**T-SQL İLE ÇALIŞMAK**

ANSI SQL, veritabanı ile ilgili veri işleme ve düzenleme işlev­leri için geliştirilmiş bir standart dildir. Ancak bu dil zaman zaman bir programlama dili olmadığından kayıt işlemede yetersiz kalabilmektedir. Çünkü gelişmiş programlama dillerinde yer aldığı halde bu dilde WHILE, IF gibi temel mantık ve döngü yapıları mevcut değildir. Değişken kullanımı da tanımda yer almaz. İşte Transact-SQL bu nok­tada devreye giren ANSI 92 dahilinde bir standart dışı dildir.

T-SQL’in geliştirilmesindeki en temel amaç, VTYS dışında döngü veya mantıksal karşılaştırma gerektiren konu­larda ayrıca bir derleyiciye, programlama dili öğrenmeye, sistem oturtmaya gerek kalmadan bu türden işlemleri yerine getirebilecek olanakları sağlamaktır. Bir sonraki konuda yer alan stored procedu­res, triggers gibi programcıkları yazarken bu dili kullanacağız.

**DİKKAT:**

T-SQL Microsoft SQL Server ve Sybase için geçerli bir dildir. Oracle kullananların, PL/SQL öğrenmeleri gerekir. Ancak bu iki dil birbiri ile yaklaşık aynı özellikleri haizdir.

Bu projede kullandığımız veritabanını oluşturalım:

CREATE database dbKutuphane1

go

use dbKutuphane1

go

CREATE TABLE kitaplar(

kitapNo INTEGER NOT NULL,

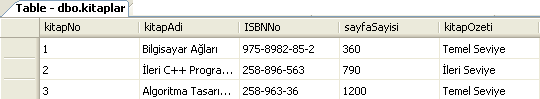
kitapAdi VARCHAR(63) NOT NULL,

ISBNNo VARCHAR(15),

sayfaSayisi INTEGER,

kitapOzeti VARCHAR(255))

go



**DEĞİŞKENLER**

Değişken: Programlama dillerinde, değeri daha sonra akışlara göre değişecek bir değer için hafızada(RAM’de) bir yer ayırmak için kullanılan yapıdır. Hafıza­daki bu değere daha sonra değişkenin adı ile erişilir ve yeni değeri atanır. Değişkene atanan her yeni değer bir eski değeri siler.

Değişken, verilerin bellekte geçici olarak kaydedilmesini ve gerektiğinde kullanılmasını sağlayan değerdir. T-SQL kullanmanın en büyük kolaylıklarından biri de değişken kullanımına olanak tanımasıdır. Burada ifade edilen; değişken diğer tüm programlama dillerinde yer alan bir veri tipi ile sınırlandırılmış, oluşturulmasının ardından hafızada belli bir yer kaplayan, üzerine veri ataması yapılabilen ve daha sonra ismi kullanılarak program içerisinden çağrılıp kullanılabilecek yapıdır.

Değişkenler şu şekilde tanımlanır:

declare @degisken\_adi veritipi[(boyut)]

**Örnek:**

declare @kitapNo INT

declare @kitapAdi VARCHAR(63)

go

Aralara virgül koyarak da birden fazla değişkeni tek bir DECLARE ifadesi ile oluşturabilirsiniz.

declare @kitapNo INT,@kitapAdi VARCHAR(63)

go

Değişkenlere değer atama,

declare @kitapNo INT, @kitapAdi VARCHAR(63)

SET @kitapNo=255

SET @kitapAdi= ‘107 Kimya Öyküsü’

SET @kitapNo=256 -- Artýk kitapNo deðiþkeninin deðeri 256,255 silindi.

go

Şeklinde yapılır.

Ancak T-SQL’in asıl amacı SQL’in yeteneklerini artırmaktır. Bundan dolayı değişkenlerin en genel kullanım amacı, bir sorgunun sonucundaki değerlerden birini alıp bir değişkene aktarmaktır.

declare @enSonEklenenKitap INT

SELECT @enSonEklenenKitap=MAX(kitapNo)

FROM Kitap

go

SQL Server'da da değişkenler yerel ve genel olmak üzere ikiye ayrılır.

