Date and Time Functions in SQL Server

Ø YEAR, MONTH, DAY – Return a part of the date (year, month, or day) from a date-time.

Ø DATEPART, DATENAME – Return a part of a date-time.

Ø GETDATE, CURRENT_TIMESTAMP, .. - Get Current Date and Time.

Ø DATEADD – Add a part of a date-time.

Ø DATEDIFF, DATEDIFF_BIG – Find the difference between two date-times.

Ø DATEFROMPARTS, TIMEFROMPARTS, … - Construct date or time from parts.

Ø ISDATE – Check for a valid date.

Ø EOMONTH – Return the end of the month.

Ø SWITCHOFFSET – Show a different time zone.

Ø TODATETIMEOFFSET – Change the original time zone.

YEAR, MONTH, DAY

YEAR, MONTH, and DAY functions are used to find the year, month, and day of a date-time.

YEAR

YEAR function returns year in yyyy format from date or date-time value.

Syntax:

```
SELECT YEAR(Datetime1);
```

The following script returns Year1 from date-time and Year2 from date.

Script:

```
SELECT YEAR('2020-11-22 14:35:55') AS Year1, YEAR('2019-02-15') AS Year2;
```

Output:

MONTH

MONTH function returns month from a date or date-time value.

Syntax:

```
SELECT MONTH(Datetime1);
```

The following script returns Month1 from date-time and Month2 from date.

Script:

```
SELECT MONTH('2020-11-22 14:35:55') AS Month1, MONTH('2019-02-15') AS Month2;
```

Output:

DAY

Day function returns Day from date-time or date.

Syntax:

```
SELECT DAY(Datetime1);
```

The following script returns the day1 from date-time and day2 from date.

Script:

```
SELECT DAY('2020-11-22 14:35:55') AS Day1, DAY('2019-02-03') AS Day2;
```

Output:

DATEPART, DATENAME

DATEPART and DATENAME functions are used to return a part of a date-time.

DATEPART

DATEPART function can be used to get the part of a date-time.

Syntax:

```
SELECT DATEPART(Parameter1, Datetime1);
```

Parameter1 will be any parameter from the below table.

Table 1:

Parameters			Description
year	yyyy	yy	Year
quarter	qq	q	Quarter
month	mm	m	Month
dayofyear	dy	y	Day of the Year
day	dd	d	Day of the Month
week	ww	wk	Week
weekday	dw	w	Weekday
hour	hh		Hour

minute	mi	n	Minute
second	ss	s	Second
millisecond	ms		Millisecond
microsecond	mcs		Microsecond
nanosecond	ns		Nanosecond
tz			Tz offset
isowk,	isoww		ISO week

The following script returns a year, quarter, month, day of the year, day, week, weekday, hour, minute, second, millisecond, microsecond, nanosecond, Tz offset, ISO Week from given date-time.

Script:

```sql
SELECT DATEPART(yyyy , '2020-11-22 14:35:55.1234567 + 05:11') AS [Year]

, DATEPART(mm , '2020-11-22 14:35:55.1234567 + 05:11') AS [Month]

, DATEPART(q, '2020-11-22 14:35:55.1234567 + 05:11') AS [Quater]

, DATEPART(dy, '2020-11-22 14:35:55.1234567 + 05:11') AS [Day of Year]

, DATEPART(dd, '2020-11-22 14:35:55.1234567 + 05:11') AS [Day of Month]

, DATEPART(ww, '2020-11-22 14:35:55.1234567 + 05:11') AS [Week]

, DATEPART(dw, '2020-11-22 14:35:55.1234567 + 05:11') AS [Week Day]

, DATEPART(hh, '2020-11-22 14:35:55.1234567 + 05:11') AS [Hour]

, DATEPART(mi, '2020-11-22 14:35:55.1234567 + 05:11') AS [Minute]

, DATEPART(ss, '2020-11-22 14:35:55.1234567 + 05:11') AS [Second]

, DATEPART(ms, '2020-11-22 14:35:55.1234567 + 05:11') AS [Millisecond]

, DATEPART(mcs, '2020-11-22 14:35:55.1234567 + 05:11') AS [Microsecond]

, DATEPART(ns, '2020-11-22 14:35:55.1234567 + 05:11') AS [Nanosecond]

, DATEPART(tz, '2020-11-22 14:35:55.1234567 + 05:11') AS [Tz Offset]

, DATEPART(isowk, '2020-11-22 14:35:55.1234567 + 05:11') AS [ISO Week]
```

