Tarih ve Saat Fonksiyonları

SQL Server'da takvim bilgilerini depolamak için tarih ve saat veri türlerini kullanabiliriz. Bu veri türleri arasında time, date, smalldatetime, datetime, datetime2 and datetimeoffset bulunur. Her veri türünün aşağıdaki gibi belirli bir formatı vardır.

tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

En çok kullanılan tarih işlevleri:

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Tarih Fonksiyonları**

GETDATE()

Geçerli SQL örneğini çalıştıran bilgisayarın geçerli tarih ve saatini belirlemek için GETDATE() işlevini kullanabilirsiniz. Bu işlev saat dilimi farkını içermez, yalnızca tarih saat biçimini döndürür.

Bir GETDATE() fonksiyon örneği görelim.





DATENAME(datepart, date)

DATENAME() işlevi, tarihin belirli bir bölümünün adını veya değerini nvarchar biçiminde döndürür.





**💡İpuçları:**

SQL Server'da kullanabileceğiniz kırk tane tarih bölümü ipucu vardır. Örneğin: DAY, HOUR, MINUTE, WEEKDAY, YEAR, DAYOFYEAR, MONTH, vb.

DATEPART(datepart, date)

DATEPART() işlevi, tarihin belirli bir bölümünün değerini tamsayı biçiminde döndürür.





DAY(date)

DAY() işlevi, tarihin gününü tamsayı biçiminde döndürür.





MONTH(date)

MONTH() işlevi, tarihin ayını tamsayı biçiminde döndürür.





YEAR(date)

YEAR() işlevi, tarihin yılını tamsayı biçiminde döndürür.





DATEDIFF(datepart, startdate, enddate)

DATEDIFF() işlevi, iki tarih arasındaki farkı tamsayı biçiminde döndürür.

Sözdiziminde tarih bölümü, farkı hesaplamak için tarihin hangi bölümünü kullanmak istediğinizi belirten parametredir. Tarih bölümü yıl, ay, hafta, gün, saat, dakika, saniye veya milisaniye olabilir. Ardından, başlangıç tarihi parametresinde başlangıç tarihini, bitiş tarihi parametresinde ise farkını bulmak istediğiniz bitiş tarihini belirtirsiniz.





DATEADD(datepart, number, date)

DATEADD() işlevi, belirli bir tarihin parçasına bir aralık eklemenizi sağlar.

DATEADD() fonksiyon örneğini görelim.





EOMONTH(startdate [, month to add])

EOMONTH() işlevi, isteğe bağlı bir mahsup ile belirtilen bir tarihi içeren ayın son gününü döndürür.

Bir EOMONTH() fonksiyon örneği görelim.





ISDATE(expression)

ISDATE(), ifade geçerli bir tarih saat değeriyse 1 döndürür; aksi halde 0.

Sistem diliniz us\_english ise, tarih formatı varsayılan olarak "mdy" (ay, gün, yıl) şeklindedir. ISDATE() işlevi, ifadeyi bu biçime göre kontrol eder.

Gerekirse, tarih biçimini aşağıdaki gibi değiştirebilirsiniz:



Bir ISDATE() fonksiyon örneği görelim. (tarih biçimi = mdy)





**Dize İşlevleri**

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Dize işlevleri, dize değerlerini işlemek için kullanılır. Veri analizinde veri temizleme işlemleri için kullanışlıdırlar.

**LEN(), CHARINDEX(), PATINDEX() Functions**

LEN()

LEN() işlevi, bir dizenin karakter sayısını döndürür (metnin sonundaki boşluklar hariç). Sonucun dönüş türü bir tamsayıdır.





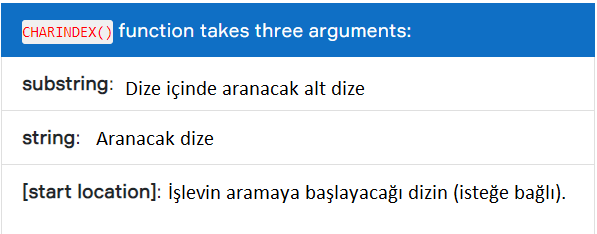
Dize NULL değeriyse, uzunluk işlevi NULL değerini döndürür. İşlev içinde belirtilen değer sayısal ise, LEN() işlevi, değerin dize temsilinin uzunluğunu döndürür. Bu, sayısal değerin bir dizgeye dönüştürüldüğü ve ardından karakter sayısının hesaplandığı anlamına gelir.





