GEÇİCİ VERİLER İLE ÇALIŞMAK

6

SQL Server'da sık kullanılan ve işlevselliği yüksek olan tarih işlemleri için farklı veri tipleri ve fonksiyonlar geliştirilmiştir. Tarih işlemleri temel anlamda basit gibi görünse de, uygulamaların doğru oluşturulabilmesi için önemli bir konudur.

Bu bölümde, genel olarak geliştiricilerin detaylarına hakim olmadığı ve yeni başlayanların detaylarını kavramakta zorlandığı tarih işlemlerini inceleyeceğiz.

Veri girişi sırasında kullanılan tarih ve saat özelliklerinin yanı sıra, çıktı olarak alınacak formatların belirlenmesi, işlenmesi, dönüştürülmesi gibi konuları detaylandıracağız.

SQL Server Tarih / Zaman Veri Tipleri

Tarih işlemleri SQL Server'da veri tipleri bazında desteklenmektedir. Yani, tarih işlemlerini gerçekleştirmek için, sütunlarda birçok farklı tarih veri tipi kullanılabilir. Bu veri tipleri temel anlamda benzer işlemleri gerçekleştirse de, özellik olarak farklılıkları vardır.

DATE

Tarih tipinden veri saklamaya yarar. `01-01-0001` ile `31-12-9999` arasında tarih değeri alabilir.

Format:

```
YYYY-MM-DD
```

Varsayılan Değer:

```
1900-01-01
```

Örnek :

```
2007-05-08
```

Yukarıdaki formata uygun olarak bir değişken tanımı ve görüntülenmesini gerçekleştirelim.

```
DECLARE @date date = '2013-01-25';
SELECT @date AS '@date';
```

	@date
1	2013-01-25

Date ile **DateTime** arasındaki fark:

```
DECLARE @date date= '12-10-25';
DECLARE @datetime datetime= @date;

SELECT @date AS '@date', @datetime AS '@datetime';
```

	@date	@datetime
1	2025-12-10	2025-12-10 00:00:00.000

TIME

Saat tipinden veri saklamaya yarar. 5 baytlık depolama alanına sahiptir. Nano saniye bölümü 7 digit değerden oluşur. Tablo oluştururken nano saniye kısmında farklı değer belirterek nano saniye hassasiyeti belirlenebilmektedir.

Time(3) kullanımı, nano saniye değerinin 3 rakamlı hassasiyetini belirtir. **Time(7)** kullanımı ise, 7 rakamlı nano saniye hassasiyetini belirtir.

Format :

```
hh:mm:ss[.nnnnnnn]
```

Varsayılan Değer :

```
00:00:00
```

Örnek :

```
12:35:29.1234567
```

Örnek bir saat bilgisi üzerinde **Time** veri tipini kullanalım.

```
DECLARE @time time(7) = '12:34:54.1234567';
DECLARE @time1 time(1) = @time;
DECLARE @time2 time(2) = @time;
DECLARE @time3 time(3) = @time;
DECLARE @time4 time(4) = @time;
DECLARE @time5 time(5) = @time;
DECLARE @time6 time(6) = @time;
DECLARE @time7 time(7) = @time;

SELECT
     @time1 AS 'time(1)', @time2 AS 'time(2)', @time3 AS 'time(3)',
     @time4 AS 'time(4)', @time5 AS 'time(5)', @time6 AS 'time(6)',
     @time7 AS 'time(7)';
```

	time(1)	time(2)	time(3)	time(4)	time(5)	time(6)	time(7)
1	12:34:54.1	12:34:54.12	12:34:54.123	12:34:54.1235	12:34:54.12346	12:34:54.123457	12:34:54.1234567

Diğer veri tiplerine göre farklılıklarını gözlemleyelim.

Time ile **DateTime** arasındaki fark:

```
DECLARE @time time(4) = '12:15:04.1234';
DECLARE @datetime datetime= @time;

SELECT @time AS '@time', @datetime AS '@datetime';
```

	@time	@datetime
1	12:15:04.1234	1900-01-01 12:15:04.123

SmallDateTime

DateTime veri tipinin daha az kapsamlı halidir. Nano saniye değerini içermez ve **01-01-1900** ile **06-06-2079** tarihleri arasında değer alabilir. 4 baytlık veri tipidir.

Format:

```
YYYY-MM-DD hh:mm:ss
```

Varsayılan Değer:

```
1900-01-01 00:00:00
```

Örnek Değer :

```
2007-05-08 12:35:00
```

`SmallDateTime` ile **`Date`** arasındaki fark:

```
DECLARE @smalldatetime smalldatetime = '1955-12-13 12:43:10';
DECLARE @date date = @smalldatetime;

SELECT @smalldatetime AS '@smalldatetime', @date AS 'date';
```

	@smalldatetime	date
1	1955-12-13 12:43:00	1955-12-13

DateTime

Tarih ve saat tipinden veri saklamaya yarar. 8 baytlık veri tipidir. 01-01-1753 ile 31-12-9999 arasında bir tarih değer alabilir. Aynı zamanda nanosaniye olarak 3 digit'i desteklemektedir.

Format:

```
YYY-MM-DD hh:mm:ss[.nnn]
```

Varsayılan Değer:

```
1900-01-01 00:00:00
```

Örnek :

```
2013-05-05 12:35:29.123
```

`DateTime` ile **`SmallDateTime`** arasındaki fark:

```
DECLARE @smalldatetime smalldatetime = '1955-12-13 12:43:10';
DECLARE @datetime datetime = @smalldatetime;

SELECT @smalldatetime AS '@smalldatetime', @datetime AS '@datetime';
```

	@smalldatetime	@datetime
1	1955-12-13 12:43:00	1955-12-13 12:43:00.000

DATETIME2

DateTime veritipinin aynısıdır. Bir farkı, nano saniye olarak 7 digit bilgisini içermesi, diğeri ise **01-01-0001** ile **31-12-9999** tarihleri arasında değer almasıdır.