Output:

DATENAME

The DATENAME function returns one part from a date-time.

Syntax:

```
SELECT DATENAME (Parameter1, DateTime1);
```

Parameter1 is a parameter from the table1.

The following script will find years, months, days, hours, minutes, seconds, milliseconds, microseconds, or nanoseconds from date-time.

Script:

```
SELECT DATENAME (yyyy, '2020-11-22 14:35:55.1234567') AS [Year]

, DATENAME(mm, '2020-11-22 14:35:55.1234567') AS [Month]

, DATENAME(dd, '2020-11-22 14:35:55.1234567') AS [Day]

, DATENAME(hh, '2020-11-22 14:35:55.1234567') AS [Hour]

, DATENAME(mi, '2020-11-22 14:35:55.1234567') AS [Minute]

, DATENAME(ss, '2020-11-22 14:35:55.1234567') AS [Second]

, DATENAME(ms, '2020-11-22 14:35:55.1234567') AS [Millisecond]

, DATENAME(mcs, '2020-11-22 14:35:55.1234567') AS [Microsecond]

, DATENAME(ns, '2020-11-22 14:35:55.1234567') AS [Nanosecond]
```

Output:

Both DATEPART and DATENAME returns a part of a date-time. ButDATEPART function returns an integer and DATENAME returns a string.

Functions to Get Current Date and Time

GETDATE, CURRENT_TIMESTAMP, GETUTCDATE, SYSDATETIME, SYSUTCDATETIME, SYSDATETIMEOFFSET functions return the current date and time.

GETDATE

The GETDATE function returns the current date and time in YYYY-MM-DD hh:mm:ss.mmm format.

Syntax:

```
SELECT GETDATE();
```

The following script returns the current date and time in YYYY-MM-DD hh:mm:ss.mmm format.

Script:

```
SELECT GETDATE()
```

Output:

The output will depend on the current date and the time.

CURRENT_TIMESTAMP

The CURRENT_TIMESTAMP function returns the current date and time in YYYY-MM-DD hh:mm:ss.mmm format.

It is almost the same as GETDATE. It returns the same result as GETDATE. It is an ANSI equivalent to GETDATE.

Syntax:

```
SELECT CURRENT_TIMESTAMP;
```

The following script returns the current date and time in YYYY-MM-DD hh:mm:ss.mmm format.

Script:

```
SELECT CURRENT_TIMESTAMP;
```

Output:

The output will depend on the current date and the time.

GETDATE and CURRENT_TIMESTAMP functions return the same output.

GETUTCDATE

The GETUTCDATE function returns the current date and time in YYYY-MM-DD hh:mm:ss.mmm format.

Syntax:

```
SELECT GETUTCDATE();
```

The following script returns the current date and time in YYYY-MM-DD hh:mm:ss.mmm format.

Script:

```
SELECT GETUTCDATE();
```

Output:

The output will depend on the date and the time.

SYSDATETIME

The SYSDATETIME function returns the current date and time in YYYY-MM-DD hh:mm:ss.ns format.

Syntax:

```
SELECT SYSDATETIME();
```

The following script returns the current date and time in YYYY-MM-DD hh:mm:ss.ns format.

Script:

```
SELECT SYSDATETIME ();
```

Output:

The output will depend on the date and the time

SYSUTCDATETIME

The SYSUTCDATETIME function returns the current date and time in YYYY-MM-DD hh:mm:ss.ns format.