CHARINDEX(substring, string [, start location])

CHARINDEX() işlevi, argüman olarak bir dize ve bunun bir alt dizesini alır ve alt dizenin ilk karakteri olan alt dizenin konumunu gösteren bir tamsayı döndürür. CHARINDEX() işlevi, alt dizenin ilk oluşumunu bulur ve tamsayı türünde bir değer döndürür.



Alt dize bulunamazsa, CHARINDEX() işlevi 0 döndürür.

Örnekler:

➊ Aşağıdaki örnek, “Reinvent” alt dizesini bulmak için “Reinvent yourself” dizesini arar. “yourself”ilk karakterinin “y” olan dizin numarasını döndürür.





➋ CHARINDEX(), girdinin harmanlamasına dayalı olarak karşılaştırmalar gerçekleştirir. Veritabanı harmanlamanız büyük/küçük harf duyarlılığına ayarlanmışsa, aradığınız karakterin büyük veya küçük harf olması önemlidir. Aşağıdaki sorgu, harmanlama büyük/küçük harfe duyarlı olmayacak şekilde ayarlandığından, 'r' veya 'R' alt dizesinin ilk oluşumunun dizin numarasını döndürür.





➌ Aşağıdaki sorgu, 'self' alt dizesinin ilk oluşumunu bulur ve ilk karakterinin dizin numarasını döndürür. Gördüğünüz gibi iki 'self' var. CHARINDEX() işlevi, yalnızca 'yourself' içindeki 'self' olan ilk oluşumu bulur.





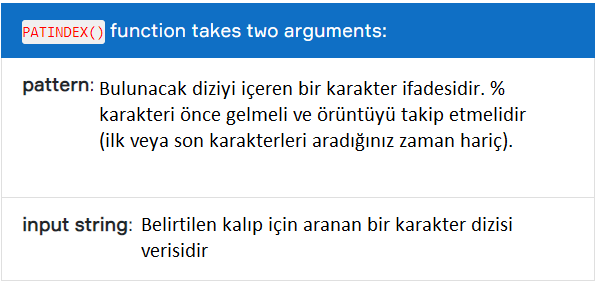
➍ Ancak aşağıdaki sorgu, isteğe bağlı parametre [başlangıç konumu] kullanarak ikinci 'self' bulur





PATINDEX(%pattern%, input string)

PATINDEX() işlevi, belirtilen bir ifadede bir kalıbın ilk oluşumunun başlangıç konumunu veya kalıp bulunamazsa tüm geçerli metin ve karakter veri türlerinde sıfırları döndürür.



💡İpuçları:

pattern veya ifade NULL ise, PATINDEX() NULL değerini döndürür.

PATINDEX() için başlangıç konumu 1'dir.

PATINDEX tıpkı LIKE gibi çalışır, böylece herhangi bir joker karakter kullanabilirsiniz. Deseni yüzdeler arasına almak zorunda değilsiniz. LIKE'dan farklı olarak, PATINDEX(), CHARINDEX()'in yaptığına benzer bir konum döndürür.





Desenin % karakter sonunu kullanmazsak:





**UPPER(), LOWER(), STRING\_SPLIT() Fonksiyonları**

UPPER() işlevi, tüm küçük harfli ASCII karakterlerinin büyük harf eşdeğerine dönüştürüldüğü bir dizenin bir kopyasını döndürür.

LOWER() işlevi, tüm büyük harfli ASCII karakterlerinin küçük harf eşdeğerine dönüştürüldüğü bir dizenin bir kopyasını döndürür.

UPPER() ve LOWER() işlevleri, boş bir değer iletirseniz boş değerler döndürür. UPPER() veya LOWER() öğesine sayısal değerler iletirseniz, her ikisi de tam sayısal değeri döndürür.



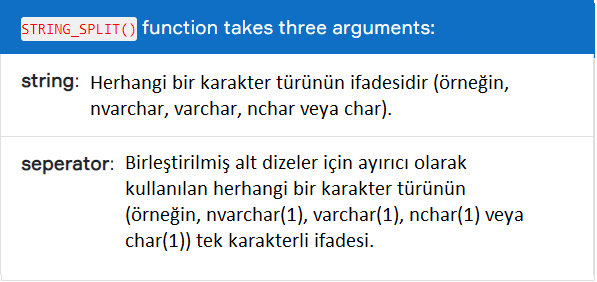






STRING\_SPLIT(string, separator)

STRING\_SPLIT() işlevi, bir dizeyi belirtilen bir ayırıcı karaktere dayalı olarak alt dize satırlarına bölen tablo değerli bir işlevdir.





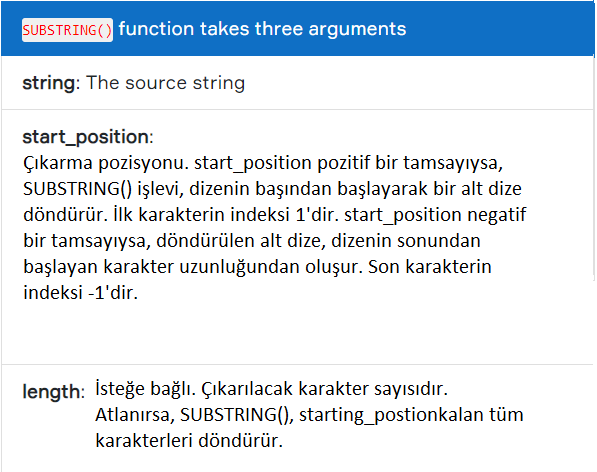
tablo içeren bir resim

Açıklama otomatik olarak oluşturuldu

**SUBSTRING(), LEFT(), RIGHT() Fonksiyonları**

SUBSTRING(string, start\_postion, [length])

SUBSTRING() işlevi, bir dizeden bir alt dize çıkarmanızı sağlar. Dönüş değeri metindir.



Herhangi bir bağımsız değişken NULL ise, SUBSTRING() işlevi NULL değerini döndürür.







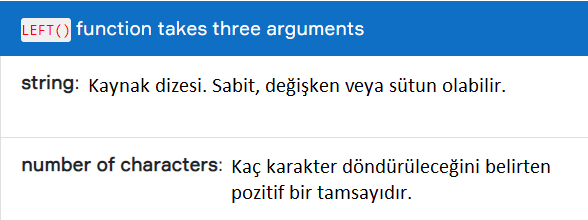






LEFT(string, number of characters)

Belirtilen sayıda karaktere sahip bir karakter dizesinin sol kısmını döndürür.







RIGHT(string, number of characters)

Belirtilen sayıda karaktere sahip bir karakter dizesinin sağ bölümünü döndürür.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

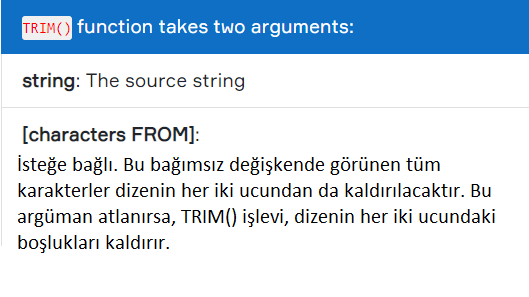




**TRIM(), LTRIM(), RTRIM() Fonksiyonları**

TRIM([characters FROM] string)

TRIM() işlevi, belirtilen karakterleri dizenin her iki ucundan kaldırır.



TRIM() işlevi, belirtilen baştaki ve sondaki karakterlerin kaldırıldığı yeni bir dize döndürür. Orijinal dizeyi değiştirmez.

➊ Aşağıdaki sorgu argüman olarak '' Reinvent Yourself ' dizesini alır ve baştaki ve sondaki boşlukları kaldırır. Karakter argümanı bu sorguda belirtilmemiş.





➋ Aşağıdaki ifade, dizenin başındaki ve sonundaki @ karakterlerini kaldırır.





➌ Aşağıdaki sorgu, verilen 'cadillac' dizesinden baştaki ve sondaki 'ca' karakterlerini çıkardıktan sonra dizeyi döndürür.





LTRIM(string)

Baştaki boşlukları kaldırdıktan sonra bir karakter ifadesi döndürür.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu





RTRIM(string)

Sondaki tüm boşlukları kestikten sonra bir karakter dizesi döndürür.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

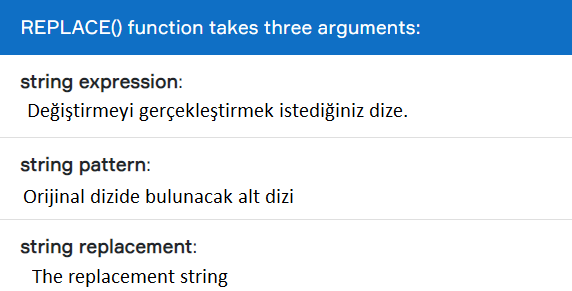




**REPLACE() ve STR() Fonksiyonları**

REPLACE(string expression, string pattern, string replacement)

REPLACE() işlevi, belirtilen bir dizenin tüm oluşumlarını başka bir dizeyle değiştirmenize olanak tanır.



REPLACE() işlevi, bir tablodaki karakter verilerini güncellemek için kullanışlıdır, örneğin ölü bağlantıları ve yazım hatalarını güncellemek.

➊ Aşağıdaki örnek, 'REIMVEMT' dizesindeki 'M' karakterini yeni 'N' karakteriyle değiştirir.





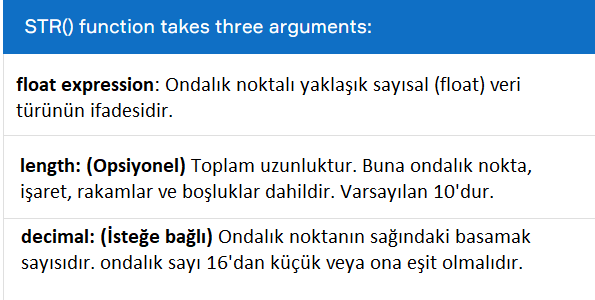
➋ Aşağıdaki sorgu, 'I do it my way' dizesindeki 'do' dizesini “did” ile değiştirmek için REPLACE işlevini kullanır





STR(float expression [, length [, decimal]])

Sayısal verilerden dönüştürülmüş karakter verilerini döndürür. Karakter verileri, belirli bir uzunluk ve ondalık kesinlik ile sağa dayalıdır.



Aşağıdaki örnek, beş basamak ve bir ondalık noktadan oluşan bir ifadeyi altı konumlu bir karakter dizesine dönüştürür. Sayının kesirli kısmı bir ondalık basamağa yuvarlanır.



İfade belirtilen uzunluğu aştığında, dize belirtilen uzunluk için \*\* döndürür.





Sayısal veriler STR içinde yuvalanmış olsa bile, sonuç belirtilen formatta karakter verileridir.









**Birleştirme için "+" Operatörü veya CONCAT() İşlevi**

SQL standardı, iki dizeyi tek bir dizede birleştirmek için CONCAT() işlevini sağlar. Ayrıca, bitiştirme operatörü (+) iki dizgiyi tek bir dizgede birleştirmek için kullanılır. (+) operatörünü birden çok kez kullanarak ikiden fazla dizgiyi birleştirmek de mümkündür.



➊ Aşağıdaki sorgu, 'Reinvent' ve ' yourself' dizelerini birleştirir (‘yourself' önünde bir boşluk karakteri vardır).





Aşağıdaki gibi CONCAT() işleviyle de yapabilirsiniz:





➋ Aşağıdaki sorgu





➌ Aşağıdaki sorgu, ad ve soyad sütunlarını boşluk karakteriyle birleştirir.





**Diğer Fonksiyonlar ve İfadeler**

metin içeren bir resim

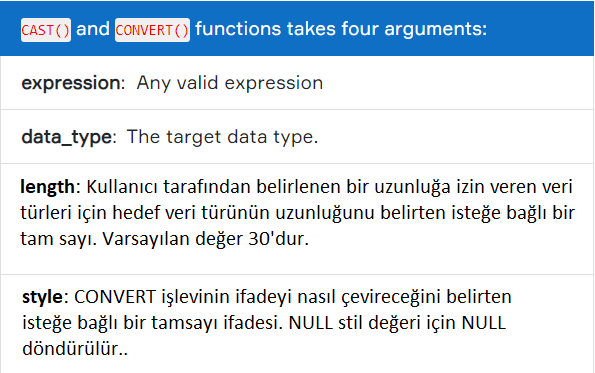
Açıklama otomatik olarak oluşturuldu

**CAST() ve CONVERT() İşlevleri**

Bu işlevler, bir veri türünün ifadesini diğerine dönüştürür.

CAST ( expression AS data\_type [ ( length ) ] )

CONVERT ( data\_type [ ( length ) ] , expression [ , style ] )







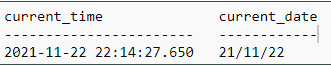








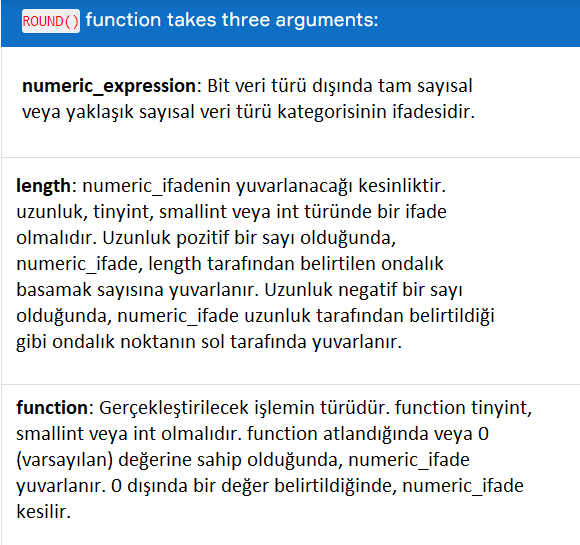




**ROUND() and ISNULL() Function**

ROUND(numeric\_expression , length [ ,function ])

Belirtilen uzunluğa veya duyarlılığa yuvarlanmış sayısal bir değer döndürür.



ROUND, uzunluk negatif bir sayı olduğunda, veri türünden bağımsız olarak yuvarlatılmış bir sayısal\_ifade döndürür.









Aşağıdaki örnek, ROUND kullanılarak son basamağın her zaman bir tahmin olduğunu gösteren iki ifadeyi göstermektedir.





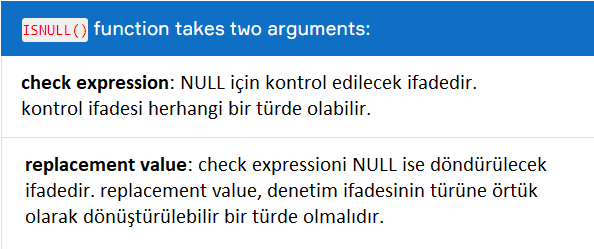
Aşağıdaki örnek, yuvarlama ve kesme arasındaki farkı göstermek için iki SELECT ifadesi kullanır. İlk ifade sonucu yuvarlar. İkinci ifade sonucu keser.





ISNULL(check expression, replacement value)

NULL'u belirtilen değiştirme değeriyle değiştirir.











Tablo değerleri üzerinde çalışıyorsanız, yukarıdaki sorguda "1" ve "2" yerine sütun adları yazabilirsiniz.

Örneğin, ortalama liste fiyatını hesaplamak istiyorsunuz. Ancak sütunda boş değerler var. Boş değerleri belirli bir değerle doldurmak istiyorsanız ISNULL() işlevini kullanabilirsiniz.

**COALESCE, NULLIF ve ISNUMERIC İfadeleri**

COALESCE(expression [, ...n])

Argümanları sırayla değerlendirir ve başlangıçta NULL olarak değerlendirilmeyen ilk ifadenin geçerli değerini döndürür.





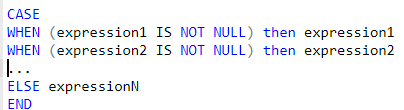
üçüncü değer boş olmayan ilk değer olduğundan üçüncü değeri döndürür.

**💡İpuçları:**

Tüm bağımsız değişkenler NULL ise, COALESCE NULL değerini döndürür. Boş değerlerden en az biri NULL yazılmalıdır.

COALESCE ve CASE'in karşılaştırılması

COALESCE ifadesi, CASE ifadesi için sözdizimsel bir kısayoldur. Yani COALESCE(expression1,...n) kodu, sorgu iyileştirici tarafından aşağıdaki CASE ifadesi olarak yeniden yazılır:











NULLIF(expression, expression)

Belirtilen iki ifade eşitse boş bir değer döndürür.





iki giriş değeri aynı olduğundan ilk sütun (4 ve 4) için NULL döndürür. İkinci sütun, iki giriş değeri farklı olduğu için ilk değeri (5) döndürür.









**💡İpuçları:**

Fiyatı değişmeyen ürünü bulmak için NULLIF() işlevini kullanabilirsiniz. Tüm bağımsız değişkenler NULL ise, COALESCE NULL değerini döndürür. Boş değerlerden en az biri NULL yazılmalıdır.

ISNUMERIC(expression)

Bir ifadenin geçerli bir sayısal tür olup olmadığını belirler.

Giriş ifadesi geçerli bir sayısal veri türü olarak değerlendirildiğinde 1 döndürür; aksi takdirde 0 döndürür. Geçerli sayısal veri türleri şunlardır: bigint, int, smallint, tinyint, bit, decimal, numeric, float, real, money, smallmoney.