Format:

```
YYYY-MM-DD hh:mm:ss[.nnnnnnn]
```

Varsayılan Değer :

```
1900-01-01 00:00:00
```

Örnek :

```
2007-05-08 12:35:29. 1234567
```

DateTime2 ile **Date** arasındaki fark;

```
DECLARE @datetime2 datetime2(4) = '12-10-25 12:32:10.1234';
DECLARE @date date = @datetime2;
SELECT @datetime2 AS '@datetime2', @date AS 'date';
```

	@datetime2	date
1	2025-12-10 12:32:10.1234	2025-12-10

DATETIMEOFFSET

UTC (*Coordinated Universal Time = Eşitlenmiş Evrensel Zaman*) bilgisi içeren tarih-saat veri tipidir.

Format:

```
YYYY-MM-DD hh:mm:ss[.nnnnnnn] [{+|-}hh:mm]
```

Varsayılan Değer:

```
1900-01-01 00:00:00 00:00
```

Örnek :

```
2007-05-08 12:35:29.1234567 +12:15
```

DateTimeOffSet ile **Date** arasındaki fark:

```
DECLARE @datetimeoffset datetimeoffset(4) = '12-10-25 12:32:10 +01:0';
DECLARE @date date= @datetimeoffset;

SELECT @datetimeoffset AS '@datetimeoffset ', @date AS 'date';
```

	@datetimeoffset	date
1	2025-12-10 12:32:10.0000 +01:00	2025-12-10

Girdi Tarih Formatları

Veritabanı uygulamalarında gerçek veriler ile çalışırken, tarih-saat verilerinin yönetimi önem arz eder. Bir tarih işlemi için farklı istemcilerden gelebilecek farklı tarih-saat formatları veritabanında çeşitli sorgu ve prosedürler ile işlenerek veritabanının ilgili sütununa doğru ve tutarlı şekilde depolanması gerekir.

Farklı istemcilere sahip ve çok dilli uygulamaların tek bir veritabanını kullanıyor olması tarih-saat verilerinin doğru yönetilmesini gerektirir. İngiltere'den İngilizce dili ile istemci kullanan ile Türkiye'den Türkçe istemci kullanan bir platformun ortak tarihler üzerinde işlemler yapabilmesi ve görüntülemesi için veritabanının doğru tasarlanması ve sorguların bu kriterlere göre optimize edilmesi gerekir.

SQL Server, bu işlemleri kolaylaştırmak için birçok veri tipi ve fonksiyon kullanır. Ancak, öncelikle bir veritabanına ne tür tarih girişi yapılabileceğine bakalım.

Bu tür karmaşık yapıdaki mimarilerin doğru tarih-saat veri formatlarını kullanabilmesi için Uluslararası Standartlar Örgütü, ISO 8601 standardını oluşturmuştur.

NOT ISO 8601 standardı, veri değişimi kapsamında; Gregorian takvimine göre tarihleri, 24 saatlik gösterime göre zamanı, zaman aralıklarını ve zaman aralıklarının yeniden gösterimi ile bu gösterimlerin formatlarını kapsar.

Standart tarih formatı olarak aşağıdaki format kullanılır.

```
yyyy-mm-dd hh:mi:ss.mmm
```

Başlangıçta anlamsız görünen bu dizilimin daha anlaşılabilir karşılığı aşağıdaki gibidir:

```
yıl-ay-gün saat:dakika:saniye:milisaniye
```

Şu anki tarihim aşağıdaki gibi gösterilebilir.

Ayrıştırma yapılmadan;

```
20130202 00:45:05
```

ISO 8601 standardına göre ise şu şekilde gösterilir.

```
2013-02-02 00:45:05
```

Sadece tarih:

```
20130202
2013-02-02
```

Sadece zaman:

```
00:45:05
```

SQL Server'da bu karmaşık görünen formatlama işlemlerini çözmek için, farklı dillere göre ayar değiştirme özelliği eklenmiştir.

Dışarıdan alınan bir tarih-saat değerini **CAST** ve **CONVERT** fonksiyonları kullanılarak, farklı birçok formata dönüştürmek mümkündür. Bu fonksiyonların kullanımını, ilerleyen bölümlerde tüm detaylarıyla işleyeceğiz.

Tarih formatında varsayılan kullanımın **yyyy-mm-dd** olduğunu belirtmiştik. Ancak bu format da isteğe göre değiştirilebilir.

```
--AY/GÜN/YIL
SET DATEFORMAT MDY                Sonuç : 2013-12-31 00:00:00.000
DECLARE @datevar datetime
SET @datevar = '12/31/13'
SELECT @datevar
GO
--YIL/GÜN/AY
SET DATEFORMAT YDM                Sonuç : 2013-12-31 00:00:00.000
DECLARE @datevar datetime
SET @datevar = '13/31/12'
SELECT @datevar
GO
```

```
--YIL/AY/GÜN
SET DATEFORMAT YMD                Sonuç : 2013-12-31 00:00:00.000
DECLARE @datevar datetime
SET @datevar = '13/12/31'
SELECT @datevar
GO
--GÜN/AY/YIL
SET DATEFORMAT DMY                Sonuç : 2013-12-31 00:00:00.000
DECLARE @datevar datetime
SET @datevar = '31/12/13'
SELECT @datevar
```

Yukarıdaki tüm sorgularda **@datevar** değişkeninin değerine bakarsanız, hepsinde tarih değerlerinin farklı konumlandırıldığını görebilirsiniz. Ancak tüm sorgu çıktılarında aynı tarih formatı ile sonuç üretildi.

SQL Server Tarih/Saat Fonksiyonları

Veritabanı işlemlerinde tarih ve saat gibi zaman bilgilerinin önemi yüksektir. Bu tür veriler farklı dil ve kültürlere göre şekillenen ve formatlanması gereken verilerdir. Bu nedenle, SQL Server gibi büyük veritabanlarında tarih/saat işlemlerini gerçekleştirecek çok sayıda veri tipi ve fonksiyon vardır.

Bu bölümde, SQL Server'da tarih/saat işlemlerini gerçekleştiren fonksiyonları inceleyeceğiz.