1. Yerel değişkenler, "@" ön eki ile tanımlanır (@değişken).
2. Genel değişkenler ise SQL Server tarafından tanımlanmıştır ve kullanıcı tarafından oluşturulamaz. "@@" ön eki ile tanımlanırlar (@@SERVERNAME). Genel değişkenler genellikle SQL Server hakkındaki bilgileri verir. SQL Server'da tanımlanmış birçok genel değişken vardır.

Bir veritabanına aynı anda bir çok kişi oturum açabilir. Bir tek oturumda geçerli değişkenlere yerel değişken denir. Bazen bütün oturumlarda geçerli değişkenlere ihtiyaç duyulur. Bu tür durumlarda, ortam değerlini ayarlamak üzere veya çeşitli durumları tanımlamak üzere fonk­siyonlar kullanılır. Bu fonksiyonların bir listesini kitabın sonundaki ek kısmında bulabilirsiniz.

**Nesne ve Değişken İsimlendirme Kuralları**

* + Nesne veya değişkene bir isimlendirme yaparken aşağıdaki kurallara dikkat etmelisiniz:
  + Harf veya alt çizgi (\_) ile başlamalıdır.
  + Türkçe karakterler ve boşluk isimlendirmede kullanılmamalıdır.
  + Değişken ismi SQL’de özel anlamı olan sembollerle (@, @@, #, ##, $) başlamamalıdır.
  + T-SQL komutları değişken ismi olarak verilmemelidir (SELECT,UPDATE vb).
  + SQL ifadeleri prensip olarak büyük harfle yazılır.
  + Nesne isimleri kısa ve anlamlı olmalıdır.
  + Nesne isimlendirilirken işlerin kolaylaştırılması açısından tekil isim tercih edilmelidir (TabloOgrenciler yerine tblOgrenci gibi).
  + NULL terimi, daha önce hiçbir şey girilmemiş (değersiz) anlamındadır.
  + Klavyedeki SPACE (ASCII 32) tuşu ile NULL aynı değerleri içermez. NULL boş veya bilinmeyen değerler için kullanılır.

Varchar, int türlerinde değişken tanımlayabildiğiniz gibi tablo türünde değişken de tanımlayabiliriz.

**Örnek:** DECLARE @değişken\_adı TABLE (tablo tanımı) şeklindedir.

Bir değişken oluşturulduğunda NULL değere sahiptir. Değişkenlere değer atamanın SET, SELECT ve tablolar için INSERT INTO gibi birkaç farklı şekli vardır.

**SELECT ifadesiyle değer atama**

SELECT @değişken\_adı=değer şeklinde yapılır.

**Tablo değişkenlere INSERT INTO ifadesi ile değer atama**

INSERT INTO @tablo\_degisken SELECT adi, soyadi FROM person ifadesi ile person tablosunun adı ve soyadı sütunlarının içerdiği değerlerden oluşan bir tabloyu @tablo\_değişken adlı değişkene atamış olursunuz.

**AÇIKLAMA SATIRLARI**

T-SQL'de bir satırın dikkate alınmamasını istiyorsanız "--" kullanabilir ya da /\*........\*/ kullanabilirsiniz.

**Örnek:**

-- CREATE TABLE ogrenci

/\* CREATE TABLE ogrenci \*/

Şeklindeki kullanımlardan birini seçebilirsiniz.

**Yığın Kavramı**

SQL Server’da yığın, sorguların sırayla işleme alınması demektir. Çalışma esnasındaSQL Server’a gönderilen birden fazla sorgu yığınlar hâlinde ele alınır.

**GO Komutu**

Bir yığının sonunu belli etmek için **GO** komutu kullanılır. Bir yığın SQL Server’da işlenmeye başladığı anda önce Parse(ayrıştırma) edilir. Daha sonra derlenerek (Compile) çalıştırılır(Execute).

