**SQL DENETİM DEYİMLERİ**

Birden fazla komutu aynı anda işletebilmek için SQL’de bloklar kullanılmaktadır. Tüm programlama dillerinde olduğu gibi akış kontrollerinde ve döngü yapılarında kullanılan komutlar birden fazla ise mutlaka BEGIN..END bloğunda yazılmalıdır.

**IF... ELSE Yapısı🡪** Bir deyimin işletilmesini belli bir koşula bağlar.

if(şart1) deki şart1 doğru ise if(şart1)’den sonra gelen BEGIN ile END arasındaki kodlar çalıştırılır. Daha sonra if bloğu­nun en altındaki kodlardan devam eder.

Şart1 doğru değil ise, elseif(şart2)deki şart2’ye bakılır. Bu şart doğru ise, şartı takip eden BEGIN ile END ifadeleri arasındaki kodlar çalıştırılır, if bloğunun sonuna gider ve takip eden kodları çalış­tırır. Şart2 de yanlışsa, elseif(şart3)’deki şart3’e bakılır. Şart3 doğru ise takip eden BEGIN ile END deyimleri arasındaki kodlar çalıştırılır, if bloğunun en sonuna gider ve takip eden kodları çalıştırır. Şart3 yanlış ise bu aradaki kodları da çalıştırmadan geçer. Genel yapısı şu şekildedir:

if(şartlar)

begin

....

end

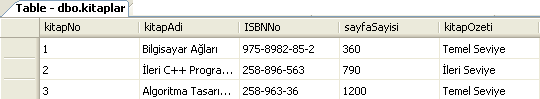
else if(şartlar)

begin

...

end

**Örnek:**



Use dbKutuphane1

Go

declare @sayfaSayisi int

SELECT @sayfaSayisi=MAX(sayfaSayisi)

FROM Kitaplar

if (@sayfaSayisi>700)

BEGIN

print 'Bu kitaplıkta çok kalın bir kitap varmış'

END

Else if (@sayfasayisi<=700)

BEGIN

Print 'Bu Kitaplıktaki kitaplar Çok kalın değil'

END

go

****

**CASE Yapısı🡪** Case yapısı, birçok durum için dallanmaya müsait bir yapıdır. Case yapısı ANSI92 standartları ile uyumlu bir karar döngüsüdür ve oldukça kısa kod ile etkin programlar oluşturma­mıza yarar. Örneğin dışarıdaki kitaplar için ‘dışarıda’ yazabilecek bir kod bu komut sayesinde basitçe yazılabilir.

Genel yapısı şu şekildedir:

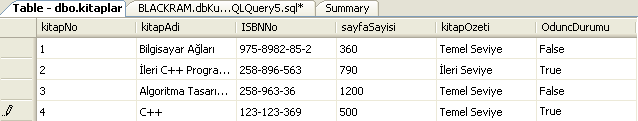
CASE

WHEN þart THEN deðer

[ ELSE deðer]

END

**Örnek:** Kütüphanedeki kitapların mevcut olup olmadığını gösteren basit bir T-SQL kodlarıdır.



Use dbKutuphane1

Go

SELECT kitapNo,oduncdurumu, 'kitapDurumu'=

Case

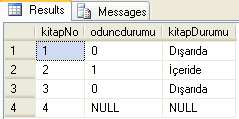
WHEN oduncdurumu = 0 THEN 'Dışarıda'

WHEN oduncdurumu= 1 THEN 'içeride'

End

FROM kitaplar

go



**ISNULL fonksiyonu🡪** ISNULL(kontrol\_edilecek\_deger, null\_ise\_cikacak\_deger) şeklinde kullanılır.

Kontrol\_edilecek\_deger: Bir sütun adı ya da değişken adı olabilir.

null\_ise\_cikacak\_deger: Bir sütun adı, değişken adı veya sabit bir değer olabilir.

declare @degisken VARCHAR(29)

SELECT ISNULL(@degisken,'boþluk ise bu deðer seçilecek’)

**WHILE Döngüsü🡪** Tekrar gerektiren işlemlerde istenilen şart gerçekleşinceye kadar işlem yapmaya olanak sağlar. While ile bir işlemi istenilen kadar tekrarlatabilirsiniz. Genel yapısı şu şekildedir: Şart gerçekleşinceye kadar BEGIN ile END arasına yazılan kodlar işlem görür.

WHILE şart

BEGIN

Tekrarlanması gereken kodlar

END



**Örnek:** Sayaç değişkeni 5 değerini alıncaya kadar tanımlanan toplam değişkenini beşer beşer arttıran programın T-SQL kod satırlarıdır.

Execute edilmesi sonucunda ekran görüntüsü aşağıdaki gibi olur.



**Örnek:** Integer bir değişken tanımlayalım ve değeri 15 oluncaya kadar bir artıralım. 15 olduğunda da yazdıralım:

SELECT @sayac = 1

WHILE (@sayac < 15)

BEGIN

SELECT @sayac = @sayac + 1

END

print ‘sayac :’

print @sayac

**BREAK Komutu🡪**İstenilen şart sağlandığında WHILE döngüsünden çıkmak için BREAK komutu kullanılır. Programın çalışması WHILE’ın END’inin altındaki satırdan çalışmaya devam eder.



**Örnek:** Sayacın 3 olması durumunda WHILE döngüsünden çıkıp toplamı ve sayacın değerini yazdıran T-SQL kodlarıdır.

Execute edilmesi sonucunda ekran görüntüsü aşağıdaki gibi olur.



**CONTINUE Komutu🡪** Programın çalışmasını WHILE yapısının başına göndermek için kullanılan bir komuttur.

**Örnek:** Sayaç değişkenin 3 değeri hariç diğer değerleri ve toplamı yazdıran T-SQL kodlarıdır.

Execute edilmesi sonucunda ekran görüntüsü aşağıdaki gibi olur.



Sonuç ekranında 3 değerinin yazılmadığını görünüz.