SQL Fonksiyonlarının Kullanımı

SQL fonksiyonları, SQL kullanımında güçlü özellikler katar. Fonksiyonlar veya işlevler, kullanıcıya bazı özel işlemleri ve hesaplamaları otomatik olarak sunar. Kullanıcının, fonksiyonun adını ve gerekli argümanları tanımlaması yeterlidir. Bu bilgilere dayalı olarak fonksiyon, bir sonuç döndürür (üretir).

SQL fonksiyonları iki ana grup altında incelenir: Tek satır fonksiyonları ve çoklu satır fonksiyonları. Tek satır fonksiyonları, tablonun her bir satırına uygulanır ve her satır için bir sonuç üretir. Çoklu satır fonksiyonları ise, bir grup satıra uygulanır ve sonuç buna göre elde edilir.

Karakter Fonksiyonları:

- **LEFT(str, n):** Metnin solundan n karakter alır.
- **RIGHT(str, n):** Metnin sağından n karakter alır.
- **UPPER(str):** Metni büyük harfe çevirir.
- **LOWER(str):** Metni küçük harfe çevirir.
- **LTRIM(str):** Metnin başındaki boşluk karakterlerini atar.
- **RTRIM(str):** Metnin sonundaki boşluk karakterlerini atar.
- **REPLACE(str, old_str, new_str):** Bir ifadeyi başka bir ifade ile değiştirir.
- **LEN(str):** Bir alandaki karakterlerin uzunluğunu verir.

Sayısal Fonksiyonlar:

- **ROUND(number, decimals):** Sayıyı belirtilen ondalık basamak sayısına yuvarlar.
- **ABS(number):** Sayının mutlak değerini alır.
- **CEILING(number):** Sayıyı yukarı yuvarlar.
- **FLOOR(number):** Sayıyı aşağı yuvarlar.
- **POWER(number, exponent):** Sayının belirtilen üsse göre değerini hesaplar.

Tarih Fonksiyonları:

- **GETDATE():** Geçerli tarih ve saat değerini döndürür.
- **DATEADD(datepart, number, date):** Belirtilen tarih kısmına belirli bir sayıyı ekler.
- **DATEDIFF(datepart, startdate, enddate):** İki tarih arasındaki farkı belirli bir tarih kısmında döndürür.
- **DATENAME(datepart, date):** Belirtilen tarih kısmının adını döndürür.
- **DATEPART(datepart, date):** Belirtilen tarih kısmının sayısal değerini döndürür.

Verileri Gruplayarak Analiz Etmek

SQL içinde aynı satırların bir defa yazılması gerektiğinde DISTINCT komutu kullanılır.

SELECT DISTINCT nakliyeciFirmaAdi FROM siparisler, nakliyeciler

Birleştirme/eşleştirme işlemleri için JOIN komutları kullanılır. Temel JOIN türleri:

- **INNER JOIN:** İki tablodaki eşleşen kayıtları döndürür.

```
SELECT a.kolon1, b.kolon2
FROM tablo1 a
INNER JOIN tablo2 b ON a.ortak_kolon = b.ortak_kolon
```

- **LEFT JOIN:** Sol tablodaki tüm kayıtları ve sağ tablodaki eşleşen kayıtları döndürür.

```
SELECT a.kolon1, b.kolon2
FROM tablo1 a
LEFT JOIN tablo2 b ON a.ortak_kolon = b.ortak_kolon
```

- **RIGHT JOIN:** Sağ tablodaki tüm kayıtları ve sol tablodaki eşleşen kayıtları döndürür.

```
SELECT a.kolon1, b.kolon2
FROM tablo1 a
RIGHT JOIN tablo2 b ON a.ortak_kolon = b.ortak_kolon
```

- **FULL OUTER JOIN:** İki tablodaki tüm kayıtları döndürür.

```
SELECT a.kolon1, b.kolon2
FROM tablo1 a
FULL OUTER JOIN tablo2 b ON a.ortak_kolon = b.ortak_kolon
```

Veriler **GROUP BY** kullanımıyla gruplandırılır.

```
SELECT musteriID, COUNT(*) AS siparisSayisi
FROM siparisler
GROUP BY musteriID
```

HAVING deyimi, gruplandırılmış verilere koşul eklemek için kullanılır.

```
SELECT musteriID, COUNT(*) AS siparisSayisi
FROM siparisler
GROUP BY musteriID
HAVING COUNT(*) > 1
```

Toplam satış miktarını hesaplama örneği:

```
SELECT SUM(satisMiktari) AS toplamSatis
FROM siparisler
```

T-SQL ve View

T-SQL, Microsoft SQL Server'ın gelişmiş sorgulama dilidir. Değişken türleri ve tanımlama kuralları bulunur. T-SQL ile yığın işlemleri ve kontrol deyimleri (IF-ELSE, CASE, WHILE, BREAK, CONTINUE) kullanılabilir.

View (Görünüm) Nesnesi: Görünümler, sanal tablolardır ve fiziksel yer kaplamazlar. Birden fazla tabloyu birleştirip sorgulama yapılmasını sağlarlar. CREATE VIEW komutuyla oluşturulurlar.

```
CREATE VIEW view_adı AS SELECT sütunlar FROM tablo_adı
```

Değişken Türleri ve Tanımlama: Yerel değişkenler @ işaretiyle tanımlanır.

```
DECLARE @Degiskenin_Adi Veri_Tipi
```

Global değişkenler @@ işaretiyle tanımlanır.

```
DECLARE @@Degiskenin_Adi Veri_Tipi
```

Değişkenlere Değer Atama:

- SET komutu ile

```
SET @Degisken = deger
```

- SELECT komutu ile

```
SELECT @Degisken = kolon
FROM tablo
WHERE koşul
```

Kontrol Deyimleri:

- IF-ELSE: Belirli bir koşul doğruysa bir blok kod, yanlışsa başka bir blok kod çalıştırılır.

```
IF koşul
BEGIN
 -- kod bloğu
END
ELSE
BEGIN
 -- kod bloğu
END
```

- CASE: Koşullara göre farklı değerler döndürür.

```
SELECT CASE
      WHEN koşul1 THEN deger1
      WHEN koşul2 THEN deger2
      ELSE deger3
     END
```

- WHILE: Belirli bir koşul doğru olduğu sürece kod bloğunu tekrar tekrar çalıştırır.

```
WHILE koşul
BEGIN
 -- kod bloğu
END
```

- BREAK: Döngüyü sonlandırır.
- CONTINUE: Döngünün geri kalan kısmını atlayarak bir sonraki yinelemeye geçer.

Stored Procedure

Stored Procedure (SP), belli bir görevi yerine getirmek için yazılmış program parçacıklarıdır. SP'ler dört kategoriye ayrılır:

- Extended Stored Procedure: .dll uzantılı prosedürler, T-SQL dışında C, C++, #C, Basic, Delphi gibi dillerde yazılıp derlenirler.
- CLR Stored Procedure: SQL Server 2005'ten sonra CLR ortamında herhangi bir dil kullanarak kodlanan SP'ler.
- Sistem Stored Procedure: Genellikle sp_ ön ekiyle başlar, master veritabanında tutulurlar.
- Kullanıcı Tanımlı Stored Procedure: Programcı tarafından programlanan prosedürler. Geçici, yerel ve uzak SP'ler olarak üçe ayrılır.

SP'lerin oluşturulması ve çalıştırılması CREATE PROCEDURE ve EXEC komutları ile yapılır. Parametre kullanımı mümkündür.

```
CREATE PROCEDURE prosedur_ismi (@parametre_ismi veri_tipi, ...)
AS
BEGIN
 SQL ifadeleri
END
```

Örnek: Öğrenciler tablosundan ortalaması 70'in üzerinde olan öğrencileri bulan SP:

```
CREATE PROCEDURE sp_yetmis_ustu AS
BEGIN
 SELECT * FROM ogrenciler WHERE ortalama > 70
END
```

SP'lerin çalıştırılması ve oluşturulması üç aşamada gerçekleşir:

- Ayrıştırma (Parsing): SQL ifadelerinin geçerli olup olmadığı denetlenir.
- Derleme (Compiling): Çalışma planı oluşturulur.
- Çalıştırma (Executing): Çalışma planına göre işlem gerçekleştirilir.

SP'de değişiklik yapmak için ALTER komutu kullanılır.

```
ALTER PROCEDURE Prosedür_ismi AS SQL ifadeleri
```

Fonksiyonlar

Kullanıcı Tanımlı Fonksiyonlar (UDF), belli bir sonucu geri döndürmek için tasarlanmış, bir veya birden fazla yerde kullanılan yapılardır. Kullanıcı tanımlı fonksiyonlar, SP'ler gibi dışarıdan parametre alabilirler ve tampon bellekte saklanırlar. View'lardan farklı olarak dışarıdan parametre alabilir ve sorguların içerisinde kullanılabilirler.

Skaler Fonksiyonlar: Tek bir değer döndüren fonksiyonlar.

```
CREATE FUNCTION fonksiyon_adı (@parametre_adı veri_türü) RETURNS skaler_veri_tipi AS BEGIN SQL_ifadeleri RETURN skaler_deger END
```

Örnek: Bir müşteri kodunu girerek müşterinin sepetinde toplam kaç adet ürün olduğunu bulan fonksiyon.

```
CREATE FUNCTION SepettekiUrunSayisi (@musteriKod INT) RETURNS INT AS
BEGIN
 DECLARE @urunSayisi INT
 SELECT @urunSayisi = COUNT(*) FROM sepet WHERE musteriKod = @musteriKod
 RETURN @urunSayisi
END
```

Tablo Döndüren Fonksiyonlar: Tablo şeklinde veri döndüren fonksiyonlar.

```
CREATE FUNCTION fonksiyon_adı (@parametre_adı veri_tipi) RETURNS TABLE AS RETURN (SELECT ifadeleri)
```

Örnek: Ürün kodu girilen bir ürünün bilgilerini tablo içerisinde listeleyen fonksiyon.

```
CREATE FUNCTION UrunBilgileri (@urunKod INT) RETURNS TABLE AS
RETURN (SELECT urunAdi, urunFiyati FROM urunler WHERE urunKod = @urunKod)
```