**Genel Yazımı**

Komutlar

Komutlar

GO

**USE Komutu**

T-SQL’de çalışacağınız veri tabanını belirme işlemi için bu komut kullanılmalıdır. USE komutuyla hangi veri tabanı üzerinde işlem yapılacağı belirtilir.

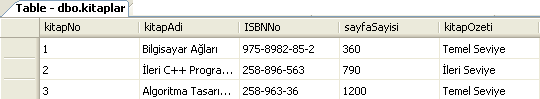
**Genel Yazımı🡪** USE Veritabani\_Adi

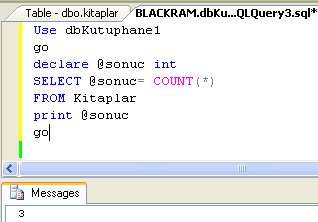
**PRINT Komutu**

Değişkenlerin değerlerini, hataları vb. diğer ihtiyaç duyulan olaylarda PRINT komutu kullanılır. DEBUG işlemi için ihtiyaç duyulan bir komuttur.

**Genel Yazımı🡪** PRINT @degisken\_adi

**Örnek:** Kitap numarasını SQL sorgusu ile bula­lım ve bu sonucu yazdıralım:





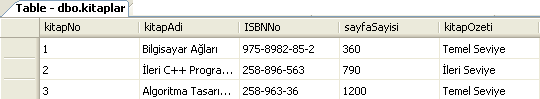
**İŞLEM OPERATÖR TÜRLERİ**

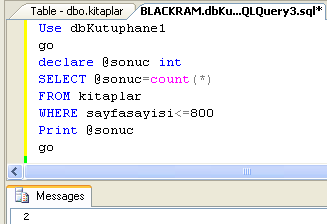
T-SQL’de işlem yapabilmek için bazı operatörlere ihtiyaç duyulur.

|  |  |
| --- | --- |
| **OPERATÖR** | **ANLAMI** |
| < | Küçük |
| > | Büyük |
| = | Eşit |
| <= | Küçük veya eşit |
| >= | Büyük veya eşit |
| <> | Eşit değil |
| != | Eşit değil |
| LIKE | Metin Karşılaştırma Operatörü |

**Karşılaştırma Operatörleri**

Karşılaştırma ifadesinde karşılaştırılan verilerin türü aynı olmalıdır. Yani karakter veri türü ile ancak karakter türünde başka bir veri; bir sayısal veri ile ancak sayısal olan başka bir veri karşılaştırılabilir.





**Örnek:** Yandaki örnekte Sayfa sayısı 800 den küçük olan kitap sayısı verilmiştir.

**Joker Karakterler**

Sadece LIKE operatörüyle kullanılan joker karakterler, bir veya daha fazla harfin yerine geçer. Belirli aralıklardaki belli harfle başlayan ve biten sorgularda joker karakterler kullanılır.

**Joker Karakterler Anlamı**

* % Birden fazla harf ya da rakamın yerini tutar.
* \_ Bir tek harf veya rakamın yerini tutar.
* [HARF] Herhangi bir harf yerine gelebilecek harfleri belirtir.
* [^HARF] Herhangi bir harf yerine gelemeyecek harfleri belirtir.
* [A-Z] A ile Z arasındaki harfleri belirtir.

**Örnek:** tablo1 tablosundaki ad alanı içinde baş tarafı “Er” ile başlayan isimleri listeleyen TSQL kodlarıdır.

SELECT \*

FROM tablo1 WHERE ad LIKE '%Er%'

**Örnek:** tablo1 tablosundaki soyad alanı içinde baş tarafı “DE” ile başlayan soyadları listeleyen T-SQL kodlarıdır.

SELECT \*

FROM tablo1 WHERE soyad LIKE '%DE%'

**Mantıksal Operatörler**

NOT, OR ve AND mantıksal operatörleri yardımı ile birden çok koşulun gerçekleşmesine bağlı olarak ifade edilebilecek karmaşık ya da birleşik koşullu listelemeleri gerçekleştirmek mümkün olmaktadır. BETWEEN ifadesi de AND operatörü ile aynı işlemi gerçekleştirebilmektedir.

