The syntax of the CAST function

```
CAST(expression AS data_type)
```

The syntax of the CONVERT function

```
CONVERT(data_type, expression [, style])
```

Common style codes for converting date/time data to character data

Code	Output format
0 or 100	Mon dd yyyy hh:miAM/PM
1 or 101	mm/dd/yy or mm/dd/yyyy
7 or 107	Mon dd, yy or Mon dd, yyyy
8 or 108	hh:mi:ss
10 or 110	mm-dd-yy or mm-dd-yyyy
12 or 112	yymmdd or yyyymmdd
14 or 114	hh:mi:ss:mmm (24-hour clock)

Common style codes for converting real data to character data

Code	Output
0 (default)	6 digits maximum
1	8 digits; must use scientific notation
2	16 digits; must use scientific notation

Common style codes for converting money data to character data

Code	Output
0 (default)	2 digits to the right of the decimal point; no commas to the left
1	2 digits to the right of the decimal point; commas to the left
2	4 digits to the right of the decimal point; no commas to the left

Some of the string functions

Function	Description
`LEN(string)`	Returns the number of characters in the string. Leading spaces are included, but trailing spaces are not.
`LTRIM(string)`	Returns the string with any leading spaces removed.
`RTRIM(string)`	Returns the string with any trailing spaces removed.
`TRIM(string)`	Returns the string with any leading and trailing spaces removed.
`LEFT(string,length)`	Returns the specified number of characters from the beginning of the string.
`RIGHT(string,length)`	Returns the specified number of characters from the end of the string.
`SUBSTRING(string,start,length)`	Returns the specified number of characters from the string starting at the specified position.
`REPLACE(search,find,replace)`	Returns the search string with all occurrences of the find string replaced with the replace string.
`TRANSLATE(search,find,replace)`	Returns the search string with characters in the find string replaced with the characters in the replace string.
`REVERSE(string)`	Returns the string with the characters in reverse order.
`CHARINDEX(find,search[,start])`	Returns an integer that represents the position of the first occurrence of the find string in the search string starting at the specified position. If the starting position isn't specified, the search starts at the beginning of the string. If the string isn't found, the function returns zero.
`PATINDEX(find,search)`	Returns an integer that represents the position of the first occurrence of the find pattern in the search string. If the pattern isn't found, the function returns zero. The find pattern can include wildcard characters. If the pattern begins with a wildcard, the value returned is the position of the first non-wildcard character.
`CONCAT(value1,value2[,value3]...)`	Returns a string that contains a concatenation of the specified values. The values are implicitly converted to strings. A null value is converted to an empty string.
`CONCAT_WS(delimiter,value1, value2[,value3]...)`	Same as CONCAT but the values are separated by the specified delimiter.
`LOWER(string)`	Returns the string converted to lowercase letters.
`UPPER(string)`	Returns the string converted to uppercase letters.
`SPACE(integer)`	Returns a string with the specified number of space characters (blanks).

String function examples

Function	Result
`LEN('SQL Server')`	`10`
`LEN('  SQL Server  ')`	`12`
`LEFT('SQL Server', 3)`	`'SQL'`
`LTRIM('  SQL Server  ')`	`'SQL Server  '`
`RTRIM('  SQL Server  ')`	`'  SQL Server'`
`TRIM('  SQL Server  ')`	`'SQL Server'`
`LOWER('SQL Server')`	`'sql server'`
`UPPER('ca')CA`	
`PATINDEX('%v_r%', 'SQL Server')`	`8`
`CHARINDEX('SQL', '  SQL Server')`	`3`
`CHARINDEX('-', '(559) 555-1212')`	`10`
`SUBSTRING('(559) 555-1212', 7, 8)`	`555-1212`
`REPLACE(RIGHT('(559) 555-1212', 13), ') ', '-')`	`559-555-1212`
`TRANSLATE('(XDG) 197.TS224', '().', '[]-')`	`[XDG] 197-TS224`
`CONCAT('Run time: ',1.52,' seconds')`	`Run time: 1.52 seconds`
`CONCAT_WS('.', 559, 555, 1212)`	`559.555.1212`

Some of the numeric functions

Function	Description
`ROUND(number,length [,function])`	Returns the number rounded to the precision specified by length. If length is positive, the digits to the right of the decimal point are rounded. If it's negative, the digits to the left of the decimal point are rounded. To truncate the number rather than round it, code a non-zero value for function.
`ISNUMERIC(expression)`	Returns a value of 1 (true) if the expression is a numeric value; returns a value of 0 (false) otherwise.
`ABS(number)`	Returns the absolute value of the number.
`CEILING(number)`	Returns the smallest integer that is greater than or equal to the number.
`FLOOR(number)`	Returns the largest integer that is less than or equal to the number.
`SQUARE(float_number)`	Returns the square of a floating-point number.
`SQRT(float_number)`	Returns the square root of a floating-point number.
`RAND([integer])`	Returns a random floating-point number between 0 and 1. If integer is coded, it provides a starting value for the function. Otherwise, the function will return the same number each time it's invoked within the same query.

Examples that use the numeric functions

Function	Result
ROUND(12.5,0)	13.0
ROUND(12.4999,0)	12.0000
ROUND(12.4999,1)	12.5000
ROUND(12.4999,-1)	10.0000
ROUND(12.5,0,1)	12.0
ISNUMERIC(-1.25)	1
ISNUMERIC('SQL Server')	0
ISNUMERIC('2020-04-30')	0
ABS(-1.25)	1.25
CEILING(-1.25)	-1
FLOOR(-1.25)	-2
CEILING(1.25)	2
FLOOR(1.25)	1
SQUARE(5.2786)	27.86361796
SQRT(125.43)	11.199553562531
RAND()	0.243729

Some of the date/time functions

Function	Description
`GETDATE()`	Returns a datetime value for the current local date and time based on the system's clock.
`GETUTCDATE()`	Returns a datetime value for the current UTC date and time based on the system's clock and time zone setting.
`SYSDATETIME()`	Returns a datetime2(7) value for the current local date and time based on the system's clock.
`SYSUTCDATETIME()`	Returns a datetime2(7) value for the current UTC date and time based on the system's clock and time zone setting.
`SYSDATETIMEOFFSET()`	Returns a datetimeoffset(7) value for the current UTC date and time based on the system's clock and time zone setting with a time zone offset that is *not* adjusted for daylight savings time.
`DAY(date)`	Returns the day of the month as an integer.
`MONTH(date)`	Returns the month as an integer.
`YEAR(date)`	Returns the 4-digit year as an integer.
`DATENAME(datepart,date)`	Returns the part of the date specified by datepart as a character string.
`DATEPART(datepart,date)`	Returns the part of the date specified by datepart as an integer.
`DATEADD(datepart,number,date)`	Returns the date that results from adding the specified number of datepart units to the date.
`DATEDIFF(datepart,startdate,enddate)`	Returns the number of datepart units between the specified start and end dates.
`TODATETIMEOFFSET(datetime2,tzoffset)`	Returns a datetimeoffset value that results from adding the specified time zone offset to the specified datetime2 value.
`SWITCHOFFSET(datetimeoffset,tzoffset)`	Returns a datetimeoffset value that results from switching the time zone offset for the specified datetimeoffset value to the specified offset.
`EOMONTH(startdate[,months])`	Returns a date value for the last day of the month specified by the start date. If months is specified, the number of months is added to the start date before the end-of-month date is calculated.
`DATEFROMPARTS(year,month,day)`	Returns a date value for the specified year, month, and day.
`ISDATE(expression)`	Returns a value of 1 (true) if the expression is a valid date/time value; returns a value of 0 (false) otherwise.

Date part values and abbreviations

Argument	Abbreviations
year	yy, yyyy
quarter	qq, q
month	mm, m
dayofyear	dy, y
day	dd, d
week	wk, ww
weekday	dw
hour	hh
minute	mi, n
second	ss, s
millisecond	ms
microsecond	mcs
nanosecond	ns
tzoffset	tz

Examples that use date/time functions

Function	Result
GETDATE()	2020-04-30 14:10:13.813
GETUTCDATE()	2020-04-30 21:10:13.813
SYSDATETIME()	2020-04-30 14:10:13.8160822
SYSUTCDATETIME()	2020-04-30 21:10:13.8160822
SYSDATETIMEOFFSET()	2020-04-30 14:10:13.8160822 -07.00
MONTH('2020-04-30')	4
DATEPART(month,'2020-04-30')	4
DATENAME(month,'2020-04-30')	April
DATENAME(m,'2020-04-30')	April
EOMONTH('2020-02-01')	2020-02-29
EOMONTH('2020-02-01',2)	2020-04-30
DATEFROMPARTS(2020,4,3)	2020-04-03
ISDATE('2020-04-30')	1
ISDATE('2020-04-31')	0
ISDATE('23:59:59')	1
ISDATE('23:99:99')	0

Examples that use the DAY, MONTH, and YEAR functions

Function	Result
`DAY('2020-04-30')`	30
`MONTH('2020-04-30')`	4
`YEAR('2020-04-30')`	2020

Examples that use the DATEPART function

Function	Result
`DATEPART(day, '2020-04-30 11:35:00')`	30
`DATEPART(month, '2020-04-30 11:35:00')`	4
`DATEPART(year, '2020-04-30 11:35:00')`	2020
`DATEPART(hour, '2020-04-30 11:35:00')`	11
`DATEPART(minute, '2020-04-30 11:35:00')`	35
`DATEPART(second, '2020-04-30 11:35:00')`	0
`DATEPART(quarter, '2020-04-30 11:35:00')`	2
`DATEPART(dayofyear, '2020-04-30 11:35:00')`	121
`DATEPART(week, '2020-04-30 11:35:00')`	18
`DATEPART(weekday, '2020-04-30 11:35:00')`	5
`DATEPART(millisecond, '11:35:00.1234567')`	123
`DATEPART(microsecond, '11:35:00.1234567')`	123456
`DATEPART(nanosecond, '11:35:00.1234567')`	123456700
`DATEPART(tzoffset, '11:35:00.1234567 -07:00')`	-420

Examples that use the DATENAME function

Function	Result
DATENAME(day, '2020-04-30 11:35:00')	30
DATENAME(month, '2020-04-30 11:35:00')	April
DATENAME(year, '2020-04-30 11:35:00')	2020
DATENAME(hour, '2020-04-30 11:35:00')	11
DATENAME(minute, '2020-04-30 11:35:00')	35
DATENAME(second, '2020-04-30 11:35:00')	0
DATENAME(quarter, '2020-04-30 11:35:00')	2
DATENAME(dayofyear, '2020-04-30 11:35:00')	121
DATENAME(week, '2020-04-30 11:35:00')	18
DATENAME(weekday, '2020-04-30 11:35:00')	Thursday
DATENAME(millisecond, '11:35:00.1234567')	123
DATENAME(microsecond, '11:35:00.1234567')	123456
DATENAME(nanosecond, '11:35:00.1234567')	123456700
DATENAME(tzoffset, '11:35:00.1234567 -07:00')	-07:00

Examples that use the DATEADD function

Function	Result
DATEADD(day, 1, '2020-04-30 11:35:00')	2020-05-01 11:35:00.000
DATEADD(month, 1, '2020-04-30 11:35:00')	2020-05-30 11:35:00.000
DATEADD(year, 1, '2020-04-30 11:35:00')	2021-04-30 11:35:00.000
DATEADD(hour, 1, '2020-04-30 11:35:00')	2020-04-30 12:35:00.000
DATEADD(minute, 1, '2020-04-30 11:35:00')	2020-04-30 11:36:00.000
DATEADD(second, 1, '2020-04-30 11:35:00')	2020-04-30 11:35:01.000
DATEADD(quarter, 1, '2020-04-30 11:35:00')	2020-07-30 11:35:00.000
DATEADD(week, 1, '2020-04-30 11:35:00')	2020-05-07 11:35:00.000
DATEADD(month, -1, '2020-04-30 11:35:00')	2020-03-30 11:35:00.000
DATEADD(year, 1.5, '2020-04-30 11:35:00')	2021-04-30 11:35:00.000

Examples that use the DATEDIFF function

Function	Result
`DATEDIFF(day, '2019-07-01', '2020-04-30')`	304
`DATEDIFF(month, '2019-07-01', '2020-04-30')`	9
`DATEDIFF(year, '2019-07-01', '2020-04-30')`	1
`DATEDIFF(hour, '06:46:45', '11:35:00')`	5
`DATEDIFF(minute, '06:46:45', '11:35:00')`	289
`DATEDIFF(second, '06:46:45', '11:35:00')`	17295
`DATEDIFF(quarter, '2019-07-01', '2020-04-30')`	3
`DATEDIFF(week, '2019-07-01', '2020-04-30')`	43
`DATEDIFF(day, '2020-04-30', '2019-07-01')`	-304

Examples that use the addition and subtraction operators

Operation	Result
`CAST('2020-04-30 11:35:00' AS smalldatetime) + 1`	2020-05-01 11:35:00
`CAST('2020-04-30 11:35:00' AS smalldatetime) - 1`	2020-04-29 11:35:00
`CAST(CAST('2020-04-30' AS datetime)` `- CAST('2019-07-01' AS datetime) AS int)`	304