Syntax:

```
SELECT SYSUTCDATETIME();
```

The following script returns the current date and time in YYYY-MM-DD hh:mm:ss.ns format.

Script:

```
SELECT SYSUTCDATETIME();
```

Output:

The output will depend on the date and the time

SYSDATETIMEOFFSET

SYSDATETIMEOFFSET function returns the current date-time in YYYY-MM-DD hh:mm: ss.ns +/- hh.mm format.

Syntax:

```
SELECT SYSDATETIMEOFFSET();
```

The following script returns the current date and time in YYYY-MM-DD hh:mm:ss.ns +/- hh.mm format.

Script:

```
SELECT SYSDATETIMEOFFSET();
```

Output:

The output will depend on the date and the time

DATEADD

The DATEADD function adds one of the below-listed dates or time parameters to date.

Syntax:

`SELECT DATEADD(Parameter1, Value1, Date1);`

Parameter1 is a parameter from the given list, Value1 is the value of the parameter.

The following script will add part of a year (year, month, day, hour, minute, second, millisecond) to a given date.

Script:

```
SELECT DATEADD(yyyy, 1, '2020-11-22 14:35:55.123') AS Date1

, DATEADD(mm,  1, '2020-11-22 14:35:55.123') AS Date2

, DATEADD(dd, 1, '2020-11-22 14:35:55.123') AS Date5

, DATEADD(ww, 1, '2020-11-22 14:35:55.123') AS Date6

, DATEADD(hh, 1, '2020-11-22 14:35:55.123') AS Date8

, DATEADD(mi, 1, '2020-11-22 14:35:55.123') AS Date9

, DATEADD(ss, 1, '2020-11-22 14:35:55.123') AS Date10

, DATEADD(ms, 1, '2020-11-22 14:35:55.123') AS Date11
```

Output:

Parameter1 will be any parameter from the below table.

Table 2:

Parameters			**Description**
year	yyyy	yy	Year
quarter	qq	q	Quarter
month	mm	m	Month
dayofyear	dy	y	Day of the Year

day	dd	d	Day of the Month
week	ww	wk	Week
weekday	dw		w
hour	hh		Hour
minute	mi	n	Minute
second	ss	s	Second
millisecond	ms		Millisecond
microsecond	mcs		Microsecond
nanosecond	ns		Nanosecond

This script gives an error. `SELECT DATEADD(ms, 1, '2020-11-22 14:35:55.1111') AS Date11`

The error is corrected in the following script.

```
DECLARE @Datetime2 datetime2='2020-11-22 14:35:55.1111111'; SELECT
DATEADD(ns,111,@datetime2);
```

`datetime2` data type should be used to add millisecond and microseconds.

DATEDIFF, DATEDIFF_BIG

DATEDIFF and DATEDIFF_BIG functions are used to find the difference between two dates.

DATEDIFF

The DATEDIFF function finds the difference between two dates (start date, end date). It returns an integer.

Syntax:

```
SELECT DATEDIFF (Parameter1, StartDate1, EndDate1);
```

Parameter1 is a parameter from the given list, StartDate1 is the start date, EndDate1 is the end date.

The following script will find the difference between two days in years, months, days, hours, minutes, seconds, milliseconds, microseconds, and nanoseconds.

