVER (<partition_by_clause> [<order_by_clause>])]	Returns the statistical variance of all values in the specified expression. May be followed by the OVER clause.
Aggregate	VARP	VARP ([ALL | DISTINCT] expression) [OVER (<partition_by_clause> [<order_by_clause>])]	Returns the statistical variance for the population for all values in the specified expression.
Conversion	CAST	CAST (expression AS data_type [(length)])	These functions convert an expression of one data type to another.
Conversion	CONVERT	CONVERT (data_type [(length)] , expression [, style])	These functions convert an expression of one data type to another.
Conversion	FORMAT	FORMAT(value , format [, culture])	Returns a value formatted with the specified format and optional culture. Use the FORMAT function for locale-aware formatting of date/time and number values as strings. For general data type conversions, use CAST or CONVERT.
Conversion	PARSE	PARSE (string_value AS data_type [USING culture])	Returns the result of an expression, translated to the requested data type in SQL Server.
Conversion	STR	STR (float_expression [, length [, decimal]])	Returns character data converted from numeric data. The character data is right-justified, with a specified length and decimal precision.
Conversion	TRY_CAST	TRY_CAST (expression AS data_type [(length)])	Returns a value cast to the specified data type if the cast succeeds; otherwise, returns null.
Conversion	TRY_CONVERT	TRY_CONVERT (data_type [(length)] , expression [, style])	Returns a value cast to the specified data type if the cast succeeds; otherwise, returns NULL.

Category	Function Name	Syntax	Description
Conversion	TRY_PARSE	TRY_PARSE (string_value AS data_type [USING culture])	Returns the result of an expression, translated to the requested data type, or null if the cast fails in SQL Server. Use TRY_PARSE only for converting from string to date/time and number types.
Date/time - Creating	DATEADD	DATEADD (*datepart*, *number*, *date*)	Returns a new **datetime** value by adding an interval to the specified *datepart* of the specified *date*.
Date/time - Creating	DATEFROMPARTS	DATEFROMPARTS (*year*, *month*, *day*)	Returns a **date** value for the specified year, month, and day.
Date/time - Creating	DATETIME2FROMPARTS	DATETIME2FROMPARTS (*year*, *month*, *day*, *hour*, *minute*, *seconds*, *fractions*, *precision*)	Returns a **datetime2** value for the specified date and time, with the specified precision.
Date/time - Creating	DATETIMEFROMPARTS	DATETIMEFROMPARTS (*year*, *month*, *day*, *hour*, *minute*, *seconds*, *milliseconds*)	Returns a **datetime** value for the specified date and time.
Date/time - Creating	DATETIMEOFFSETFROMPARTS	DATETIMEOFFSETFROMPARTS (*year*, *month*, *day*, *hour*, *minute*, *seconds*, *fractions*, *hour_offset*, *minute_offset*, *precision*)	Returns a **datetimeoffset** value for the specified date and time, with the specified offsets and precision.
Date/time - Creating	EOMONTH	EOMONTH (*start_date* [, *month_to_add*])	Returns the last day of the month containing the specified date, with an optional offset.
Date/time - Creating	SMALLDATETIMEFROMPARTS	SMALLDATETIMEFROMPARTS (*year*, *month*, *day*, *hour*, *minute*)	Returns a **smalldatetime** value for the specified date and time.
Date/time - Creating	SWITCHOFFSET	SWITCHOFFSET (*DATETIMEOFFSET*, *time_zone*)	SWITCHOFFSET changes the time zone offset of a DATETIMEOFFSET value, and preserves the UTC value.
Date/time - Creating	TIMEFROMPARTS	TIMEFROMPARTS (*hour*, *minute*, *seconds*, *fractions*, *precision*)	Returns a **time** value for the specified time, with the specified precision.
Date/time - Creating	TODATETIMEOFFSET	TODATETIMEOFFSET (*expression*, *time_zone*)	TODATETIMEOFFSET transforms a datetime2 value into a datetimeoffset value. *TODATETIMEOFFSET* interprets the datetime2 value in local time, for the specified time_zone.
Date/time - Differences	DATEDIFF	DATEDIFF (*datepart*, *startdate*, *enddate*)	Returns the number of date or time *datepart* boundaries, crossed between two specified dates.
Date/time - Differences	DATEDIFF_BIG	DATEDIFF_BIG (*datepart*, *startdate*, *enddate*)	Returns the number of date or time *datepart* boundaries, crossed between two specified dates.
Date/time - Conversion	DATENAME	DATENAME (*datepart*, *date*)	Returns a character string representing the specified *datepart* of the specified date.