**Örnek:** Doğum tarihi 1970’ten önce olan, maaşı 700 – 1200 arasında olan personeli listeleyen kodlardır.

SELECT \*

FROM Person

WHERE dogum\_tarih<{01/01/70} AND

Maas>=700 AND maas<=1200

**Örnek:** BETWEEN..AND yapısıyla, maaşı 700 ile 1200 YTL arasında olanları gösteren T\_SQL kodlarıdır.

SELECT \*

FROM Person

WHERE maas BETWEEN 700 AND 1200

**Örnek:** Doğum tarihi 1970’ten büyük ve cinsiyeti erkek olan veya doğum tarihi 1975’ten büyük ve cinsiyeti kadın olan personeli listeleyen T-SQL kodlarıdır.

SELECT \*

FROM Person

WHERE dogum\_tarih>={01/01/70} AND cinsiyet=Erkek OR dogum\_tarih>={01/01/75} AND cinsiyet=Kadın

AND operatörü OR operatörüne göre daha önceliklidir.

**Örnek:** Sinif tablosunda adı Ali olmayanları listeleyen T-SQL kodlarıdır.

SELECT \* FROM Sinif

WHERE NOT ad=’Ali’

**Örnek:** OR operatörü yerine In operatörü de kullanarak kod satırlarını yazabiliriz.

SELECT \*

FROM Ogrenci

WHERE bolum=’Bilgisayar’ OR bolum=’Elektronik’ OR bolum=’Elektrik’

SELECT \*

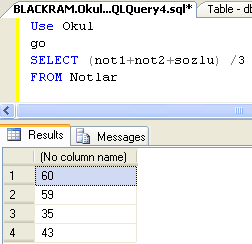
FROM tablo1

WHERE bolum In ('Bilgisayar','Elektronik','Elektrik')

|  |  |
| --- | --- |
| **OPERATÖR** | **İŞLEVİ** |
| % | Mod alma |
| \* | Çarpma |
| / | Bölme |
| + | Toplama |
| - | Çıkarma |

**Aritmetiksel Operatörler**

T-SQL’de kullanılan aritmetiksel operatörler şunlardır.

SELECT komutu ile veri tabanında mevcut tablolardan listeleme yaparken tabloda ayrı bir sütun (alan) olarak yer almamış ve ancak bir hesaplama sonucunda üretilebilecek bilgileri de listeleme içine almak mümkündür.

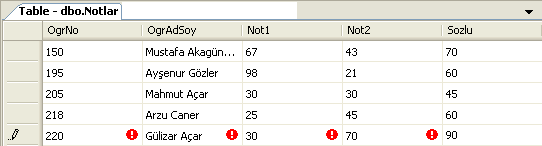
**Örnek:** Ogr\_Notlar tablosunda notların not ortalamasını hesaplayan T-SQL kodlarıdır.

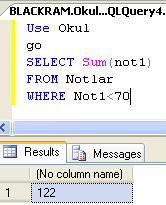
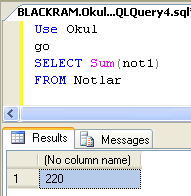
Öncelik sırası, matematikte ve diğer bilgisayar dillerinde olduğu gibidir. Üs alma, hepsinden öncedir. Sonra çarpma (\*) ve bölme (/) gelir. Toplama (+) ve çıkarma (-) en son önceliklidir. Parantez kullanılarak öncelik sırası değiştirilebilir.

**FONKSİYONLAR**

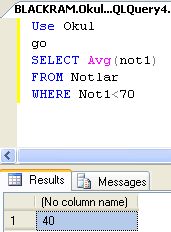
**Kümeleme Fonksiyonları**

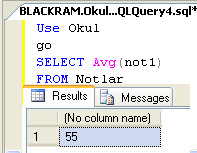
SQL, tablo içinden çeşitli matematiksel işlemlerin sonucunu otomatik olarak üretmeyi sağlayan fonksiyonlara sahiptir.