Script:

```
SELECT DATEDIFF(yyyy, '2020-11-22 14:35:55.123', '2021-12-23 15:36:56.123') AS
Years

, DATEDIFF(mm, '2020-11-22 14:35:55.123', '2021-12-23 15:36:56.123') AS Months

, DATEDIFF(dd, '2020-11-22 14:35:55.123', '2021-12-23 15:36:56.123') AS [Days]

, DATEDIFF(hh, '2020-11-22 14:35:55.123', '2021-12-23 15:36:56.123') AS [Hours]

, DATEDIFF(mi, '2020-11-22 14:35:55.123', '2021-12-23 15:36:56.123') AS [Minutes]

, DATEDIFF(ss, '2020-11-22 14:35:55.1111111', '2020-11-22 14:35:55.2222222') AS
Seconds

, DATEDIFF(ms, '2020-11-22 14:35:55.1111111', '2020-11-22 14:35:55.2222222') AS
Milliseconds

, DATEDIFF(mcs, '2020-11-22 14:35:55.1111111', '2020-11-22 14:35:55.2222222') AS
MicroSeconds

, DATEDIFF(ns, '2020-11-22 14:35:55.1111111', '2020-11-22 14:35:55.2222222') AS
NanoSeconds
```

Output:

DATEDIFF_BIG

The DATEDIFF_BIG function finds the difference between two dates (start date, end date). It returns a big integer.

Syntax:

`SELECT DATEDIFF_BIG (Parameter1, StartDate1, EndDate1);`

Parameter1 is a parameter from the given list, StartDate1 is the start date, EndDate1 is the end date.

The following script will find the difference between two days in seconds, milliseconds, microseconds, and nanoseconds.

Script:

```
SELECT DATEDIFF_BIG(ss, '2000-01-01 01:01:01.0000001', '2069-01-01
01:01:01.0000001') AS SS

, DATEDIFF_BIG(ms, '2000-01-01 01:01:01.0000001', '2000-01-26 01:01:01.0000001') AS
MS

, DATEDIFF_BIG(mcs, '2000-01-01 01:01:01.0000001', '2000-01-01 01:37:01.0000001')
AS MCS

, DATEDIFF_BIG(ns, '2000-01-01 01:01:01.0000001', '2000-01-01 01:01:04.0000001') AS
NS
```

Output:

The following script will give an error as integer type is not enough to return data.

Script:

```
SELECT DATEDIFF(ss, '2000-01-01 01:01:01.0000001', '2069-01-01 01:01:01.0000001')
AS SS;

SELECT DATEDIFF(ms, '2000-01-01 01:01:01.0000001', '2000-01-26 01:01:01.0000001')
AS MS;

SELECT DATEDIFF(mcs, '2000-01-01 01:01:01.0000001', '2000-01-01 01:37:01.0000001')
AS MCS;

SELECT DATEDIFF(ns, '2000-01-01 01:01:01.0000001', '2000-01-01 01:01:04.0000001')
AS NS;
```

DATEDIFF returns an integer and DATEDIFF_BIG returns a big integer. You should use DATEDIFF_BIG when return high voumn of data.

Functions to construct date-time from parts

DATEFROMPARTS, DATETIMEFROMPARTS, and DATETIME2FROMPARTS functions are used to construct a date or and date-time.

DATEFROMPARTS

The DATEFROMPARTS function constructs a date using year, month, and day.

Syntax:

```
SELECT DATEFROMPARTS (Year1, Month1, Day1);
```

The following script constructs a date using year, month, and day.

Script:

```
SELECT DATEFROMPARTS (2020, 12, 22);
```

Output:

TIMEFROMPARTS

The TIMEFROMPARTS function constructs a date using year, month, and day.

Syntax:

```
SELECT TIMEFROMPARTS (Hour1, Minute1, Seconds1, Fraction1, Precisions1);
```

The following script constructs a date using year, month, and day.

Script:

```
SELECT TIMEFROMPARTS (14, 30, 10, 1234567, 7);
```

```
```

Output:

SMALLDATETIMEFROMPARTS

The SMALLDATETIMEFROMPARTS function constructs a date-time using year, month, day, hour, and minute.

Syntax:

```
SELECT SMALLDATETIMEFROMPARTS (Year1, Month1, Day1, Hour1, Minute1);
```

The following script constructs a date-time using year, month, day, hour, and minute.