Category	Function Name	Syntax	Description
Date/time - Conversion	DATEPART	DATEPART (*datepart*, *date*)	Returns an integer representing the specified *datepart* of the specified *date*.
Date/time - Conversion	DATETRUNC	DATETRUNC (*datepart*, *date*)	Returns an input *date* truncated to a specified *datepart*.
Date/time - Conversion	DAY	DAY (*date*)	Returns an integer representing the day part of the specified *date*.
Date/time - Conversion	MONTH	MONTH (*date*)	Returns an integer representing the month part of a specified *date*.
Date/time - Conversion	YEAR	YEAR (*date*)	Returns an integer representing the year part of a specified *date*.
Date/time - Now	SYSDATETIME	SYSDATETIME ( )	Returns a **datetime2(7)** value containing the date and time of the computer on which the instance of SQL Server runs. The returned value doesn't include the time zone offset.
Date/time - Now	SYSDATETIMEOFFSET	SYSDATETIMEOFFSET ( )	Returns a **datetimeoffset(7)** value containing the date and time of the computer on which the instance of SQL Server runs. The returned value includes the time zone offset.
Date/time - Now	SYSUTCDATETIME	SYSUTCDATETIME ( )	Returns a **datetime2(7)** value containing the date and time of the computer on which the instance of SQL Server is running. The function returns the date and time values as UTC time (Coordinated Universal Time).
Date/time - Now	CURRENT_TIMESTAMP	CURRENT_TIMESTAMP	Returns a **datetime** value containing the date and time of the computer on which the instance of SQL Server runs. The returned value doesn't include the time zone offset.
Date/time - Now	GETDATE	GETDATE ( )	Returns a **datetime** value containing the date and time of the computer on which the instance of SQL Server runs. The returned value doesn't include the time zone offset.
Date/time - Now	GETUTCDATE	GETUTCDATE ( )	Returns a **datetime** value containing the date and time of the computer on which the instance of SQL Server runs. The function returns the date and time values as UTC time (Coordinated Universal Time).

Category	Function Name	Syntax	Description
Date/time - Now	CURRENT_DATE	CURRENT_DATE	Returns a **date** value containing only the date of the computer on which the instance of the Database Engine runs. The returned value doesn't include the time and the time zone offset.
Date/time - Validation	ISDATE	ISDATE (*expression*)	Determines whether a **datetime** or **smalldatetime** input expression has a valid date or time value.
Logical	CHOOSE	CHOOSE (index, val_1, val_2 [, val_n])	Returns the item at the specified index from a list of values in SQL Server.
Logical	GREATEST	GREATEST (expression1 [, ...expressionN])	This function returns the maximum value from a list of one or more expressions.
Logical	IIF	IIF(boolean_expression, true_value, false_value)	Returns one of two values, depending on whether the Boolean expression evaluates to true or false in SQL Server.
Logical	LEAST	LEAST (expression1 [, ...expressionN])	This function returns the minimum value from a list of one or more expressions.
Mathematical	ABS	ABS (numeric_expression)	A mathematical function that returns the absolute (positive) value of the specified numeric expression. (ABS changes negative values to positive values. ABS has no effect on zero or positive values.)
Mathematical	CEILING	CEILING (numeric_expression)	This function returns the smallest integer greater than, or equal to, the specified numeric expression.
Mathematical	EXP	EXP (float_expression)	Returns the exponential value of the specified **float** expression.
Mathematical	FLOOR	FLOOR (numeric_expression)	Returns the largest integer less than or equal to the specified numeric expression.
Mathematical	LOG	LOG (float_expression [, base])	Returns the natural logarithm of the specified **float** expression in SQL Server.
Mathematical	LOG10	LOG10 (float_expression)	Returns the base-10 logarithm of the specified **float** expression.
Mathematical	POWER	POWER (float_expression , y)	Returns the value of the specified expression to the specified power.
Mathematical	ROUND	ROUND (numeric_expression , length [,function])	Returns a numeric value, rounded to the specified length or precision.
Mathematical	SIGN	SIGN (numeric_expression)	Returns the positive (+1), zero (0), or negative (-1) sign of the specified expression.

Category	Function Name	Syntax	Description
Mathematical	SQRT	SQRT (float_expression)	Returns the square root of the specified float value.
Mathematical	SQUARE	SQUARE (float_expression)	Returns the square of the specified float value.
String	ASCII	ASCII (character_expression)	Returns the ASCII code value of the leftmost character of a character expression.
String	CHAR	CHAR (integer_expression)	Returns the single-byte character with the specified integer code, as defined by the character set and encoding of the default collation of the current database.
String	CHARINDEX	CHARINDEX (expressionToFind , expressionToSearch [, start_location])	This function searches for one character expression inside a second character expression, returning the starting position of the first expression if found.
String	CONCAT	CONCAT (argument1 , argument2 [, argumentN] ...)	This function returns a string resulting from the concatenation, or joining, of two or more string values in an end-to-end manner.
String	CONCAT_WS	CONCAT_WS (separator , argument1 , argument2 [, argumentN] ...)	This function returns a string resulting from the concatenation, or joining, of two or more string values in an end-to-end manner. It separates those concatenated string values with the delimiter specified in the first function argument. (CONCAT_WS indicates concatenate with separator.)
String	DIFFERENCE	DIFFERENCE (character_expression , character_expression)	This function returns an integer value measuring the difference between the SOUNDEX() values of two different character expressions.
String	LEFT	LEFT (character_expression , integer_expression)	Returns the left part of a character string with the specified number of characters.
String	LEN	LEN (string_expression)	Returns the number of characters of the specified string expression, excluding trailing spaces.
String	LOWER	LOWER (character_expression)	Returns a character expression after converting uppercase character data to lowercase.
String	LTRIM	LTRIM (character_expression , [characters])	Removes space character char(32) or other specified characters from the start of a string.
String	NCHAR	NCHAR (integer_expression)	Returns the Unicode character with the specified integer code, as defined by the Unicode standard.
String	PATINDEX	PATINDEX ('%pattern%' , expression)	Returns the starting position of the first occurrence of a pattern in a specified expression, or zero if the pattern is not found, on all valid text and character data types.

Category	Function Name	Syntax	Description
String	QUOTENAME	QUOTENAME ('character_string' [, 'quote_character'])	Returns a Unicode string with the delimiters added to make the input string a valid SQL Server delimited identifier.
String	REPLACE	REPLACE (string_expression , string_pattern , string_replacement)	Replaces all occurrences of a specified string value with another string value.
String	REPLICATE	REPLICATE (string_expression , integer_expression)	Repeats a string value a specified number of times.
String	REVERSE	REVERSE (string_expression)	Returns the reverse order of a string value.
String	RIGHT	RIGHT (character_expression , integer_expression)	Returns the right part of a character string with the specified number of characters.
String	RTRIM	RTRIM (character_expression , [characters])	Removes space character char(32) or other specified characters from the end of a string.
String	SOUNDEX	SOUNDEX (character_expression)	Returns a four-character (SOUNDEX) code to evaluate the similarity of two strings.
String	SPACE	SPACE (integer_expression)	Returns a string of repeated spaces.
String	STRING_AGG	STRING_AGG (expression, separator) [WITHIN GROUP (ORDER BY <order_by_expression_list> [ASC | DESC])]	Concatenates the values of string expressions and places separator values between them. The separator isn't added at the end of string.
String	STRING_ESCAPE	STRING_ESCAPE(text , type)	Escapes special characters in texts and returns text with escaped characters.
String	STRING_SPLIT	STRING_SPLIT (string , separator [, enable_ordinal])	STRING_SPLIT is a table-valued function that splits a string into rows of substrings, based on a specified separator character.
String	STUFF	STUFF (character_expression , start , length , replace_with_expression)	The STUFF function inserts a string into another string. It deletes a specified length of characters in the first string at the start position and then inserts the second string into the first string at the start position.
String	SUBSTRING	SUBSTRING (expression, start, length)	Returns part of a character, binary, text, or image expression in SQL Server.
String	TRANSLATE	TRANSLATE (inputString, characters, translations)	Returns the string provided as a first argument, after some characters specified in the second argument are translated into a destination set of characters, specified in the third argument.

Category	Function Name	Syntax	Description
String	TRIM	TRIM ([characters FROM] string)	Removes the space character char(32) or other specified characters from the start and end of a string.
String	UNICODE	UNICODE ('ncharacter_expression')	Returns the integer value, as defined by the Unicode standard, for the first character of the input expression.
String	UPPER	UPPER (character_expression)	Returns a character expression with lowercase character data converted to uppercase.
Analytic	CUME_DIST	CUME_DIST( ) OVER ([partition_by_clause] order_by_clause)	For SQL Server, this function calculates the cumulative distribution of a value within a group of values. In other words, CUME_DIST calculates the relative position of a specified value in a group of values.
Analytic	FIRST_VALUE	FIRST_VALUE ([scalar_expression]) [IGNORE NULLS | RESPECT NULLS] OVER ([partition_by_clause] order_by_clause [rows_range_clause])	Returns the first value in an ordered set of values.
Analytic	LAG	LAG (scalar_expression [, offset] [, default]) [IGNORE NULLS | RESPECT NULLS] OVER ([partition_by_clause] order_by_clause)	Accesses data from a previous row in the same result set. LAG provides access to a row at a given physical offset that comes before the current row. Use this analytic function in a SELECT statement to compare values in the current row with values in a previous row.
Analytic	LAST_VALUE	LAST_VALUE ([scalar_expression]) [IGNORE NULLS | RESPECT NULLS] OVER ([partition_by_clause] order_by_clause [rows_range_clause])	Returns the last value in an ordered set of values.
Analytic	LEAD	LEAD (scalar_expression [, offset] [, default]) [IGNORE NULLS | RESPECT NULLS] OVER ([partition_by_clause] order_by_clause)	Accesses data from a subsequent row in the same result set. LEAD provides access to a row at a given physical offset that follows the current row. Use this analytic function in a SELECT statement to compare values in the current row with values in a following row.
Analytic	PERCENT_RANK	PERCENT_RANK( ) OVER ([partition_by_clause] order_by_clause)	Calculates the relative rank of a row within a group of rows in SQL Server. Use PERCENT_RANK to evaluate the relative standing of a value within a query result set or partition. PERCENT_RANK is similar to the CUME_DIST function.

Category	Function Name	Syntax	Description
Analytic	PERCENTILE_CONT	PERCENTILE_CONT (numeric_literal) WITHIN GROUP (ORDER BY order_by_expression [ASC | DESC]) OVER ([<partition_by_clause>])	Calculates a percentile based on a continuous distribution of the column value in the SQL Server Database Engine. The result is interpolated, and might not equal any of the specific values in the column.
Analytic	PERCENTILE_DISC	PERCENTILE_DISC (numeric_literal) WITHIN GROUP (ORDER BY order_by_expression [ASC | DESC]) OVER ([<partition_by_clause>])	Computes a specific percentile for sorted values in an entire rowset or within a rowset's distinct partitions in SQL Server. For a given percentile value P, PERCENTILE_DISC sorts the expression values in the ORDER BY clause.
Bit manipulation	LEFT_SHIFT()	LEFT_SHIFT (expression_value, shift_amount) expression_value << shift_amount	LEFT_SHIFT takes two parameters, and returns the first parameter bit-shifted left by the number of bits specified in the second parameter. The LEFT_SHIFT function is also accessible through the << operator.
Bit manipulation	RIGHT_SHIFT()	RIGHT_SHIFT (expression_value, shift_amount) expression_value >> shift_amount	RIGHT_SHIFT takes two parameters, and returns the first parameter bit-shifted right by the number of bits specified in the second parameter. The RIGHT_SHIFT function is also accessible through the >> operator.
Bit manipulation	BIT_COUNT()	BIT_COUNT (expression_value)	BIT_COUNT takes one parameter and returns the number of bits set to 1 in that parameter as a bigint type.
Bit manipulation	GET_BIT()	GET_BIT (expression_value, bit_offset)	GET_BIT takes two parameters and returns the bit in expression_value that is in the offset defined by bit_offset.
Bit manipulation	SET_BIT()	SET_BIT (expression_value, bit_offset [, bit_value])	SET_BIT returns expression_value offset by the bit defined by bit_offset. The bit value defaults to 1, or is set by bit_value.
Collation	COLLATIONPROPERTY	COLLATIONPROPERTY(collation_name , property)	This function returns the requested property of a specified collation.
Collation	TERTIARY_WEIGHTS	TERTIARY_WEIGHTS(non_Unicode_character_string_expression)	For each character in a non-Unicode string expression - defined with a SQL tertiary collation - this function returns a binary string of weights.
Ranking	DENSE_RANK	DENSE_RANK ( ) OVER ([<partition_by_clause>] < order_by_clause >)	This function returns the rank of each row within a result set partition, with no gaps in the ranking values. The rank of a specific row is one plus the number of distinct rank values that come before that specific row.

Category	Function Name	Syntax	Description
Ranking	NTILE	NTILE (integer_expression) OVER ([<partition_by_clause>] < order_by_clause >)	Distributes the rows in an ordered partition into a specified number of groups. The groups are numbered, starting at one. For each row, NTILE returns the number of the group to which the row belongs.
Ranking	RANK	RANK ( ) OVER ([partition_by_clause] order_by_clause)	Returns the rank of each row within the partition of a result set. The rank of a row is one plus the number of ranks that come before the row in question.
Ranking	ROW_NUMBER	ROW_NUMBER ( ) OVER ([PARTITION BY value_expression , ... [n]] order_by_clause)	Numbers the output of a result set. More specifically, returns the sequential number of a row within a partition of a result set, starting at 1 for the first row in each partition.
Trigonomic	ACOS	ACOS (float_expression)	A function that returns the angle, in radians, whose cosine is the specified float expression. This is also called arccosine.
Trigonomic	ASIN	ASIN (float_expression)	A function that returns the angle, in radians, whose sine is the specified **float** expression. This is also called **arcsine**.
Trigonomic	ATAN	ATAN (float_expression)	A function that returns the angle, in radians, whose tangent is a specified **float** expression. This is also called arctangent.
Trigonomic	ATN2	ATN2 (float_expression , float_expression)	Returns the angle, in radians, between the positive x-axis and the ray from the origin to the point (y, x), where x and y are the values of the two specified float expressions.
Trigonomic	COS	COS (float_expression)	A mathematical function that returns the trigonometric cosine of the specified angle - measured in radians - in the specified expression.
Trigonomic	COT	COT (float_expression)	A mathematical function that returns the trigonometric cotangent of the specified angle - in radians - in the specified **float** expression.
Trigonomic	DEGREES	DEGREES (numeric_expression)	This function returns the corresponding angle, in degrees, for an angle specified in radians.
Trigonomic	PI	PI ( )	Returns the constant value of PI.
Trigonomic	RADIANS	RADIANS (numeric_expression)	Returns radians when a numeric expression, in degrees, is entered.

Category	Function Name	Syntax	Description
Trigonomic	SIN	SIN (float_expression)	Returns the trigonometric sine of the specified angle, in radians, and in an approximate numeric, **float**, expression.
Trigonomic	TAN	TAN (float_expression)	Returns the tangent of the input expression.
Expressions	CASE	CASE input_expression WHEN when_expression or CASE WHEN Boolean_expression THEN result_expression [...n] [ELSE else_result_expression] END	Evaluates a list of conditions and returns one of multiple possible result expressions.
Expressions	COALESCE	COALESCE (expression [,...n])	Evaluates the arguments in order and returns the current value of the first expression that initially doesn't evaluate to NULL.
Expressions	NULLIF	NULLIF (expression , expression)	Returns a null value if the two specified expressions are equal.