GETDATE

Sistemin anlık tarihini gösterir.

Söz Dizimi:

```
GETDATE()
```

Örnek:

```
SELECT GETDATE();
```

	(No column name)
1	2013-02-03 13:28:23.383

GETDATE fonksiyonu, genel olarak veritabanında en sık kullanılan tarih fonksiyonlarından biridir. Tablolarda **DEFAULT** olarak kullanılan **GETDATE** fonksiyonu, o anki sistem tarih-saat bilgisini otomatik olarak alarak veritabanına ekler.

CAST ve CONVERT İle Tarih Formatlama

Tarih formatını ayarlamak için kullanılır.

Söz Dizimi:

```
CONVERT(VARCHAR, Tarih, KOD)
```

Örnek:

```
SELECT CONVERT(VARCHAR(16), GETDATE(), 100)
```

Tarih dönüştürme işlemlerinde **CONVERT** sıklıkla kullanıldığı gibi **CAST** fonksiyonu da kullanılır.

```
SELECT GETDATE() AS DonusturulmemisTarih,
      CAST(GETDATE() AS NVARCHAR(30)) AS Cast_ile,
      CONVERT(NVARCHAR(30), GETDATE(), 126) AS Convert_ile;
```

	DonusturulmemisTarih	Cast_ile	Convert_ile
1	2013-02-03 13:28:50.923	Feb 3 2013 1:28PM	2013-02-03T13:28:50.923

String olarak alınan bir değer, **CAST** ve **CONVERT** fonksiyonlarının her ikisiyle de DATETIME veri tipine dönüştürülebilir.

```
SELECT '2006-04-25T15:50:59.997' AS DonusturulmemisTarih,
       CAST('2006-04-25T15:50:59.997' AS DATETIME) AS Cast_ile,
       CONVERT(DATETIME, '2006-04-25T15:50:59.997', 126) AS Convert_ile;
```

	DonusturulmemisTarih	Cast_ile	Convert_ile
1	2006-04-25T15:50:59.997	2006-04-25 15:50:59.997	2006-04-25 15:50:59.997

Tarih işlemlerinde birçok farklı kültür ve tarih kullanım formatı olması nedeniyle, dönüştürme ve tarih verilerini işleme sırasında birçok farklı sorunla karşılaşılabilmektedir.

Bu nedenle tarih biçimlerinin tamamını elde edebileceğiniz tüm yöntemleri göstermek adına, aşağıdaki tarih dönüştürme işlemlerini gerçekleştirelim.

CONVERT ile ilgili farklı formatlama kodları:

```
SELECT CONVERT(VARCHAR, GETDATE(), 0)    Sonuç : Feb  1 2013 10:06AM
SELECT CONVERT(VARCHAR, GETDATE(), 1)    Sonuç : 02/01/13
SELECT CONVERT(VARCHAR, GETDATE(), 2)    Sonuç : 13.02.01
SELECT CONVERT(VARCHAR, GETDATE(), 3)    Sonuç : 01/02/13
SELECT CONVERT(VARCHAR, GETDATE(), 4)    Sonuç : 01.02.13
SELECT CONVERT(VARCHAR, GETDATE(), 5)    Sonuç : 01-02-13
SELECT CONVERT(VARCHAR, GETDATE(), 6)    Sonuç : 01 Feb 13
SELECT CONVERT(VARCHAR, GETDATE(), 7)    Sonuç : Feb 01, 13
SELECT CONVERT(VARCHAR, GETDATE(), 8)    Sonuç : 10:08:01
SELECT CONVERT(VARCHAR, GETDATE(), 9)    Sonuç : Feb  1 2013 10:08:10:327AM
SELECT CONVERT(VARCHAR, GETDATE(), 10)   Sonuç : 02-01-13
SELECT CONVERT(VARCHAR, GETDATE(), 11)   Sonuç : 13/02/01
SELECT CONVERT(VARCHAR, GETDATE(), 12)   Sonuç : 130201
SELECT CONVERT(VARCHAR, GETDATE(), 13)   Sonuç : 01 Feb 2013 10:08:45:857
SELECT CONVERT(VARCHAR, GETDATE(), 14)   Sonuç : 10:08:54:127
SELECT CONVERT(VARCHAR, GETDATE(), 20)   Sonuç : 2013-02-01 10:09:07
SELECT CONVERT(VARCHAR, GETDATE(), 21)   Sonuç : 2013-02-01 10:09:23.973
SELECT CONVERT(VARCHAR, GETDATE(), 22)   Sonuç : 02/01/13 10:09:32 AM
SELECT CONVERT(VARCHAR, GETDATE(), 23)   Sonuç : 2013-02-01
SELECT CONVERT(VARCHAR, GETDATE(), 24)   Sonuç : 10:09:49
SELECT CONVERT(VARCHAR, GETDATE(), 25)   Sonuç : 2013-02-01 10:09:57.257
SELECT CONVERT(VARCHAR, GETDATE(), 100)  Sonuç : Feb  1 2013 10:10AM
SELECT CONVERT(VARCHAR, GETDATE(), 101)  Sonuç : 02/01/2013
SELECT CONVERT(VARCHAR, GETDATE(), 102)  Sonuç : 2013.02.01
SELECT CONVERT(VARCHAR, GETDATE(), 103)  Sonuç : 01/02/2013
SELECT CONVERT(VARCHAR, GETDATE(), 104)  Sonuç : 01.02.2013
SELECT CONVERT(VARCHAR, GETDATE(), 105)  Sonuç : 01-02-2013
SELECT CONVERT(VARCHAR, GETDATE(), 106)  Sonuç : 01 Feb 2013
SELECT CONVERT(VARCHAR, GETDATE(), 107)  Sonuç : Feb 01, 2013
SELECT CONVERT(VARCHAR, GETDATE(), 108)  Sonuç : 10:12:24
SELECT CONVERT(VARCHAR, GETDATE(), 109)  Sonuç : Feb 1 2013 10:12:31:237AM
SELECT CONVERT(VARCHAR, GETDATE(), 110)  Sonuç : 02-01-2013
SELECT CONVERT(VARCHAR, GETDATE(), 111)  Sonuç : 2013/02/01
SELECT CONVERT(VARCHAR, GETDATE(), 112)  Sonuç : 20130201
SELECT CONVERT(VARCHAR, GETDATE(), 113)  Sonuç : 01 Feb 2013 10:13:01:103
```

```
SELECT CONVERT(VARCHAR, GETDATE(), 114)  Sonuç : 10:13:14:437
SELECT CONVERT(VARCHAR, GETDATE(), 120)  Sonuç : 2013-02-01 10:13:21
SELECT CONVERT(VARCHAR, GETDATE(), 121)  Sonuç : 2013-02-01 10:13:27.237
SELECT CONVERT(VARCHAR, GETDATE(), 126)  Sonuç : 2013-02-01T10:13:34.317
SELECT CONVERT(VARCHAR, GETDATE(), 127)  Sonuç : 2013-02-01T10:13:41.897
```

Farklı ihtiyaçlara göre şekillenecek tarih formatı gereksinimi, yukarıdaki formatlar ile karşılanabilir.

FORMAT

Verilen tarihi istenen formata dönüştürerek string olarak verir.

Söz Dizimi:

```
FORMAT(tarih_zaman, format)
```

Örnek:

```
SELECT FORMAT(GETDATE(), 'yyyy.MM.d HH:MM:ss');
```

FORMAT fonksiyonu, genel olarak tarih bilgilerini farklı kültür formatlarında görüntülemek için kullanılır.

	(No column name)
1	2013.02.3 13:02:27

Belirli bir tarih bilgisini farklı kültür bilgilerine göre görüntüleyelim.

```
DECLARE @tarih DATETIME = GETDATE()
SELECT FORMAT ( @tarih, 'd', 'tr-TR' ) AS 'Türkçe'
      ,FORMAT ( @tarih, 'd', 'en-US' ) AS 'Amerikan İngilizcesi'
      ,FORMAT ( @tarih, 'd', 'en-gb' ) AS 'İngiltere İngilizcesi'
      ,FORMAT ( @tarih, 'd', 'de-de' ) AS 'Almanca'
      ,FORMAT ( @tarih, 'd', 'zh-cn' ) AS 'Çince';
```

	Türkçe	Amerikan İngilizcesi	İngiltere İngilizcesi	Almanca	Çince
1	03.02.2013	2/3/2013	03/02/2013	03.02.2013	2013/2/3

Küçük harfli `d` parametresini büyük harf `D` ile değiştirdiğimizde ise; ay ve gün bilgilerini ilgili dillere çevirerek açıkça görüntüleyecektir.

```
DECLARE @tarih DATETIME = GETDATE()
SELECT FORMAT ( @tarih, 'D', 'tr-TR' ) AS 'Türkçe'
      ,FORMAT ( @tarih, 'D', 'en-US' ) AS 'Amerikan İngilizcesi'
      ,FORMAT ( @tarih, 'D', 'en-gb' ) AS 'İngiltere İngilizcesi'
      ,FORMAT ( @tarih, 'D', 'de-de' ) AS 'Almanca'
      ,FORMAT ( @tarih, 'D', 'zh-cn' ) AS 'Çince';
```

	Türkçe	Amerikan İngilizcesi	İngiltere İngilizcesi	Almanca	Çince
1	03 Şubat 2013 Pazar	Sunday, February 03, 2013	03 February 2013	Sonntag, 3. Februar 2013	2013年2月3日

Farklı tarih işlemleri için **FORMAT** fonksiyonu aşağıdaki formatlarda kullanılabilir.

```
SELECT FORMAT ( GETDATE(), 'd', 'tr-TR' ) AS Turkiye;
SELECT FORMAT ( GETDATE(), 'dd', 'tr-TR' ) AS Turkiye;
SELECT FORMAT ( GETDATE(), 'ddd', 'tr-TR' ) AS Turkiye;
SELECT FORMAT ( GETDATE(), 'dddd', 'tr-TR' ) AS Turkiye;
```

	Turkiye
1	03.02.2013

	Turkiye
1	03

	Turkiye
1	Paz

	Turkiye
1	Pazar

```
SELECT FORMAT ( GETDATE(), 'm', 'tr-TR' ) AS Turkiye;
SELECT FORMAT ( GETDATE(), 'mm', 'tr-TR' ) AS Turkiye;
SELECT FORMAT ( GETDATE(), 'mmm', 'tr-TR' ) AS Turkiye;
SELECT FORMAT ( GETDATE(), 'mmmm', 'tr-TR' ) AS Turkiye;
```

	Turkiye
1	03 Şubat

	Turkiye
1	33

	Turkiye
1	33

	Turkiye
1	33

```
SELECT FORMAT ( GETDATE(), 'y', 'tr-TR' ) AS Turkiye;
SELECT FORMAT ( GETDATE(), 'yy', 'tr-TR' ) AS Turkiye;
SELECT FORMAT ( GETDATE(), 'yyy', 'tr-TR' ) AS Turkiye;
```

	Turkiye
1	Şubat 2013

	Turkiye
1	13

	Turkiye
1	2013

FORMAT Fonksiyonunun Farklı Ülke Para Birimleri İle Kullanımı

SQL Server'da maaş, ücret, ödeme gibi işlemler için saklanan para birimlerini farklı ülkelerin farklı para birimlerine göre dönüştürme ihtiyacı duyulur. Bu tür durumlarda SQL Server varsayılan destek sağlar.

```
DECLARE @para INT = 500;
SELECT FORMAT ( @para, 'c', 'tr-TR' ) AS Turkiye;
SELECT FORMAT ( @para, 'c', 'en-US' ) AS ABD;
SELECT FORMAT ( @para, 'c', 'fr-FR' ) AS Fransa;
SELECT FORMAT ( @para, 'c', 'de-DE' ) AS Almanca;
SELECT FORMAT ( @para, 'c', 'zh-cn' ) AS Cin;
```