Script:

```
SELECT SMALLDATETIMEFROMPARTS (2020, 12, 22, 14, 30);
```

Output:

DATETIMEFROMPARTS

The DATETIMEFROMPARTS function constructs a date-time using year, month, day, hour, minute, second, and millisecond.

Syntax:

`SELECT DATETIMEFROMPARTS (`Year1, Month1, Day1, Hour1, Minute1, Second1, Millisecond1`);`

The following script constructs a date-time using years, months, days, hours, minutes, seconds.

Script:

```
SELECT DATETIMEFROMPARTS (2020, 12, 22, 14, 30, 50, 0);
```

Output:

DATETIME2FROMPARTS

The DATETIMEFROMPARTS function constructs a date-time using year, month, day, hour, minute, and second.

Syntax:

`SELECT DATETIME2FROMPARTS (`Year1, Month1, Day1, Hour1, Minute1, Second1, Fraction1, Precision1`);`

Fraction1 is a fractional second in integer.

Precision1 is the precisions of Fraction1 in integer.

The following script constructs a date-time using years, months, days, hours, minutes, seconds, and fractional seconds.

Script:

```
SELECT DATETIME2FROMPARTS (2020, 12, 22, 14, 30, 50, 1234567, 7);
```

Output:

DATEDIFF returns an integer and DATEDIFF_BIG returns a big integer. You should use DATEDIFF_BIG when you return high voume of data.

DATETIMEOFFSETFROMPARTS

The DATETIMEOFFSETFROMPARTS function constructs a date-time using year, month, day, hour, minute, second, time zone hour, and time zone minutes.

Syntax:

```
SELECT DATETIMEOFFSETFROMPARTS (Year1, Month1, Day1, Hour1, Minute1, Second1, Fraction1,
HourOffset1, MinuteOffset1, Precision1);
```

Fraction1 is the fractional seconds as an integer.

Precision1 is the number of decimals of Fraction1 as an integer.

HourOffset1 is the hour portion of the time zone.

MinuteOffset1 is the minute portion of the time zone.

The following script constructs a date-time using year, month, day, hours, minutes, seconds, fractional seconds, and time zone details.

Script:

```
SELECT DATETIMEOFFSETFROMPARTS (2020, 12, 22, 14, 30, 50, 1234567, 2, 0, 7);
```

Output:

ISDATE

ISDATE function checks for the validity of date, time, or data-time. It returns 1 when it is true. Otherwise, return 0.

Syntax:

```
SELECT ISDATE (DateTime1);
```

The following script will show how to check the validity of a date.

Script:

```
SELECT ISDATE('2020-12-31') AS Date1, ISDATE('2020-12-32') AS Date2;
```

Output:

EOMONTH

EOMONTH function returns the end date of the month.

Syntax:

```
SELECT EOMONTH (DateTime1);
```

The following script returns the end date of this month, last month, and next month.

Script:

```
DECLARE @date DATETIME = GETDATE(); SELECT EOMONTH (@date) AS 'This month-end'

, EOMONTH (@date, 1) AS 'Next month-end', EOMONTH (@date, -1) AS 'Last month-end';
```

Output:

SWITCHOFFSET

The SWITCHOFFSET function used to show a different time zone instead of the original. But the function will not update the original time zone.

Syntax:

```
SELECT SWITCHOFFSET (DateTime1);
```

The following script will show a different time zone instead of the original.

Script:

```
SELECT SWITCHOFFSET ('2020-11-22 14:35:55.1234567 + 05:11', '-08:00');
```

Output:

TODATETIMEOFFSET

The TODATETIMEOFFSET function is used to change the original time zone.

Syntax:

```
SELECT TODATETIMEOFFSET (DateTime1);
```

The following script shows two ways of changing the original time zone.

Script:

```
SELECT TODATETIMEOFFSET ('2020-11-22 14:35:55.1234567 + 05:11', '-04:00')

, TODATETIMEOFFSET ('2020-11-22 14:35:55.1234567 + 05:11', -240);
```

Output: