SQL | String Functions

Following is the list of String functions with examples.

## ASCII()

Ascii code value will come as output for a character expression.

### Example

The following query will give the Ascii code value of a given character.

Select ASCII ('word')

## CHAR()

Character will come as output for given Ascii code or integer.

### Example

The following query will give the character for a given integer.

Select CHAR(97)

## NCHAR()

Unicode character will come as output for a given integer.

### Example

The following query will give the Unicode character for a given integer.

Select NCHAR(300)

## CHARINDEX()

Starting position for given search expression will come as output in a given string expression.

### Example

The following query will give the starting position of 'G' character for given string expression 'KING'.

Select CHARINDEX('G', 'KING')

## LEFT()

Left part of the given string till the specified number of characters will come as output for a given string.

### Example

The following query will give the 'WORL' string as mentioned 4 number of characters for given string 'WORLD'.

Select LEFT('WORLD', 4)

## RIGHT()

Right part of the given string till the specified number of characters will come as output for a given string.

### Example

The following query will give the 'DIA' string as mentioned 3 number of characters for given string 'INDIA'.

Select RIGHT('INDIA', 3)

## SUBSTRING()

Part of a string based on the start position value and length value will come as output for a given string.

### Example

The following queries will give the 'WOR', 'DIA', 'ING' strings as we mentioned (1,3), (3,3) and (2,3) as start and length values respectively for given strings 'WORLD', 'INDIA' and 'KING'.

Select SUBSTRING ('WORLD', 1,3)

Select SUBSTRING ('INDIA', 3,3)

Select SUBSTRING ('KING', 2,3)

## LEN()

Number of characters will come as output for a given string expression.

### Example

The following query will give the 5 for the 'HELLO' string expression.

Select LEN('HELLO')

## LOWER()

Lowercase string will come as output for a given string data.

### Example

The following query will give the 'sqlserver' for the 'SQLServer' character data.

Select LOWER('SQLServer')

## UPPER()

Uppercase string will come as output for a given string data.

### Example

The following query will give the 'SQLSERVER' for the 'SqlServer' character data.

Select UPPER('SqlServer')

## LTRIM()

String expression will come as output for a given string data after removing leading blanks.

### Example

The following query will give the 'WORLD' for the '   WORLD' character data.

Select LTRIM(' WORLD')

## RTRIM()

String expression will come as output for a given string data after removing trailing blanks.

### Example

The following query will give the 'INDIA' for the 'INDIA   ' character data.

Select RTRIM('INDIA ')

## REPLACE()

String expression will come as output for a given string data after replacing all occurrences of specified character with specified character.

### Example

The following query will give the 'KNDKA' string for the 'INDIA' string data.

Select REPLACE('INDIA', 'I', 'K')

## REPLICATE()

Repeat string expression will come as output for a given string data with specified number of times.

### Example

The following query will give the 'WORLDWORLD' string for the 'WORLD' string data.

Select REPLICATE('WORLD', 2)

## REVERSE()

Reverse string expression will come as output for a given string data.

### Example

The following query will give the 'DLROW' string for the 'WORLD' string data.

Select REVERSE('WORLD')

## SPACE()

String will come as output with the specified number of spaces.

### Example

The following query will give the 'I LOVE INDIA'.

Select 'I'+space(1)+'LOVE'+space(1)+'INDIA'

## STUFF()

String expression will come as output for a given string data after replacing from starting character till the specified length with specified character.

### Example

The following query will give the 'AIJKFGH' string for the 'ABCDEFGH' string data as per given starting character and length as 2 and 4 respectively and 'IJK' as specified target string.

Select STUFF('ABCDEFGH', 2,4,'IJK')

SQL | Numeric Functions

**Numeric Functions** are used to perform operations on numbers and return numbers.  
Following are the numeric functions defined in SQL:

1. **ABS():** It returns the absolute value of a number.

**Syntax:** SELECT ABS(-243.5);

**Output:**243.5

SQL> SELECT ABS(-10);

+--------------------------------------+

| ABS(10)

+--------------------------------------+

| 10

+--------------------------------------+

1. **ACOS():** It returns the cosine of a number.

**Syntax:**  SELECT ACOS(0.25);

**Output:**1.318116071652818

1. **ASIN():** It returns the arc sine of a number.

**Syntax:** SELECT ASIN(0.25);

**Output:**0.25268025514207865

1. **ATAN():** It returns the arc tangent of a number.

**Syntax:** SELECT ATAN(2.5);

**Output:**1.1902899496825317

1. **CEIL():** It returns the smallest integer value that is greater than or equal to a number.

**Syntax:** SELECT CEIL(25.75);

**Output:**26

1. **CEILING():** It returns the smallest integer value that is greater than or equal to a number.

**Syntax:** SELECT CEILING(25.75);

**Output:**26

1. **COS():** It returns the cosine of a number.

**Syntax:** SELECT COS(30);

**Output:**0.15425144988758405

1. **COT():** It returns the cotangent of a number.

**Syntax:** SELECT COT(6);

**Output:**-3.436353004180128

1. **DEGREES():** It converts a radian value into degrees.

**Syntax:** SELECT DEGREES(1.5);

**Output:**85.94366926962348

SQL>SELECT DEGREES(PI());

+------------------------------------------+

| DEGREES(PI())

+------------------------------------------+

| 180.000000

+------------------------------------------+

1. **DIV():** It is used for integer division.

**Syntax:** SELECT 10 DIV 5;

**Output:**2

1. **EXP():** It returns e raised to the power of number.

**Syntax:** SELECT EXP(1);

**Output:**2.718281828459045

1. **FLOOR():** It returns the largest integer value that is less than or equal to a number.

**Syntax:** SELECT FLOOR(25.75);

**Output:**25

1. **GREATEST():** It returns the greatest value in a list of expressions.

**Syntax:** SELECT GREATEST(30, 2, 36, 81, 125);

**Output:**125

1. **LEAST():** It returns the smallest value in a list of expressions.

**Syntax:** SELECT LEAST(30, 2, 36, 81, 125);

**Output:**2

1. **LN():** It returns the natural logarithm of a number.

**Syntax:** SELECT LN(2);

**Output:**0.6931471805599453

1. **LOG10():** It returns the base-10 logarithm of a number.

**Syntax:** SELECT LOG(2);

**Output:**0.6931471805599453

1. **LOG2():** It returns the base-2 logarithm of a number.

**Syntax:** SELECT LOG2(6);

**Output:**2.584962500721156

1. **MOD():** It returns the remainder of n divided by m.

**Syntax:** SELECT MOD(18, 4);

**Output:**2

1. **PI():** It returns the value of PI displayed with 6 decimal places.

**Syntax:** SELECT PI();

**Output:**3.141593

1. **POW():** It returns m raised to the nth power.

**Syntax:** SELECT POW(4, 2);

**Output:**16

1. **RADIANS():** It converts a value in degrees to radians.

**Syntax:** SELECT RADIANS(180);

**Output:**3.141592653589793

1. **RAND():** It returns a random number.

**Syntax:** SELECT RAND();

**Output:**0.33623238684258644

1. **ROUND():** It returns a number rounded to a certain number of decimal places.

**Syntax:** SELECT ROUND(5.553);

**Output:**6

1. **SIGN():** It returns a value indicating the sign of a number.

**Syntax:** SELECT SIGN(255.5);

**Output:**1

1. **SIN():** It returns the sine of a number.

**Syntax:** SELECT SIN(2);

**Output:**0.9092974268256817

1. **SQRT():** It returns the square root of a number.

**Syntax:** SELECT SQRT(25);

**Output:**5

1. **TAN():** It returns the tangent of a number.

**Syntax:** SELECT TAN(1.75);

**Output:**-5.52037992250933

1. **ATAN2():** It returns the arctangent of the x and y coordinates, as an angle and expressed in radians.

**Syntax:** SELECT ATAN2(7);

**Output:**1.42889927219073

1. **TRUNCATE():** This doesn’t work for SQL Server. It returns 7.53635 truncated to 2 places right of the decimal point.

**Syntax:** SELECT TRUNCATE(7.53635, 2);

**Output:**7.53

## GETDATE()

It will return the current date along with time.

### Syntax

Syntax for the above function −

GETDATE()

### Example

The following query will return the current date along with time in MS SQL Server.

Select getdate() as currentdatetime

Get Date out of Current Date

SELECT

## CONVERT(Date, GETDATE()) [Current Date];

Get Time out of Current Date

SELECT

## CONVERT(Date, GETDATE()) [Current Date];

## DATEPART()

It will return the part of date or time.

### Syntax

Syntax for the above function −

DATEPART(datepart, datecolumnname)

### Example

**Example 1** − The following query will return the part of current date in MS SQL Server.

Select datepart(day, getdate()) as currentdate

**Example 2** − The following query will return the part of current month in MS SQL Server.

Select datepart(month, getdate()) as currentmonth

## DATEADD()

It will display the date and time by add or subtract date and time interval.

### Syntax

Syntax for the above function −

DATEADD(datepart, number, datecolumnname)

### Example

The following query will return the after 10 days date and time from the current date and time in MS SQL Server.

Select dateadd(day, 10, getdate()) as after10daysdatetimefromcurrentdatetime

## DATEDIFF()

It will display the date and time between two dates.

### Syntax

Syntax for the above function −

DATEDIFF(datepart, startdate, enddate)

### Example

The following query will return the difference of hours between 2015-11-16 and 2015-11-11 dates in MS SQL Server.

Select datediff(hour, 2015-11-16, 2015-11-11) as

differencehoursbetween20151116and20151111

## CONVERT()

It will display the date and time in different formats.

### Syntax

Syntax for the above function −

CONVERT(datatype, expression, style)

### Example

The following queries will return the date and time in different format in MS SQL Server.

SELECT CONVERT(VARCHAR(19),GETDATE())

SELECT CONVERT(VARCHAR(10),GETDATE(),10)

SELECT CONVERT(VARCHAR(10),GETDATE(),110)

DECLARE

@local\_time DATETIME,

@utc\_time DATETIME;

SET @local\_time = GETDATE();

SET @utc\_time = GETUTCDATE();

SELECT

CONVERT(VARCHAR(40), @local\_time)

AS 'Server local time';

SELECT

CONVERT(VARCHAR(40), @utc\_time)

AS 'Server UTC time'

SELECT

CONVERT(VARCHAR(40), DATEDIFF(hour, @utc\_time, @local\_time))

AS 'Server time zone';

GO

SELECT

CONVERT(DATE, GETDATE()),

CAST(GETDATE() AS DATE);

SELECT

CONVERT(TIME,GETDATE()),

CAST(GETDATE() AS TIME);

## SQL Server DATENAME() function overview

The DATENAME() function returns a string, [NVARCHAR](https://www.sqlservertutorial.net/sql-server-basics/sql-server-nvarchar/) type, that represents a specified date part e.g., year, month and day of a specified date.

The following shows the syntax of the DATENAME() function:

DATENAME(date\_part,input\_date)

The DATENAME() function accepts two arguments:

* date\_part is a part of the date that you want to return. The table below lists all valid date part values.
* input\_date is a literal date or an expression that can resolve to a [TIME](https://www.sqlservertutorial.net/sql-server-basics/sql-server-time/), [DATE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-date/), SMALLDATETIME, DATETIME, [DATETIME2](https://www.sqlservertutorial.net/sql-server-basics/sql-server-datetime2/), or [DATETIMEOFFSET](https://www.sqlservertutorial.net/sql-server-basics/sql-server-datetimeoffset/) value.

| **date\_part** | **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 |
| ISO\_WEEK | isowk, isoww |

## DATENAME() vs. DATEPART()

Note that DATENAME() is similar to the [DATEPART()](https://www.sqlservertutorial.net/sql-server-date-functions/sql-server-datepart-function/) except for the return type. The DATENAME() function returns the date part as a character string whereas the DATEPART() returns the date part as an integer.

See the following example:

**SELECT**

**DATEPART**(**year**, '2018-05-10') [**datepart**],

**DATENAME**(**year**, '2018-05-10') [**datename**];

The output looks the same:

datepart datename

----------- -----------

2018 2018

(1 row affected)

However, their [data types](https://www.sqlservertutorial.net/sql-server-basics/sql-server-data-types/) are different as shown in the following example:

**SELECT**

**DATEPART**(**year**, '2018-05-10') + '1' [**datepart**],

**DATENAME**(**year**, '2018-05-10') + '1' [**datename**] ;

The following shows the result:

datepart datename

----------- -----------

2019 20181

(1 row affected)

Because the DATEPART() function returns an integer, the expression evaluates to 2019 (2018 + 1). However, the DATENAME() function returns a character string, therefore, the + is the concatenation operator which results in '20181' (2018 + 1).

## SQL Server DATENAME() function example

This example uses the DATENAME() function to return various date parts of the '2020-10-02 10:20:30.1234567 +08:10':

**DECLARE** @dt DATETIME2= '2020-10-02 10:20:30.1234567 +08:10';

**SELECT** 'year,yyy,yy' date\_part,

**DATENAME**(**year**, @dt) **result**

**UNION**

**SELECT** 'quarter, qq, q',

**DATENAME**(**quarter**, @dt)

**UNION**

**SELECT** 'month, mm, m',

**DATENAME**(**month**, @dt)

**UNION**

**SELECT** 'dayofyear, dy, y',

**DATENAME**(**dayofyear**, @dt)

**UNION**

**SELECT** 'day, dd, d',

**DATENAME**(**day**, @dt)

**UNION**

**SELECT** 'week, wk, ww',

**DATENAME**(**week**, @dt)

**UNION**

**SELECT** 'weekday, dw, w',

**DATENAME**(**weekday**, @dt)

**UNION**

**SELECT** 'hour, hh' date\_part,

**DATENAME**(**hour**, @dt)

**UNION**

**SELECT** 'minute, mi,n',

**DATENAME**(**minute**, @dt)

**UNION**

**SELECT** 'second, ss, s',

**DATENAME**(**second**, @dt)

**UNION**

**SELECT** 'millisecond, ms',

**DATENAME**(millisecond, @dt)

**UNION**

**SELECT** 'microsecond, mcs',

**DATENAME**(**microsecond**, @dt)

**UNION**

**SELECT** 'nanosecond, ns',

**DATENAME**(nanosecond, @dt)

**UNION**

**SELECT** 'TZoffset, tz',

**DATENAME**(tz, @dt)

**UNION**

**SELECT** 'ISO\_WEEK, ISOWK, ISOWW',

**DATENAME**(ISO\_WEEK, @dt);

Here is the output:

date\_part result

---------------------- -----------

day, dd, d 2

dayofyear, dy, y 276

hour, hh 10

ISO\_WEEK, ISOWK, ISOWW 40

microsecond, mcs 123456

millisecond, ms 123

minute, mi,n 20

month, mm, m October

nanosecond, ns 123456700

quarter, qq, q 4

second, ss, s 30

TZoffset, tz +00:00

week, wk, ww 40

weekday, dw, w 40

year,yyy,yy 2020

(15 rows affected)

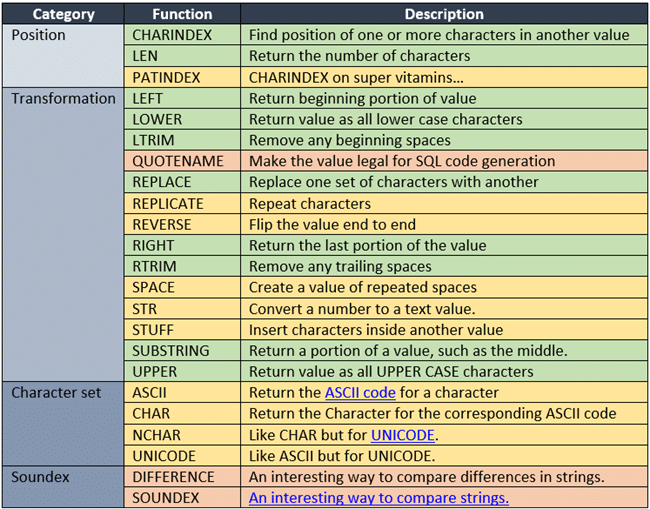
The SQL statements used below to return the different date formats use the SYSDATETIME() date function, which is new to SQL Server 2008.  The SYSDATETIME() function returns a datetime2(7) value that contains the date and time of the computer on which the instance of SQL Server is running.  The SYSDATETIME() function used below can be replaced by the GETDATE() or GETUTCDATE() functions.  The results will be the same unless the date format includes the nanosecond portion of the time.

To make the date format results consistent, the date and time used to generate the sample output is June 8, 2011 1:30:45.9428675 PM.

|  |  |  |  |
| --- | --- | --- | --- |
| **Standard Date Formats** | | | |
| **Date Format** | **Standard** | **SQL Statement** | **Sample Output** |
| Mon DD YYYY 1 HH:MIAM (or PM) | Default | SELECT CONVERT(VARCHAR(20), SYSDATETIME(), 100) | Jun 8 2011 1:30PM 1 |
| MM/DD/YY | USA | SELECT CONVERT(VARCHAR(8), SYSDATETIME(), 1) AS [MM/DD/YY] | 06/08/11 |
| MM/DD/YYYY | USA | SELECT CONVERT(VARCHAR(10), SYSDATETIME(), 101) AS [MM/DD/YYYY] | 06/08/2011 |
| YY.MM.DD | ANSI | SELECT CONVERT(VARCHAR(8), SYSDATETIME(), 2) AS [YY.MM.DD] | 11.06.08 |
| YYYY.MM.DD | ANSI | SELECT CONVERT(VARCHAR(10), SYSDATETIME(), 102) AS [YYYY.MM.DD] | 2011.06.08 |
| DD/MM/YY | British/French | SELECT CONVERT(VARCHAR(8), SYSDATETIME(), 3) AS [DD/MM/YY] | 08/06/11 |
| DD/MM/YYYY | British/French | SELECT CONVERT(VARCHAR(10), SYSDATETIME(), 103) AS [DD/MM/YYYY] | 08/06/2011 |
| DD.MM.YY | German | SELECT CONVERT(VARCHAR(8), SYSDATETIME(), 4) AS [DD.MM.YY] | 08.06.11 |
| DD.MM.YYYY | German | SELECT CONVERT(VARCHAR(10), SYSDATETIME(), 104) AS [DD.MM.YYYY] | 08.06.2011 |
| DD-MM-YY | Italian | SELECT CONVERT(VARCHAR(8), SYSDATETIME(), 5) AS [DD-MM-YY] | 08-06-11 |
| DD-MM-YYYY | Italian | SELECT CONVERT(VARCHAR(10), SYSDATETIME(), 105) AS [DD-MM-YYYY] | 08-06-2011 |
| DD Mon YY 1 | - | SELECT CONVERT(VARCHAR(9), SYSDATETIME(), 6) AS [DD MON YY] | 08 Jun 11 1 |
| DD Mon YYYY 1 | - | SELECT CONVERT(VARCHAR(11), SYSDATETIME(), 106) AS [DD MON YYYY] | 08 Jun 2011 1 |
| Mon DD, YY 1 | - | SELECT CONVERT(VARCHAR(10), SYSDATETIME(), 7) AS [Mon DD, YY] | Jun 08, 11 1 |
| Mon DD, YYYY 1 | - | SELECT CONVERT(VARCHAR(12), SYSDATETIME(), 107) AS [Mon DD, YYYY] | Jun 08, 2011 1 |
| HH:MM:SS | - | SELECT CONVERT(VARCHAR(8), SYSDATETIME(), 8) SELECT CONVERT(VARCHAR(8), SYSDATETIME(), 108) | 13:30:45 |
| Mon D YYYY H:MI:SS.NNNNNNNAM (or PM) 1 | Default + nanoseconds | SELECT CONVERT(VARCHAR(26), SYSDATETIME(), 9) SELECT CONVERT(VARCHAR(26), SYSDATETIME(), 109) | Jun 8 2011 1:30:45.9428675PM 1 |
| MM-DD-YY | USA | SELECT CONVERT(VARCHAR(8), SYSDATETIME(), 10) AS [MM-DD-YY] | 06-08-11 |
| MM-DD-YYYY | USA | SELECT CONVERT(VARCHAR(10), SYSDATETIME(), 110) AS [MM-DD-YYYY] | 06-08-2011 |
| YY/MM/DD | - | SELECT CONVERT(VARCHAR(8), SYSDATETIME(), 11) AS [YY/MM/DD] | 11/06/08 |
| YYYY/MM/DD | - | SELECT CONVERT(VARCHAR(10), SYSDATETIME(), 111) AS [YYYY/MM/DD] | 2011/06/08 |
| YYMMDD | ISO | SELECT CONVERT(VARCHAR(6), SYSDATETIME(), 12) AS [YYMMDD] | 110608 |
| YYYYMMDD | ISO | SELECT CONVERT(VARCHAR(8), SYSDATETIME(), 112) AS [YYYYMMDD] | 20110608 |
| DD Mon YYYY HH:MM:SS.NNNNNNN(24h) 1 | Europe default + nanoseconds | SELECT CONVERT(VARCHAR(30), SYSDATETIME(), 13) SELECT CONVERT(VARCHAR(30), SYSDATETIME(), 113) | 08 Jun 2011 13:30:45.94286751 |
| HH:MI:SS.NNNNNNN(24H) | - | SELECT CONVERT(VARCHAR(16), SYSDATETIME(), 14) AS [HH:MI:SS:MMM(24H)] SELECT CONVERT(VARCHAR(16), SYSDATETIME(), 114) AS [HH:MI:SS:MMM(24H)] | 13:30:45.9428675 |
| YYYY-MM-DD HH:MI:SS(24h) | ODBC Canonical | SELECT CONVERT(VARCHAR(19), SYSDATETIME(), 120) | 2011-06-08 13:30:45 |
| YYYY-MM-DD HH:MI:SS.NNNNNNN(24h) | ODBC Canonical (with nanoseconds) | SELECT CONVERT(VARCHAR(23), SYSDATETIME(), 121) | 2011-06-08 13:30:45.9428675 |
| MM/DD/YY HH:MI:SS AM | - | SELECT CONVERT(VARCHAR(20), SYSDATETIME(), 22) | 06/08/11 1:30:45 PM |
| YYYY-MM-DD | - | SELECT CONVERT(VARCHAR(26), SYSDATETIME(), 23) | 2011-06-091 |
| HH:MI:SS (24h) | - | SELECT CONVERT(VARCHAR(8), SYSDATETIME(), 24) | 13:30:45 |
| YYYY-MM-DD HH:MI:SS.NNNNNNN | - | SELECT CONVERT(VARCHAR(26), SYSDATETIME(), 25) | 2011-06-08 13:30:45.94286751 |
| YYYY-MM-DDTHH:MM:SS:NNNNNNN | ISO8601 | SELECT CONVERT(VARCHAR(27), SYSDATETIME(), 126) | 2011-06-08T13:30:45.9428675 |
| DD Mon YYYY HH:MI:SS.NNNNNNNAM 1 | Kuwaiti | SELECT CONVERT(VARCHAR(26), SYSDATETIME(), 130) | 08 Jun 2011 1:30:45.9428675PM1 |
| DD/MM/YYYY HH:MI:SS.NNNNNNNAM | Kuwaiti | SELECT CONVERT(VARCHAR(25), SYSDATETIME(), 131) | 08/06/2011 |

Top of Form

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Here are some more date formats that does not come standard in SQL Server as part of the **CONVERT** function.   |  |  |  | | --- | --- | --- | | **Extended Date Formats** | | | | **Date Format** | **SQL Statement** | **Sample Output** | | YY-MM-DD | SELECT RIGHT(CONVERT(VARCHAR(10), SYSDATETIME(), 20), 8) AS [YY-MM-DD]  SELECT REPLACE(CONVERT(VARCHAR(8), SYSDATETIME(), 11), '/', '-') AS [YY-MM-DD] | 11-06-08 | | YYYY-MM-DD | SELECT CONVERT(VARCHAR(10), SYSDATETIME(), 120) AS [YYYY-MM-DD]  SELECT REPLACE(CONVERT(VARCHAR(10), SYSDATETIME(), 111), '/', '-') AS [YYYY-MM-DD] | 2011-06-08 | | YYYY-M-D | SELECT CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)) + '-' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '-' + CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) AS [YYYY-M-D] | 2011-6-8 | | YY-M-D | SELECT RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) + '-' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '-' + CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) AS [YY-M-D] | 11-6-8 | | M-D-YYYY | SELECT CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '-' + CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + '-' + CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)) AS [M-D-YYYY] | 6-8-2011 | | M-D-YY | SELECT CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '-' + CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + '-' + RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) AS [M-D-YY] | 6-8-11 | | D-M-YYYY | SELECT CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + '-' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '-' + CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)) AS [D-M-YYYY] | 8-6-2011 | | D-M-YY | SELECT CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + '-' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '-' + RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) AS [D-M-YY] | 8-6-11 | | YY-MM | SELECT RIGHT(CONVERT(VARCHAR(7), SYSDATETIME(), 20), 5) AS [YY-MM] SELECT SUBSTRING(CONVERT(VARCHAR(10), SYSDATETIME(), 120), 3, 5) AS [YY-MM] | 11-06 | | YYYY-MM | SELECT CONVERT(VARCHAR(7), SYSDATETIME(), 120) AS [YYYY-MM] | 2011-06 | | YY-M | SELECT RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) + '-' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) AS [YY-M] | 11-6 | | YYYY-M | SELECT CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)) + '-' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) AS [YYYY-M] | 2011-6 | | MM-YY | SELECT RIGHT(CONVERT(VARCHAR(8), SYSDATETIME(), 5), 5) AS [MM-YY] SELECT SUBSTRING(CONVERT(VARCHAR(8), SYSDATETIME(), 5), 4, 5) AS [MM-YY] | 06-11 | | MM-YYYY | SELECT RIGHT(CONVERT(VARCHAR(10), SYSDATETIME(), 105), 7) AS [MM-YYYY] | 06-2011 | | M-YY | SELECT CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '-' + RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) AS [M-YY] | 6-11 | | M-YYYY | SELECT CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '-' + CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)) AS [M-YYYY] | 6-2011 | | MM-DD | SELECT CONVERT(VARCHAR(5), SYSDATETIME(), 10) AS [MM-DD] | 06-08 | | DD-MM | SELECT CONVERT(VARCHAR(5), SYSDATETIME(), 5) AS [DD-MM] | 08-06 | | M-D | SELECT CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '-' + CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) AS [M-D] | 6-8 | | D-M | SELECT CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + '-' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) AS [D-M] | 8-6 | | M/D/YYYY | SELECT CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '/' + CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + '/' + CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)) AS [M/D/YYYY] | 6/8/2011 | | M/D/YY | SELECT CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '/' + CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + '/' + RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) AS [M/D/YY] | 6/8/11 | | D/M/YYYY | SELECT CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + '/' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '/' + CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)) AS [D/M/YYYY] | 8/6/2011 | | D/M/YY | SELECT CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + '/' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '/' + RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) AS [D/M/YY] | 8/6/11 | | YYYY/M/D | SELECT CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)) + '/' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '/' + CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) AS [YYYY/M/D] | 2011/6/8 | | YY/M/D | SELECT RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) + '/' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '/' + CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) AS [YY/M/D] | 11/6/8 | | MM/YY | SELECT RIGHT(CONVERT(VARCHAR(8), SYSDATETIME(), 3), 5) AS [MM/YY] | 06/11 | | MM/YYYY | SELECT RIGHT(CONVERT(VARCHAR(10), SYSDATETIME(), 103), 7) AS [MM/YYYY] | 06/2011 | | M/YY | SELECT CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '/' + RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) AS [M/YY] | 6/11 | | M/YYYY | SELECT CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '/' + CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)) AS [M/YYYY] | 6/2011 | | YY/MM | SELECT CONVERT(VARCHAR(5), SYSDATETIME(), 11) AS [YY/MM] | 11/06 | | YYYY/MM | SELECT CONVERT(VARCHAR(7), SYSDATETIME(), 111) AS [YYYY/MM] | 2011/06 | | YY/M | SELECT RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) + '/' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) AS [YY/M] | 11/6 | | YYYY/M | SELECT CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)) + '/' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) AS [YYYY/M] | 2011/6 | | MM/DD | SELECT CONVERT(VARCHAR(5), SYSDATETIME(), 1) AS [MM/DD] | 06/08 | | DD/MM | SELECT CONVERT(VARCHAR(5), SYSDATETIME(), 3) AS [DD/MM] | 08/06 | | M/D | SELECT CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '/' + CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) AS [M/D] | 6/8 | | D/M | SELECT CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + '/' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) AS [D/M] | 8/6 | | MM.DD.YYYY | SELECT REPLACE(CONVERT(VARCHAR(10), SYSDATETIME(), 101), '/', '.') AS [MM.DD.YYYY] | 06.08.2011 | | MM.DD.YY | SELECT REPLACE(CONVERT(VARCHAR(8), SYSDATETIME(), 1), '/', '.') AS [MM.DD.YY] | 06.08.11 | | M.D.YYYY | SELECT CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '.' + CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + '.' + CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)) AS [M.D.YYYY] | 6.8.2011 | | M.D.YY | SELECT CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '.' + CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + '.' + RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) AS [M.D.YY] | 6.8.11 | | DD.MM.YYYY | SELECT CONVERT(VARCHAR(10), SYSDATETIME(), 104) AS [DD.MM.YYYY] | 08.06.2011 | | DD.MM.YY | SELECT CONVERT(VARCHAR(10), SYSDATETIME(), 4) AS [DD.MM.YY] | 08.06.11 | | D.M.YYYY | SELECT CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + '.' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '.' + CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)) AS [D.M.YYYY] | 8.6.2011 | | D.M.YY | SELECT CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + '.' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '.' + RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) AS [D.M.YY] | 8.6.11 | | YYYY.M.D | SELECT CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)) + '.' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '.' + CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) AS [YYYY.M.D] | 2011.6.8 | | YY.M.D | SELECT RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) + '.' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '.' + CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) AS [YY.M.D] | 11.6.8 | | MM.YYYY | SELECT RIGHT(CONVERT(VARCHAR(10), SYSDATETIME(), 104), 7) AS [MM.YYYY] | 06.2011 | | MM.YY | SELECT RIGHT(CONVERT(VARCHAR(8), SYSDATETIME(), 4), 5) AS [MM.YY] | 06.11 | | M.YYYY | SELECT CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '.' + CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)) AS [M.YYYY] | 6.2011 | | M.YY | SELECT CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) + '.' + RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) AS [M.YY] | 6.11 | | YYYY.MM | SELECT CONVERT(VARCHAR(7), SYSDATETIME(), 102) AS [YYYY.MM] | 2011.06 | | YY.MM | SELECT CONVERT(VARCHAR(5), SYSDATETIME(), 2) AS [YY.MM] | 11.06 | | YYYY.M | SELECT CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)) + '.' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) AS [YYYY.M] | 2011.6 | | YY.M | SELECT RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) + '.' + CAST(MONTH(SYSDATETIME()) AS VARCHAR(2)) AS [YY.M] | 11.6 | | MM.DD | SELECT RIGHT(CONVERT(VARCHAR(8), SYSDATETIME(), 2), 5) AS [MM.DD] | 06.08 | | DD.MM | SELECT CONVERT(VARCHAR(5), SYSDATETIME(), 4) AS [DD.MM] | 08.06 | | MMDDYYYY | SELECT REPLACE(CONVERT(VARCHAR(10), SYSDATETIME(), 101), '/', '') AS [MMDDYYYY] | 06082011 | | MMDDYY | SELECT REPLACE(CONVERT(VARCHAR(8), SYSDATETIME(), 1), '/', '') AS [MMDDYY] | 060811 | | DDMMYYYY | SELECT REPLACE(CONVERT(VARCHAR(10), SYSDATETIME(), 103), '/', '') AS [DDMMYYYY] | 08062011 | | DDMMYY | SELECT REPLACE(CONVERT(VARCHAR(8), SYSDATETIME(), 3), '/', '') AS [DDMMYY] | 080611 | | MMYYYY | SELECT RIGHT(REPLACE(CONVERT(VARCHAR(10), SYSDATETIME(), 103), '/', ''), 6) AS [MMYYYY] | 062011 | | MMYY | SELECT RIGHT(REPLACE(CONVERT(VARCHAR(8), SYSDATETIME(), 3), '/', ''), 4) AS [MMYY] | 0611 | | YYYYMM | SELECT CONVERT(VARCHAR(6), SYSDATETIME(), 112) AS [YYYYMM] | 201106 | | YYMM | SELECT CONVERT(VARCHAR(4), SYSDATETIME(), 12) AS [YYMM] | 1106 | | Month DD, YYYY 1 | SELECT DATENAME(MONTH, SYSDATETIME())+ ' ' + RIGHT('0' + DATENAME(DAY, SYSDATETIME()), 2) + ', ' + DATENAME(YEAR, SYSDATETIME()) AS [Month DD, YYYY] | June 08, 2011 1 | | Mon YYYY 1 | SELECT LEFT(DATENAME(MONTH, SYSDATETIME()), 3) + ' ' + DATENAME(YEAR, SYSDATETIME()) AS [Mon YYYY] | Jun 2011 1 | | Month YYYY 1 | SELECT DATENAME(MONTH, SYSDATETIME()) + ' ' + DATENAME(YEAR, SYSDATETIME()) AS [Month YYYY] | June 2011 1 | | DD Month 1 | SELECT RIGHT('0' + DATENAME(DAY, SYSDATETIME()), 2) + ' ' + DATENAME(MONTH, SYSDATETIME()) AS [DD Month] | 08 June 1 | | Month DD 1 | SELECT DATENAME(MONTH, SYSDATETIME()) + ' ' + RIGHT('0' + DATENAME(DAY, SYSDATETIME()), 2) AS [Month DD] | June 08 1 | | DD Month YY 1 | SELECT CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + ' ' + DATENAME(MM, SYSDATETIME()) + ' ' + RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) AS [DD Month YY] | 08 June 11 1 | | DD Month YYYY 1 | SELECT RIGHT('0' + DATENAME(DAY, SYSDATETIME()), 2) + ' ' + DATENAME(MONTH, SYSDATETIME())+ ' ' + DATENAME(YEAR, SYSDATETIME()) AS [DD Month YYYY] | 08 June 2011 1 | | Mon-YY 1 | SELECT REPLACE(RIGHT(CONVERT(VARCHAR(9), SYSDATETIME(), 6), 6), ' ', '-') AS [Mon-YY] | Jun-08 1 | | Mon-YYYY 1 | SELECT REPLACE(RIGHT(CONVERT(VARCHAR(11), SYSDATETIME(), 106), 8), ' ', '-') AS [Mon-YYYY] | Jun-2011 1 | | DD-Mon-YY 1 | SELECT REPLACE(CONVERT(VARCHAR(9), SYSDATETIME(), 6), ' ', '-') AS [DD-Mon-YY] | 08-Jun-11 1 | | DD-Mon-YYYY 1 | SELECT REPLACE(CONVERT(VARCHAR(11), SYSDATETIME(), 106), ' ', '-') AS [DD-Mon-YYYY] | 08-Jun-2011 1 |   1 To make the month name in upper case, simply use the UPPER string function. | | **Related Articles :** | |  | | |

Bottom of Form