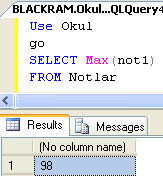
****

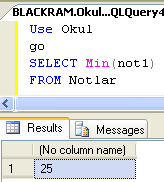


**SUM Fonksiyonu🡪** Fonksiyonla belirtilen sütun ile ilişkili olarak toplama işlemini yapar.



**AVG Fonksiyonu🡪** Aritmetiksel ortalama (avarage) hesaplamak için kullanılır.





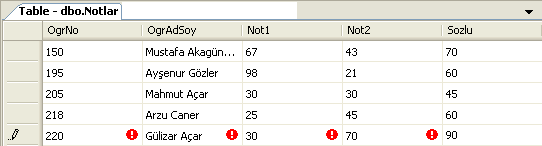
**MAX Fonksiyonu**

Tablo içinde, belirtilen sütun (alan) içindeki en büyük değeri bulur.

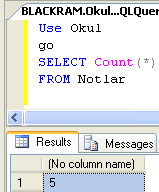
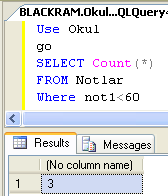
**MIN Fonksiyonu**

Tablo içinde belirtilen sütun (alan) içindeki en küçük değeri bulur.

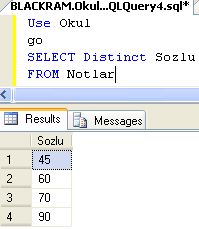
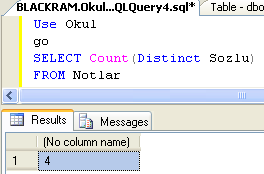
**COUNT Fonksiyonu**

****

Tablo içerisinde herhangi bir sayma işlemi gerçekleştirmek için kullanılır.

COUNT fonksiyonu, DISTINCT sözcüğü ile de kullanılabilir. DISTINCT, her bir kaydın tekil olarak yer alması istendiğinde bu deyim kullanılır. Normalde SELECT ile aynı özelliğe sahip birden fazla kayıt listelenebilir.

** **

**T-SQL’de Gruplandırma**

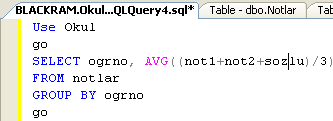
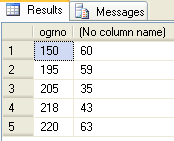
“GROUP BY” yardımcı sözcüğü bir alana göre kayıtları gruplamak için kullanılır. Genel kullanımı aşağıdaki gibidir:

SELECT [ DISTINCT | ALL ] <sütun(lar)> FROM <tablo adý (lar)>

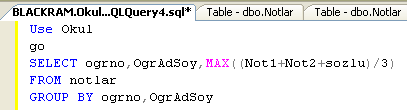
[ WHERE <þart (lar)> ]

[ GROUP BY <sütunlar>]

**Örnek:** Öğrenci notları tablosunda öğrenci numarasına göre her bir öğrencinin almış olduğu not ortalamalarını gösteren T-SQL kodlarıdır.

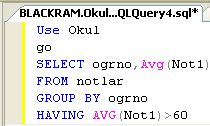
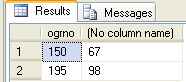
 

**Örnek:** Öğrenci notları tablosunda öğrenci numarasına ve bölüm adına göre her bir öğrencinin almış olduğu not ortalamalarını gösteren T-SQL kodlarıdır.



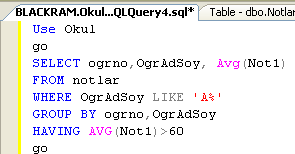
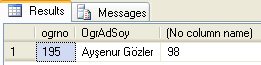
Gruplandırarak kümeleme fonksiyonlarını uygularken koşul da verilebilir. Bu durumda, grup üzerindeki hesaplamalarla ilişkili koşul belirtirken **HAVING** sözcüğünü kullanmak gerekir.

**Örnek:** Öğrenci notları tablosunda öğrenci numarasına göre her bir öğrencinin almış olduğu not ortalamasını 70’den büyük gösteren T-SQL kodlarıdır.