	Turkiye
1	500,00 ₺

	ABD
1	$500.00

	Fransa
1	500,00 €

	Almanca
1	500,00 €

	Cin
1	¥500.00

Para değerinin sonuna 0 (sıfır) konulabilir. Bu durumda, sorgulamanın ikinci parametresi olan 'c' parametresi şu şekilde kullanılacaktır.

```
DECLARE @para INT = 50
SELECT FORMAT(@para,'c') AS Para;
SELECT FORMAT(@para,'c1') AS Para;
SELECT FORMAT(@para,'c2') AS Para;
SELECT FORMAT(@para,'c3') AS Para;
```

	Para
1	$50.00

	Para
1	$50.0

	Para
1	$50.00

	Para
1	$50.000

Bilindiği gibi Türk Lirası için belirlenen para simgesi, 1 Mart 2012 tarihinde resmi olarak kullanıma sunulduğu duyuruldu. Bu resmi duyurudan sonra global ve yerel yazılımlar, Türk Lirası simgesini kullanmak için entegrasyon yazılımları hazırladı. Microsoft'ta TL simgesini bir güncelleştirme ile duyurarak SQL Server ve Windows'ta kullanılabilir hale getirdi.

Windows'ta, dolayısıyla SQL Server'da, TL simgesini kullanabilmek için (**AltGr+T**) tuş kombinasyonlarını kullanmanız gerekir.

Aşağıdaki Windows güncelleştirme bağlantısından ilgili güncelleştirmeyi bilgisayarınıza kurarak, TL simgesinin Windows ve SQL Server tarafından desteklenmesini sağlamanız gerekir.

TL Simgesi: ₺

İndirme Bağlantısı: *http://support.microsoft.com/kb/2739286*

FORMAT fonksiyonu farklı bazı işlemler için de kullanılabilir. Bu işlemleri bir kaç örnek vererek açıklayalım.

Yüzde hesaplamaları, bilimsel ve heksadesimal (*hexa*) gibi işlemler için kullanılabilir.

```
DECLARE @var INT = 50
SELECT FORMAT(@var,'p') AS Yüzde;         --(P = Percentage)
SELECT FORMAT(@var,'e') AS Bilimsel;      --(E = Scientific)
SELECT FORMAT(@var,'x') AS Hexa;
SELECT FORMAT(@var,'x4') AS Hexa1;
```

O anki sistem tarihini, kendi belirlediğiniz tarih formatı ve kültür formatlarında görüntüleyebilir.

```
SELECT FORMAT (GETDATE(), N'dddd MMMM dd, yyyy', 'tr-TR') AS Turkce;
SELECT FORMAT (GETDATE(), N'dddd MMMM dd, yyyy', 'en-US') AS Ingilizce;
SELECT FORMAT (GETDATE(), N'dddd MMMM dd, yyyy', 'hi') AS HindistanDili;
SELECT FORMAT (GETDATE(), N'dddd MMMM dd, yyyy', 'gu') AS GujaratDili;
SELECT FORMAT (GETDATE(), N'dddd MMMM dd, yyyy', 'zh-cn') AS Cince;
```

	Yüzde
1	5,000.00 %

	Bilimsel
1	5.000000e+001

	Hexa
1	32

	Hexa1
1	0032

FORMAT fonksiyonunun, veritabanındaki veriler üzerinde kullanımı da basittir.

```
SELECT ModifiedDate, ListPrice,
  FORMAT(ModifiedDate, N'dddd MMMM dd, yyyy','tr') TR_DegistirmeTarih,
  FORMAT(ListPrice, 'c','tr') TR_UrunFiyat
FROM Production.Product;
```

	ModifiedDate	ListPrice	TR_DegistirmeTarih	TR_UrunFiyat
1	2008-03-11 10:01:36.827	0,00	Salı Mart 11, 2008	0,00 ₺
2	2008-03-11 10:01:36.827	0,00	Salı Mart 11, 2008	0,00 ₺
3	2008-03-11 10:01:36.827	0,00	Salı Mart 11, 2008	0,00 ₺
4	2008-03-11 10:01:36.827	0,00	Salı Mart 11, 2008	0,00 ₺
5	2008-03-11 10:01:36.827	0,00	Salı Mart 11, 2008	0,00 ₺
6	2008-03-11 10:01:36.827	0,00	Salı Mart 11, 2008	0,00 ₺
7	2008-03-11 10:01:36.827	0,00	Salı Mart 11, 2008	0,00 ₺

DATEPART

Tarih ile ilgili bazı rakamsal bilgiler alınmasını sağlar. Verilen tarihin gün, ay, yıl gibi parçalar halinde geri almak için kullanılır.

Söz Dizimi:

```
DATEPART (parca_datepart, tarih)
```

Örnek:

```
SELECT DATEPART(WEEKDAY, GETDATE());
```

	(No column name)
1	1

DATEPART fonksiyonu ile kullanılan parçalar aşağıdaki gibidir:

YY, YYYY ya da YEAR

Verilen tarih içerisinden yıl bilgisini almayı sağlar.

```
SELECT DATEPART(YY, GETDATE());                    Sonuç : 2013
SELECT DATEPART(YYYY, GETDATE());                  Sonuç : 2013
SELECT DATEPART(YEAR, GETDATE());                  Sonuç : 2013
```

QQ, Q ya da Quarter

Verilen tarihin çeyreğini almak için kullanılır. 12 ayın 4'e bölümü sonucunda ortaya çıkan her 3 ay bir çeyrektir. İlk 3 ay ilk çeyrek, son 3 ay son çeyrektir.

```
SELECT DATEPART(QQ, GETDATE());                    Sonuç : 1
SELECT DATEPART(Q, GETDATE());                     Sonuç : 1
SELECT DATEPART(QUARTER, GETDATE());               Sonuç : 1
```

Sorgu çalıştırıldığında ilk çeyrekte olduğumuz için bu sonuç dönmektedir.

MM, M ya da MONTH

Verilen tarihin ay bilgisini almak için kullanılır.

```
SELECT DATEPART(MM, GETDATE());
SELECT DATEPART(M, GETDATE());
SELECT DATEPART(MONTH, GETDATE());
```

DY, Y ya da DAYOFYEAR

Yılın kaçıncı günü olduğunu almak için kullanılır.

```
SELECT DATEPART(DY, GETDATE());
SELECT DATEPART(Y, GETDATE());
SELECT DATEPART(DAYOFYEAR, GETDATE());
```

DD, D ya da DAY

Verilen tarihin gün bilgisini almak için kullanılır.

```
SELECT DATEPART(DD, GETDATE());
SELECT DATEPART(D, GETDATE());
SELECT DATEPART(DAY, GETDATE());
```

WK ya da WW

Verilen tarihin hafta bilgisini almak için kullanılır.

```
SELECT DATEPART(WK, GETDATE());
SELECT DATEPART(WW, GETDATE());
SELECT DATEPART(WEEK, GETDATE());
```

DW ya da WEEKDAY

Haftanın kaçıncı günü olduğunu almak için kullanılır.

```
SELECT DATEPART(DW, GETDATE());
SELECT DATEPART(WEEKDAY, GETDATE());
```

HH ya da HOUR

Verilen tarih-zamanın saat bilgisini almak için kullanılır.

```
SELECT DATEPART(HH, GETDATE());
SELECT DATEPART(HOUR, GETDATE());
```

MI, N ya da MINUTE

Verilen tarih-zamanın dakika bilgisini almak için kullanılır.

```
SELECT DATEPART(MI, GETDATE());
SELECT DATEPART(N, GETDATE());
SELECT DATEPART(MINUTE, GETDATE());
```

SS, S YA DA SECOND

Verilen tarih-zamanın saniye bilgisini almak için kullanılır.

```
SELECT DATEPART(SS, GETDATE());
SELECT DATEPART(S, GETDATE());
SELECT DATEPART(SECOND, GETDATE());
```

MS, MCS MILLISECOND

Verilen tarih-zamanın milisaniye bilgisini almak için kullanılır.

```
SELECT DATEPART(MS, GETDATE());
SELECT DATEPART(MILLISECOND, GETDATE());
```

ISDATE

Tarih formatının geçerliliğini denetler. Bu fonksiyonun kullanımında kültür kodu önemlidir. Tarih formatlama işlemlerinde bir kültür kodu (tr-TR, en-EN vb.) ile başarılı şekilde çalışan tarih formatı bir başkasında çalışmayabilir. Veri işlemlerinde hataya sebep olmamak için bazen tarih formatlarını test etmek ve geçerliliğini onayladıktan sonra işleme almak gerekebilir. Bu tür durumlarda **ISDATE** fonksiyonu kullanılabilir.

Tarih formatı geçerliyse 1, geçerli değilse 0 değerini döndürür.

Söz Dizimi:

```
ISDATE(tarih)
```

Varsayılan olarak SQL Server dil ayarı **us_english**, yani Amerikan İngilizcesi ile kullanılır. Bu ayarda iken aşağıdaki sorguları çalıştıralım.

```
SELECT ISDATE('02/15/2013');      Sonuç : 1
SELECT ISDATE('15/02/2013');      Sonuç : 0
```

İlk sorgudaki format, 1 değerini döndürdü. Yani, ilk tarih formatımız geçerlidir. Ancak ikinci sorgudaki format, 0 değerini döndürerek geçerli olmadığını bildirdi.

Şimdi ise dil kodunu değiştirelim.

```
SET LANGUAGE Turkish
```

```
Dil ayarı 'Türkçe' olarak değiştirildi.
```

Yukarıda geçerlilik testi yaptığımız iki tarihi de aynı şekilde tekrar sorgulayalım.

```
SELECT ISDATE('02/15/2013');      Sonuç : 0
SELECT ISDATE('15/02/2013');      Sonuç : 1
```

Sorgular aynı, ancak sonuçlar farklı. Bunun nedeni, farklı dillerde farklı formatların kullanılıyor olmasıdır.

Yazılımsal olarak geçerlilik testi şu şekilde yapılabilir.

```
IF ISDATE('15/02/2013') = 1
    PRINT 'GEÇERLİ'
ELSE
    PRINT 'GEÇERSİZ'
```

```
GEÇERLI
```

Şu anki geçerli dil Türkçe olduğu için bu format geçerli olarak sonuç döndürdü.

Dil ayarını değiştirelim.

```
SET LANGUAGE us_english
```

```
Changed language setting to us_english.
```

Aynı sorguyu tekrar çalıştıralım.

```
IF ISDATE('15/02/2013') = 1
    PRINT 'GEÇERLİ'
ELSE
    PRINT 'GEÇERSİZ'
```

```
GEÇERSIZ
```

Benzer şekilde farklı diller ile şu şekilde farklı sonuçlar üretecek bir örnek oluşturalım.

```
SET LANGUAGE Turkish;
SELECT ISDATE('15/02/2013'); -- Sonuç : 1
SET LANGUAGE English;
SELECT ISDATE('15/02/2013'); -- Sonuç : 0
SET LANGUAGE Hungarian;
SELECT ISDATE('15/2013/02'); -- Sonuç : 0
SET LANGUAGE Swedish;
SELECT ISDATE('2013/15/02'); -- Sonuç : 0
SET LANGUAGE Italian;
SELECT ISDATE('15/02/2013'); -- Sonuç : 1
```

Dil ayarını değiştirdikten sonra haliyle sonuçta değişti. Bu örnek, tarih işlemlerinin aslında göründüğü kadar kolay olmadığını ve gelişmiş veritabanlarında ne kadar önemli ve kritik olabileceğini gösterir.

Çok dilli veritabanı mimarilerinde bir çok para birimi ve dil ile uğraşmak cidden çok can sıkıcı hal alabilir. Bu nedenle tarih işlemlerine hakim olmanız işleri kolaylaştıracaktır.

DATEADD

Verilen tarihin ay, gün ya da yıl verisini artırmak ya da azaltmak için kullanılır.

Söz Dizimi:

```
DATEADD ( datepart , sayi, tarih )
```

- **`Datepart`**: Değişiklik yapılacak kısım.
- **`Number`**: Eklenecek ya da çıkarılacak sayı.
- **`Date`**: Değişiklik yapılacak tarih.

Örnek:

01.01.2013 tarihine 2 yıl ekleyelim.

```
SELECT DATEADD(YY, 2, '01.01.2013');
```

Belirtilen tarihin 2 ay ekleyelim.

```
SELECT DATEADD(MM, 2, '01.01.2013');
```

Belirtilen tarihin çeyreğini değiştirelim.

```
SELECT DATEADD(QQ, 3, '01.01.2013');
```

Belirtilen tarihin ayını değiştirelim. 3 ay öncesine gidelim.

```
SELECT DATEADD(MM, -3, '01.01.2013');
```

DATEPART fonksiyonunda kullanılan tüm parçalar **DATEADD** fonksiyonunda da kullanılarak azaltma ya da artırma işlemleri yapılabilir.

Bu parçaları özetle hatırlatmak gerekirse;

- **dy / y**: Yılın gününü değiştirmek için kullanılır.
- **dd / d**: Günü değiştirmek için kullanılır.
- **wk / ww**: Haftayı değiştirmek için kullanılır.
- **dw**: Haftanın gününü değiştirmek için kullanılır.
- **hh**: Saati değiştirmek için kullanılır.
- **mi / n**: Dakikayı değiştirmek için kullanılır.
- **ss / s**: Saniyeyi değiştirmek için kullanılır.
- **ms**: Milisaniyeyi değiştirmek için kullanılır.

DATEDIFF

Verilen iki tarih arasındaki **Datepart** parametresi farkını verir.

Söz Dizimi:

```
DATEDIFF(datepart, baslangic_tarih, bitis_tarih)
```

Örnek:

İki tarih arasındaki yıl farkını bulalım.

```
SELECT DATEDIFF(YY,'01.01.2012','01.01.2013');
```

İki tarih arasındaki ay farkını bulalım.

```
SELECT DATEDIFF(MONTH,'01.01.2012','01.01.2013');
```

İki tarih arasındaki hafta farkını bulalım.

```
SELECT DATEDIFF(WK,'01.01.2012','01.01.2013');
```

İki tarih arasındaki gün farkını bulalım.

```
SELECT DATEDIFF(DD,'01.01.2012','01.01.2013');
```

Bu ve benzer işlemler için diğer fonksiyonlarda kullanılan tüm tarih datepart'ları (*parçaları*) kullanılabilir.

Örneğin; bu tarihler arası, yani bir yılın kaç saniye olduğunu hesaplayalım.

```
SELECT DATEDIFF(ss,'01.01.2012','01.01.2013');
```

DATENAME

Datepart olarak verilen parametrenin adını döndürür.

Söz Dizimi:

```
DATENAME(datepart, tarih)
```

Örnek:

Bulunduğumuz tarihin yıl bilgisini alalım.

```
SELECT DATENAME(YY, GETDATE());
```

Bulunduğumuz tarihin ay bilgisini alalım.

```
SELECT DATENAME(YY, GETDATE());
```

Benzer şekilde diğer tüm **Datepart** parametreleri kullanılarak ilgili bilgiler alınabilir.

Bulunduğumuz zamanın milisaniye bilgisini alalım.

```
SELECT DATENAME(MS, GETDATE());
```

DAY

Verilen bir tarihin sadece gün değerini alır.

Söz Dizimi:

```
DAY(GETDATE())
```

Örnek:

```
SELECT DAY(GETDATE());
```

MONTH

Verilen bir tarihin sadece ay değerini alır.

Söz Dizimi:

```
MONTH(GETDATE())
```

Örnek:

```
SELECT MONTH(GETDATE());
```

YEAR

Verilen bir tarihin sadece yıl değerini alır.

Söz Dizimi:

```
YEAR(GETDATE())
```

Örnek:

```
SELECT YEAR(GETDATE());
```

DATEFROMPARTS

Tarihin el ile verilmesini sağlar ve tarih döndürür.

Söz Dizimi:

```
DATEFROMPARTS(yıl, ay, gün)
```

Örnek:

```
SELECT DATEFROMPARTS(2013, 1, 1);
```

DATETIMEFROMPARTS

Tarih zamanın el ile verilmesini sağlar.

Söz Dizimi:

```
DATETIMEFROMPARTS(yıl, ay, gün, saat, dakika, saniye, salise, milisaniye)
```

Örnek:

```
SELECT DATETIMEFROMPARTS(2013, 02, 01, 17, 37, 12, 997);
```

Bu tür el ile işlem yapan fonksiyonlar genel olarak uygulamalardan ziyade, veri ve sorgu testlerinde kullanılabilir.

SMALLDATETIMEFROMPARTS

`SmallDateTime`'ın el ile verilmesini sağlar ve tarih döndürür.

Söz Dizimi:

```
SMALLDATETIMEFROMPARTS(yıl, ay, gün, saat, dakika)
```

Örnek:

```
SELECT SMALLDATETIMEFROMPARTS(2013, 2, 1, 17, 45)
```

TIMEFROMPARTS

El ile verilen bir zamanın, zaman tipinde bir değer olarak döndürülmesini sağlar.

Söz Dizimi:

```
TIMEFROMPARTS(saat, dakika, saniye, milisaniye, ondalık)
```

Örnek:

```
SELECT TIMEFROMPARTS (17, 25, 55, 5, 1);
```

TIMEFROMPARTS'ın aldığı son parametre, bir önceki parametrenin ondalık değerini belirtmek için kullanılır. Dördüncü parametre tek basamaklı ise beşinci parametre 1 değerini alır. Dördüncü parametre iki basamaklı ise beşinci parametre iki değerini alır. Son parametre on basamağa kadar değer alabilir.

```
SELECT TIMEFROMPARTS (17, 25, 55, 5, 1);
SELECT TIMEFROMPARTS (17, 25, 55, 50, 2);
SELECT TIMEFROMPARTS (17, 25, 55, 500, 3);
```

Yukarıdaki kullanım doğru olmakla birlikte, aşağıdaki kullanım hata üretecektir.

```
SELECT TIMEFROMPARTS (17, 25, 55, 5000, 3);
```

TIMEFROMPARTS fonksiyonu, milisaniye değerinin maksimum 7 digit olması nedeniyle, 7 ondalık basamağına kadar değer alabilir.

```
SELECT TIMEFROMPARTS (17, 25, 55, 5000000, 7);
```

Son parametre `0` (*sıfır*) ya da **NULL** değer de içeremez.

EOMONTH

Verilen bir tarihin ayının kaç çektiğini verir

Söz Dizimi:

```
EOMONT(tarih)
EOMONT(tarih, kod)
```

Örnek:

```
SELECT EOMONTH(GETDATE());
```

Örnek olarak belirtilen **EOMONTH(GETDATE())** kullanımı ile sistem tarihindeki ay bilgisinin son gününün tam tarihini verecektir.

```
SELECT EOMONTH(GETDATE());
```

EOMONTH fonksiyonu ile şu anki tarihin kaç gün çektiği öğrenilebileceği gibi, önceki, sonraki ya da belirtilen bir başka ayın da kaç gün çektiği öğrenilebilir.

```
DECLARE @date DATETIME = GETDATE();
SELECT EOMONTH (@date, -1) AS 'Önceki Ay';
SELECT EOMONTH (@date) AS 'Şimdiki Ay';
SELECT EOMONTH (@date, 1) AS 'Sonraki Ay';
```

	Önceki Ay
1	2013-01-31

	Şimdiki Ay
1	2013-02-28

	Sonraki Ay
1	2013-03-31

Yukarıdaki örnek ile ilk olarak bir önceki ayın son gün bilgisi, sonra şuan bulunulan ay ve son olarak bir sonraki ayın son gün bilgisi listelenir.

SYSDATETIME

DateTime2 veri tipinde, sistemde anlık tarih ve zaman bilgisini döndürür. **GETDATE** fonksiyonu 3 digit milisaniye değeri döndürürken, **SYSDATETIME** fonksiyonu 7 digit'in tamamını döndürür.

Söz Dizimi:

```
SELECT SYSDATETIME
```

Örnek:

```
SELECT GETDATE() [GetDate], SYSDATETIME() [SysDateTime]
```

	GetDate	SysDateTime
1	2013-02-03 13:45:33.813	2013-02-03 13:45:33.8157828

SYSUTCDATETIME

UTC zamanını **DateTime2** veri tipinde döndürür.

Söz Dizimi:

```
SYSUTCDATETIME()
```

Örnek:

```
SELECT GETDATE(), SYSUTCDATETIME();
```

	(No column name)	(No column name)
1	2013-02-03 13:46:13.217	2013-02-03 11:46:13.2188881

SYSDATETIMEOFFSET

UTC'ye göre offset bilgisi ile yerel SQL Server saatini döndürür.

Söz Dizimi:

```
SYSDATETIMEOFFSET()
```

Örnek:

```
SELECT SYSDATETIMEOFFSET() OffSet;
SELECT TODATETIMEOFFSET(SYSDATETIMEOFFSET(), '-04:00') 'OffSet -4';
SELECT TODATETIMEOFFSET(SYSDATETIMEOFFSET(), '-02:00') 'OffSet -2';
SELECT TODATETIMEOFFSET(SYSDATETIMEOFFSET(), '+00:00') 'OffSet +0';
SELECT TODATETIMEOFFSET(SYSDATETIMEOFFSET(), '+02:00') 'OffSet +2';
SELECT TODATETIMEOFFSET(SYSDATETIMEOFFSET(), '+04:00') 'OffSet +4';
```

	OffSet
1	2013-02-03 13:46:58.3020805 +02:00

	OffSet -4
1	2013-02-03 13:46:58.3020805 -04:00

	OffSet -2
1	2013-02-03 13:46:58.3020805 -02:00

	OffSet +0
1	2013-02-03 13:46:58.3020805 +00:00

	OffSet +2
1	2013-02-03 13:46:58.3020805 +02:00

	OffSet +4
1	2013-02-03 13:46:58.3020805 +04:00

SWITCHOFFSET

Herhangi bir tarihi, başka bölge tarihine dönüştürmek için kullanılır. Girdi tarihte OffSet bilgisinin bulunması gerekir.

Söz Dizimi:

```
SWITCHOFFSET (DATETIMEOFFSET, zaman_dilimi)
```

Örnek:

`OffSetTest` isimli tablo oluşturalım.

```
CREATE TABLE dbo.OffSetTest(
    ColDatetimeoffset datetimeoffset
);
```

Bir tarih verisi ekleyelim.

```
INSERT INTO dbo.OffSetTest VALUES ('2013-02-01 8:25:50.71345 -5:00');
```

`SWITCHOFFSET` fonksiyonunu test edelim.

```
SELECT SWITCHOFFSET (ColDatetimeoffset, '-08:00') FROM dbo.OffSetTest;

SELECT ColDatetimeoffset FROM dbo.OffSetTest;
```

	(No column name)
1	2013-02-01 05:25:50.7134500 -08:00

	ColDatetimeoffset
1	2013-02-01 08:25:50.7134500 -05:00

TODATETIMEOFFSET

Herhangi bir tarihi koruyarak sadece UTC OffSet değerini ekler ya da varsa değiştirir.

Söz Dizimi:

```
TODATETIMEOFFSET()
```

Örnek:

```
SELECT TODATETIMEOFFSET(SYSDATETIMEOFFSET(), '-05:00') AS TODATETIME;
```

	TODATETIME
1	2013-02-03 13:50:25.2232926 -05:00

GETUTCDATE

Anlık olarak Greenwich, yani GMT tarih-saat bilgisini döndürmeye yarar.

Söz Dizimi:

```
GETUTCDATE()
```

Örnek:

```
SELECT GETUTCDATE()
```

	(No column name)
1	2013-02-03 11:50:54.213