# System Functions for Transact-SQL

|  |  |  |
| --- | --- | --- |
| Analytic Functions | | |
| **CUME\_DIST** |  |  |
| **FIRST\_VALUE** | FIRST\_VALUE ( [scalar\_expression ] ) OVER ( [ partition\_by\_clause ] order\_by\_clause [ rows\_range\_clause ] ) | Returns the first value in an ordered set of values |
| **LAG** | LAG (scalar\_expression [,offset] [,default]) OVER ( [ partition\_by\_clause ] order\_by\_clause ) | Accesses data from a previous row in the same result set without the use of a self-join |
| **LAST\_VALUE** | LAST\_VALUE ( [scalar\_expression ) OVER ( [ partition\_by\_clause ] order\_by\_clause rows\_range\_clause ) | Returns the last value in an ordered set of values hire date of the last employee in each department for the given salary |
| **LEAD** | LEAD ( scalar\_expression [ ,offset ] , [ default ] ) OVER ( [ partition\_by\_clause ] order\_by\_clause ) | Accesses data from a subsequent row in the same result set without the use of a self-join |
| **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 SQL Server. |
| **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 distinct partitions of a rowset |
| **PERCENT\_RANK** | PERCENT\_RANK( ) OVER ( [ partition\_by\_clause ] order\_by\_clause ) | Calculates the relative rank of a row within a group of rows |
| Collation Functions | | |
| **Collation - COLLATIONPROPERTY** | COLLATIONPROPERTY( collation\_name , property ) | Returns the property of a specified collation |
| **Collation - TERTIARY\_WEIGHTS** | TERTIARY\_WEIGHTS( non\_Unicode\_character\_string\_expression ) | Returns a binary string of weights for each character in a non-Unicode string expression |
| Configuration Functions | | |
| **@@DBTS** | Returns the value of the current timestamp data type for the current database. This timestamp is guaranteed to be unique in the database. | 0x00000000000007D0 |
| **@@LANGID** | Returns the local language identifier (ID) of the language that is currently being used. | 0 |
| **@@LANGUAGE** | Returns the name of the language currently being used. | us\_english |
| **@@LOCK\_TIMEOUT** | SET LOCK\_TIMEOUT allows an application to set the maximum time that a statement waits on a blocked resource. When a statement has waited longer than the LOCK\_TIMEOUT setting, the blocked statement is automatically canceled, and an error message is returned to the application. | Returns the current lock time-out setting in milliseconds for the current session. returns a value of -1 if SET LOCK\_TIMEOUT has not yet been run in the current session |
| **@@MAX\_CONNECTIONS** | Returns the maximum number of simultaneous user connections allowed on an instance of SQL Server. | 32767 |
| **@@MAX\_PRECISION** | Returns the precision level used by decimal and numeric data types as currently set in the server. | 38 - By default, the maximum precision returns 38. |
| **@@NESTLEVEL** | Returns the nesting level of the current stored procedure execution (initially 0) on the local server. | 0 |
| **@@OPTIONS** | Returns information about the current SET options. | 5496 |
| **@@REMSERVER** | Returns the name of the remote SQL Server database server as it appears in the login record. | NULL |
| **@@SERVERNAME** | Returns the name of the local server that is running SQL Server. | DOCD-DEV-ND1 |
| **@@SERVICENAME** | Returns the name of the registry key under which SQL Server is running. @@SERVICENAME returns 'MSSQLSERVER' if the current instance is the default instance; this function returns the instance name if the current instance is a named instance. | MSSQLSERVER |
| **@@SPID** | Returns the session ID of the current user process. | 73 |
| **@@TEXTSIZE** | Returns the current value of the TEXTSIZE option. | 2147483647 |
| **@@VERSION** | Microsoft SQL Server 2014 (SP2-CU8) (KB4037356) - 12.0.5557.0 (X64)   Oct 3 2017 14:56:10   Copyright (c) Microsoft Corporation  Enterprise Edition: Core-based Licensing (64-bit) on Windows NT 6.3 <X64> (Build 9600: ) (Hypervisor) | Returns system and build information for the current installation of SQL Server. |
| Conversion Functions | | |
| **CAST and CONVERT** | CAST ( expression AS data\_type [ ( length ) ] ) CONVERT ( data\_type [ ( length ) ] , expression [ , style ] ) | Converts an expression of one data type to another. |
| **PARSE** | PARSE ( string\_value AS data\_type [ USING culture ] ) | Returns the result of an expression, translated to the requested data type |
| **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. |
| **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. |
| **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. |
| Cursor Functions | | |
| **@@CURSOR\_ROWS** | Returns the number of qualifying rows currently in the last cursor opened on the connection. | -m -1 0 n |
| **@@FETCH\_STATUS** | Returns the status of the last cursor FETCH statement issued against any cursor currently opened by the connection. | 0 -1 -2 -9 |
| **CURSOR\_STATUS** | CURSOR\_STATUS ( { 'local' , 'cursor\_name' } | { 'global' , 'cursor\_name' } | { 'variable' , 'cursor\_variable' } ) | A scalar function that allows the caller of a stored procedure to determine whether or not the procedure has returned a cursor and result set for a given parameter. |
| Data Type Functions | | |
| **DATALENGTH** | Returns the number of bytes used to represent any expression. |  |
| **IDENT\_CURRENT** | IDENT\_CURRENT( 'table\_name' ) | Returns the last identity value generated for a specified table or view. The last identity value generated can be for any session and any scope. |
| **IDENT\_INCR** | IDENT\_INCR ( 'table\_or\_view' ) | Returns the increment value during the creation of an identity column in a table or view that has an identity column. |
| **IDENT\_SEED** | IDENT\_SEED ( 'table\_or\_view' ) | Returns the original seed value that was specified when an identity column in a table or a view was created. |
| **IDENTITY (Function)** | IDENTITY (data\_type [ , seed , increment ] ) AS column\_name | Is used only in a SELECT statement with an INTO table clause to insert an identity column into a new table.Although similar, the IDENTITY function is not the IDENTITY property that is used with CREATE TABLE and ALTER TABLE. |
| **SQL\_VARIANT\_PROPERTY** | SQL\_VARIANT\_PROPERTY ( expression , property ) | Returns the base data type and other information about a sql\_variant value. |
| Date and Time Data Types and Functions | | |
| **@@DATEFIRST** | Returns the current value, for a session, of SET DATEFIRST. | SET DATEFIRST specifies the first day of the week. The U.S. English default is 7, Sunday. |
| **CURRENT\_TIMESTAMP** | Returns the current database system timestamp as a datetime value without the database time zone offset. This value is derived from the operating system of the computer on which the instance of SQL Server is running. | 2018-01-05 02:34:45.350 |
| **DATEADD** | Returns a specified date with the specified number interval (signed integer) added to a specified datepart of that date. | DATEADD (datepart , number , date ) SELECT DATEADD(YY,1,'2006-07-31');  Adds one year to input. |
| **DATEDIFF** | Returns the count (signed integer) of the specified datepart boundaries crossed between the specified startdate and enddate. | DATEDIFF ( datepart , startdate , enddate ) |
| **DATEDIFF\_BIG** | Returns the count (signed big integer) of the specified datepart boundaries crossed between the specified startdate and enddate. | DATEDIFF\_BIG ( datepart , startdate , enddate ) |
| **DATEFROMPARTS** | Returns a date value for the specified year, month, and day. DATEFROMPARTS ( year, month, day ) | SELECT DATEFROMPARTS ( 2010, 12, 31 ) O/P: 2010-12-31 |
| **DATENAME** | Returns a character string that represents the specified datepart of the specified date | DATENAME ( datepart , date ) SELECT DATENAME(year, '2018-01-05') O/P: 2018 |
| **DATEPART** | Returns an integer that represents the specified datepart of the specified date. | DATEPART ( datepart , date ) |
| **DATETIME2FROMPARTS** | DATETIME2FROMPARTS ( year, month, day, hour, minute, seconds, fractions, precision ) SELECT DATETIME2FROMPARTS ( 2010, 12, 31, 23, 59, 59, 0, 0 )  O/P: 2010-12-31 23:59:59 SELECT DATETIME2FROMPARTS ( 2010, 12, 31, 23, 59, 59, 10, 4 ) 2010-12-31 23:59:59.0010 | Returns a datetime2 value for the specified date and time and with the specified precision. |
| **DATETIMEFROMPARTS** | DATETIMEFROMPARTS ( year, month, day, hour, minute, seconds, milliseconds ) SELECT DATETIMEFROMPARTS ( 2010, 12, 31, 23, 59, 59, 0 ) o/P: 2010-12-31 23:59:59.000 | Returns a datetime value for the specified date and time. |
| **DATETIMEOFFSETFROMPARTS** | DATETIMEOFFSETFROMPARTS ( year, month, day, hour, minute, seconds, fractions, hour\_offset, minute\_offset, precision ) SELECT DATETIMEOFFSETFROMPARTS (2010, 12, 31, 14, 23, 23, 0, 12, 0, 7 ) O/P: 2010-12-31 14:23:23.0000000 +12:00 SELECT DATETIMEOFFSETFROMPARTS(2010, 12, 31, 14, 23, 23, 10, 12, 2,7) 2010-12-31 14:23:23.0000010 +12:02 | Returns a datetimeoffset value for the specified date and time and with the specified offsets and precision. |
| **DAY** | DAY ( date ) SELECT DAY('2015-04-30 01:01:01.1234567'); O/P: 30 | Returns an integer representing the day (day of the month) of the specified date. |
| **EOMONTH** | EOMONTH ( start\_date [, month\_to\_add ] ) DECLARE @date DATETIME = GETDATE(); SELECT EOMONTH ( @date ) AS 'This Month';--2018-01-31 SELECT EOMONTH ( @date, 1 ) AS 'Next Month';--2018-02-28 SELECT EOMONTH ( @date, -1 ) AS 'Last Month';--2017-12-31 | Returns the last day of the month that contains the specified date, with an optional offset. |
| **GETDATE** | Returns the current database system timestamp as a datetime value without the database time zone offset. This value is derived from the operating system of the computer on which the instance of SQL Server is running. | GETDATE ( ) |
| **GETUTCDATE** | Returns the current database system timestamp as a datetime value. The database time zone offset is not included. This value represents the current UTC time (Coordinated Universal Time). |  |
| **ISDATE** | Returns 1 if the expression is a valid date, time, or datetime value; otherwise, 0. | ISDATE ( expression ) |
| **MONTH** | Returns an integer that represents the month of the specified date. | MONTH ( date ) |
| **SMALLDATETIMEFROMPARTS** | Returns a smalldatetime value for the specified date and time. SMALLDATETIMEFROMPARTS ( year, month, day, hour, minute ) |  |
| **SWITCHOFFSET** | Returns a datetimeoffset value that is changed from the stored time zone offset to a specified new time zone offset. | SWITCHOFFSET ( DATETIMEOFFSET, time\_zone ) |
| **SYSDATETIME** | Returns a datetime2(7) value that contains the date and time of the computer on which the instance of SQL Server is running. |  |
| **SYSDATETIMEOFFSET** | Returns a datetimeoffset(7) value that contains the date and time of the computer on which the instance of SQL Server is running. The time zone offset is included. |  |
| **SYSUTCDATETIME** | Returns a datetime2 value that contains the date and time of the computer on which the instance of SQL Server is running. The date and time is returned as UTC time (Coordinated Universal Time). The fractional second precision specification has a range from 1 to 7 digits. The default precision is 7 digits. |  |
| **TIMEFROMPARTS** | Returns a time value for the specified time and with the specified precision. | TIMEFROMPARTS ( hour, minute, seconds, fractions, precision ) |
| **TODATETIMEOFFSET** | Returns a datetimeoffset value that is translated from a datetime2 expression. | TODATETIMEOFFSET ( expression , time\_zone ) |
| **YEAR** | Returns an integer that represents the year of the specified date. | YEAR ( date ) |
| JSON Functions | | |
| **ISJSON** | Tests whether a string contains valid JSON. | ISJSON ( expression ) |
| **JSON\_VALUE** | Extracts a scalar value from a JSON string. | JSON\_VALUE ( expression , path ) |
| **JSON\_QUERY** | Extracts an object or an array from a JSON string. | JSON\_QUERY ( expression [ , path ] ) |
| **JSON\_MODIFY** | Updates the value of a property in a JSON string and returns the updated JSON string. | JSON\_MODIFY ( expression , path , newValue ) |
| LOGICAL Functions | | |
|  |  |  |
| **CHOOSE** | Returns the item at the specified index from a list of values. SELECT CHOOSE ( 2, 'Manager', 'Director', 'Developer') O/P: Director | CHOOSE ( index, val\_1, val\_2 [, val\_n ] ) |
| **IIF** | Returns one of two values, depending on whether the Boolean expression evaluates to true or false SELECT IIF ( 45 > 30, 1,0 ) -- 1 | IIF ( boolean\_expression, true\_value, false\_value ) |
| Ranking Functions | | |
| **DENSE\_RANK** | Returns the rank of rows within the partition of a result set, without any gaps in the ranking. The rank of a row is one plus the number of distinct ranks that come before the row in question. | DENSE\_RANK ( ) OVER ( [ <partition\_by\_clause> ] < order\_by\_clause > ) |
| **NTILE** | 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 | NTILE (integer\_expression) OVER ( [ <partition\_by\_clause> ] < order\_by\_clause > ) |
| **RANK** | 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. ROW\_NUMBER and RANK are similar. ROW\_NUMBER numbers all rows sequentially (for example 1, 2, 3, 4, 5). RANK provides the same numeric value for ties (for example 1, 2, 2, 4, 5). | RANK ( ) OVER ( [ partition\_by\_clause ] order\_by\_clause ) |
| **ROW\_NUMBER** | 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. | ROW\_NUMBER ( ) OVER ( [ PARTITION BY value\_expression , ... [ n ] ] order\_by\_clause ) |
| String Functions | | |
| **ASCII** | Returns the ASCII code value of the leftmost character of a character expression. | ASCII ( character\_expression ) |
| **CHAR** | Converts an int ASCII code to a character. | CHAR ( integer\_expression ) |
| **CHARINDEX** | CHARINDEX (expressionToFind , expressionToSearch [ , start\_location ]) SELECT CHARINDEX('is', 'This is a string', 4); --6 | Searches an expression for another expression and returns its starting position if found. |
| **CONCAT** | CONCAT ( string\_value1, string\_value2 [, string\_valueN ] ) SELECT CONCAT ( 'Happy ', 11, '/', '25' );--Happy 11/25 | Returns a string that is the result of concatenating two or more string values. |
| **CONCAT\_WS** | CONCAT\_WS ( separator, argument1, argument1 [, argumentN]… ) SELECT CONCAT\_WS(',','Way',NULL,98052); --Way,98052 | Concatenates a variable number of arguments with a delimiter specified in the 1st argument. ( CONCAT\_WS indicates concatenate with separator.) 2017 feature |
| **DIFFERENCE** | DIFFERENCE ( character\_expression , character\_expression ) SELECT SOUNDEX('Green'), SOUNDEX('Greene'), DIFFERENCE('Green','Greene');--G650 G650 4 | Returns an integer value that indicates the difference between the SOUNDEX values of two character expressions. The integer returned is the number of characters in the SOUNDEX values that are the same. The return value ranges from 0 through 4: 0 indicates weak or no similarity, and 4 indicates strong similarity or the same values. DIFFERENCE and SOUNDEX are collation sensitive. |
| **FORMAT** | FORMAT ( value, format [, culture ] ) SELECT FORMAT( Getdate(), 'dd/MM/yyyy', 'en-US' ),FORMAT(123456789,'###-##-####');-- 05/01/2018 123-45-6789 | Returns a value formatted with the specified format and optional culture in SQL Server 2017. Use the FORMAT function for locale-aware formatting of date/time and number values as strings. |
| **LEFT** | LEFT ( character\_expression , integer\_expression ) SELECT LEFT('abcdefg',2);--ab | Returns the left part of a character string with the specified number of characters. |
| **LEN** | Returns the number of characters of the specified string expression, excluding trailing blanks. | LEN ( string\_expression ) |
| **LOWER** | Returns a character expression after converting uppercase character data to lowercase. | LOWER ( character\_expression ) |
| **LTRIM** | Returns a character expression after it removes leading blanks. | LTRIM ( character\_expression ) |
| **NCHAR** | Returns the Unicode character with the specified integer code, as defined by the Unicode standard | NCHAR ( integer\_expression ) Select NCHAR(20) -- |
| **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. | SELECT PATINDEX('%en\_ure%', 'please ensure the door is locked');-- 8 |
| **QUOTENAME** | Returns a Unicode string with the delimiters added to make the input string a valid SQL Server delimited identifier. | QUOTENAME ( 'character\_string' [ , 'quote\_character' ] ) SELECT QUOTENAME('abc def'); -- [abc def] SELECT QUOTENAME('abc def','{'); -- {abc def} |
| **REPLACE** | Replaces all occurrences of a specified string value with another string value. | REPLACE ( string\_expression , string\_pattern , string\_replacement ) |
| **REPLICATE** | Repeats a string value a specified number of times. | REPLICATE ( string\_expression ,integer\_expression ) |
| **REVERSE** | Returns the reverse order of a string value. | REVERSE ( string\_expression ) |
| **RIGHT** | Returns the right part of a character string with the specified number of characters. | RIGHT ( character\_expression , integer\_expression ) |
| **RTRIM** | Returns a character string after truncating all trailing spaces. | RTRIM ( character\_expression ) |
| **SOUNDEX** | Returns a four-character (SOUNDEX) code to evaluate the similarity of two strings. SOUNDEX converts an alphanumeric string to a four-character code that is based on how the string sounds when spoken. The first character of the code is the first character of character\_expression, converted to upper case. The second through fourth characters of the code are numbers that represent the letters in the expression. The letters A, E, I, O, U, H, W, and Y are ignored unless they are the first letter of the string. | SOUNDEX ( character\_expression ) |
| **SPACE** | Returns a string of repeated spaces. | SPACE ( integer\_expression ) |
| **STR** | Returns character data converted from numeric data. | STR ( float\_expression [ , length [ , decimal ] ] ) |
| **STRING\_AGG** | Concatenates the values of string expressions and places separator values between them. The separator is not added at the end of string. | STRING\_AGG ( expression, separator ) [ <order\_clause> ] <order\_clause> ::=WITHIN GROUP ( ORDER BY <order\_by\_expression\_list> [ ASC | DESC ] ) 2017 Feature |
| **STRING\_ESCAPE** | Escapes special characters in texts and returns text with escaped characters. STRING\_ESCAPE is a deterministic function. | STRING\_ESCAPE( text , type ) SELECT STRING\_ESCAPE('\ / \\ " ', 'json') AS escapedText; O/P: \\\t\/\n\\\\\t\"\t 2016-Feature |
| **STRING\_SPLIT** | Splits the character expression using specified separator | STRING\_SPLIT ( string , separator ) |
| **STUFF** | 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. | STUFF ( character\_expression , start , length , replaceWith\_expression ) |
| **SUBSTRING** | Returns part of a character, binary, text, or image expression in SQL Server. | SUBSTRING ( expression ,start , length ) |
| **TRANSLATE** | Returns the string provided as a first argument after some characters specified in the second argument are translated into a destination set of characters. TRANSLATE function will return an error if characters and translations have different lengths. TRANSLATE function should return unchanged input if null vales are provided as characters or replacement arguments. The behavior of the TRANSLATE function should be identical to the REPLACE function. The behavior of the TRANSLATE function is equivalent to using multiple REPLACE functions. | TRANSLATE ( inputString, characters, translations) SELECT TRANSLATE('2\*[3+4]/{7-2}', '[]{}', '()()'); O/P: 2\*(3+4)/(7-2) 2017-Feature |
| **TRIM** | Removes the space character char(32) or other specified characters from the start or end of a string. | TRIM ( [ characters FROM ] string ) SELECT TRIM( '.,! ' FROM '# test .') AS Result; O/P: # test 2017-Feature |
| **UNICODE** | Returns the integer value, as defined by the Unicode standard, for the first character of the input expression. | UNICODE ( 'ncharacter\_expression' ) |
| **UPPER** | Returns a character expression with lowercase character data converted to uppercase. | UPPER ( character\_expression ) |
| System Functions | | |
| **$PARTITION** | Returns the partition number into which a set of partitioning column values would be mapped for any specified partition function | [ database\_name. ] $PARTITION.partition\_function\_name(expression) |
| **@@ERROR** | Returns the error number for the last Transact-SQL statement executed. Returns 0 if the previous Transact-SQL statement encountered no errors. | integer |
| **@@IDENTITY** | Is a system function that returns the last-inserted identity value. | numeric(38,0) |
|  | If the statement did not affect any tables with identity columns, @@IDENTITY returns NULL |  |
| **@@PACK\_RECEIVED** | Returns the number of input packets read from the network by SQL Server since it was last started. | integer |
| **@@ROWCOUNT** | Returns the number of rows affected by the last statement. If the number of rows is more than 2 billion, use ROWCOUNT\_BIG. | int |
| **@@TRANCOUNT** | Returns the number of BEGIN TRANSACTION statements that have occurred on the current connection. The BEGIN TRANSACTION statement increments @@TRANCOUNT by 1. ROLLBACK TRANSACTION decrements @@TRANCOUNT to 0, except for ROLLBACK TRANSACTION savepoint\_name, which does not affect @@TRANCOUNT. COMMIT TRANSACTION or COMMIT WORK decrement @@TRANCOUNT by 1. | integer |
| **BINARY\_CHECKSUM** | Returns the binary checksum value computed over a row of a table or over a list of expressions. | BINARY\_CHECKSUM ( \* | expression [ ,...n ] ) |
| **CHECKSUM** | The CHECKSUM function returns the checksum value computed over a table row, or over an expression list. Use CHECKSUM to build hash indexes. | CHECKSUM ( \* | expression [ ,...n ] ) |
| **COMPRESS** | This function compresses the input expression, using the GZIP algorithm. The function returns a byte array of type varbinary(max). | COMPRESS ( expression ) |
| **CONNECTIONPROPERTY** | For a request that comes in to the server, this function returns information about the connection properties of the unique connection which supports that request. | CONNECTIONPROPERTY ( property ) |
| **CONTEXT\_INFO** | This function returns the context\_info value either set for the current session or batch, or derived through use of the SET CONTEXT\_INFO statement. | CONTEXT\_INFO()  If context\_info was not set: In SQL Server returns NULL. In SQL Database returns a unique session-specific GUID. |
| **CURRENT\_REQUEST\_ID** | This function returns the ID of the current request within the current session. | CURRENT\_REQUEST\_ID()  smallint |
| **CURRENT\_TRANSACTION\_ID** | This function returns the transaction ID of the current transaction in the current session. | CURRENT\_TRANSACTION\_ID( )  bigint -2016 |
| **DECOMPRESS** | Decompress input expression using GZIP algorithm. Result of the compression is byte array (VARBINARY(MAX) type). | DECOMPRESS ( expression ) |
| **ERROR\_LINE** | Returns the line number at which an error occurred that caused the CATCH block of a TRY…CATCH construct to be run. | ERROR\_LINE ( )  Int |
| **ERROR\_MESSAGE** | Returns the message text of the error that caused the CATCH block of a TRY…CATCH construct to be run. | ERROR\_MESSAGE ( )  nvarchar(4000) |
| **ERROR\_NUMBER** | Returns the error number of the error that caused the CATCH block of a TRY…CATCH construct to be run. Returns NULL if called outside the scope of a CATCH block. | ERROR\_NUMBER ( )  Int |
| **ERROR\_PROCEDURE** | Returns the name of the stored procedure or trigger where an error occurred that caused the CATCH block of a TRY…CATCH construct to be run. | ERROR\_PROCEDURE ( )  nvarchar(128) |
| **ERROR\_SEVERITY** | Returns the severity of the error that caused the CATCH block of a TRY…CATCH construct to be run. Returns NULL if called outside the scope of a CATCH block. | ERROR\_SEVERITY ( )  int |
| **ERROR\_STATE** | Returns the state number of the error that caused the CATCH block of a TRY…CATCH construct to be run. Returns NULL if called outside the scope of a CATCH block. | ERROR\_STATE ( )  int |
| **FORMATMESSAGE** | Constructs a message from an existing message in sys.messages or from a provided string. The functionality of FORMATMESSAGE resembles that of the RAISERROR statement. However, RAISERROR prints the message immediately, while FORMATMESSAGE returns the formatted message for further processing. | FORMATMESSAGE ( { msg\_number | ' msg\_string ' } , [ param\_value [ ,...n ] ] ) |
| **GET\_FILESTREAM\_TRANSACTION\_CONTEXT** | Returns a token that represents the current transaction context of a session. The token is used by an application to bind FILESTREAM file-system streaming operations to the transaction. | GET\_FILESTREAM\_TRANSACTION\_CONTEXT ()  varbinary(max) |
| **GETANSINULL** | Returns the default nullability for the database for this session. When the nullability of the specified database allows for null values and the column or data type nullability is not explicitly defined, GETANSINULL returns 1. This is the ANSI NULL default. | GETANSINULL ( [ 'database' ] ) |
| **HOST\_ID** | Returns the workstation identification number. The workstation identification number is the process ID (PID) of the application on the client computer that is connecting to SQL Server. | HOST\_ID ()  char(10) |
| **HOST\_NAME** | Returns the workstation name. When the parameter to a system function is optional, the current database, host computer, server user, or database user is assumed. Built-in functions must always be followed by parentheses. | HOST\_NAME () |
| **ISNULL** | Replaces NULL with the specified replacement value. | ISNULL ( check\_expression , replacement\_value ) |
| **ISNUMERIC** | Determines whether an expression is a valid numeric type. ISNUMERIC returns 1 when the input expression evaluates to a valid numeric data type; otherwise it returns 0. | ISNUMERIC ( expression )  int |
| **MIN\_ACTIVE\_ROWVERSION** | Returns the lowest active rowversion value in the current database. A rowversion value is active if it is used in a transaction that has not yet been committed. | MIN\_ACTIVE\_ROWVERSION  Returns a binary(8) value. |
| **NEWID** | Creates a unique value of type uniqueidentifier. | NEWID ( )  uniqueidentifier |
| **NEWSEQUENTIALID** | Creates a GUID that is greater than any GUID previously generated by this function on a specified computer since Windows was started. After restarting Windows, the GUID can start again from a lower range, but is still globally unique. | NEWSEQUENTIALID ( ) |
| **ROWCOUNT\_BIG** | Returns the number of rows affected by the last statement executed. This function operates like @@ROWCOUNT, except the return type of ROWCOUNT\_BIG is bigint. | ROWCOUNT\_BIG ( )  bigint |
| **SESSION\_CONTEXT** | Returns the value of the specified key in the current session context. The value is set by using the sp\_set\_session\_context | SESSION\_CONTEXT(N'key') |
| **XACT\_STATE** | Is a scalar function that reports the user transaction state of a current running request. XACT\_STATE indicates whether the request has an active user transaction, and whether the transaction is capable of being committed. | XACT\_STATE() Smallint |

# System Information Schema Views

# System Stored Procedures

# System Tables

# [System views](https://docs.microsoft.com/en-us/sql/relational-databases/system-views/replication-views-transact-sql)

# [Tables](https://docs.microsoft.com/en-us/sql/relational-databases/tables/tables)

# [Track changes](https://docs.microsoft.com/en-us/sql/relational-databases/track-changes/track-data-changes-sql-server)