HAVING sözcüğü, SELECT komutunda GROUP BY sözcükleri bulunmadığı zaman geçersizdir. HAVING sözcüğünü izleyen ifade içinde SUM, COUNT (\*), AVG, MAX ya da MIN gibi kümeleme fonksiyonlarından en az biri bulunmalıdır. WHERE sözcüğü bir tablonun tek tek satırları üzerinde işlem yapan koşullar için geçerli iken HAVING sözcüğü, sadece gruplanmış veriler üzerindeki işlemlerde geçerlidir.

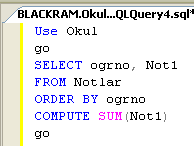
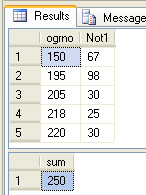
**Örnek:** Öğrenci notları tablosunda öğrenci numarasına göre bölümü “Bilgisayar” ve öğrencinin almış olduğu not ortalaması 65’ten büyük olanları gösteren T-SQL kodlarıdır.

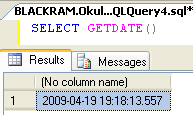
**COMPUTE ve COMPUTE BY Deyimleri**

Toplama fonksiyonunu kullanarak sonuç olarak bir toplam değeri üretir. COMPUTE deyimi her SELECT ifadesiyle kullanılır. COMPUTE BY ise ORDER BY deyimine gerek duyar.

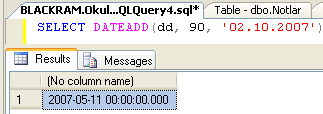
**Örnek:** Öğrenci notları tablosundaki öğrenci numarasına göre ortalama sütunundaki değerleri toplayarak bir sonuç üreten T-SQL kodlarıdır.

**Tarih ve Zaman Fonksiyonları**

Tarih ve zaman üzerinde işlem yapmayı sağlayan fonksiyonlardır.

**GETDATE () Fonksiyonu🡪** Şimdiki tarih ve saat değerini gösterir.

**DATEADD () Fonksiyonu🡪**Verilen bir tarihe istenilen sayıda bir tarih bilgisi eklemek için kullanılır.

DATEADD fonksiyonunda dd gün, mm ay, yy yıl anlamındadır.

**Örnek:** Belirtilen tarih bilgisine 90 gün ekleyen T-SQL kodudur.

**DATEDIFF () Fonksiyonu 🡪**Belirtilen iki tarih arasındaki gün sayısını göstermektedir.

**Örnek:** SELECT DATEDIFF(dd, '04.04.1974', '02.10.2007')

Ayrıca hafta için wk, saat için hh, dakika için mi, saniye için ss kullanılabilir.

**DATEPART () Fonksiyonu🡪**Tarihle ilgili sayısal bilgilerin alınmasını sağlar.

**Örnek:**

SELECT DATEPART(dd, '01.04.1974')

SELECT DATEPART(mm, '01.04.1974')

SELECT DATEPART(yy, '01.04.1974')

**Karakter Fonksiyonları**

Karakter alanlarla ilgili işlem yapmak için bu fonksiyonlar kullanılır.

**CHAR ()🡪** ASCII kodu verilen karakteri görüntüler.

SELECT CHAR(65) A harfini verir.

SELECT ASCII(A) 65 rakamını verir. Değer int tipindedir.

**CHARINDEX()🡪**Bir metin içerisindeki metin parçasını istenilen konumdan itibaren arar.

SELECT CHARINDEX ('bu',@soyad,1)

**LEFT ()🡪**Metnin baş taraftan itibaren istenilen sayıdaki harflerini alır.

SELECT LEFT (@ad,2)

**RIGHT ()🡪**Metnin sonundan istenilen kadar karakteri almak için kullanılır.

**LEN ()🡪** Metnin uzunluğunu veren fonksiyondur.

SELECT LEN(@soyad)

**LTRIM ()🡪**Metnin başında bulunan boşlukları siler.

**RTRIM ()🡪**Metnin sonundaki boşlukları siler.

**LOWER ()🡪**Metni küçük harfe çevirir.

**UPPER ()🡪**Metnin tümünü büyük harfe çevirir.

**REVERSE ()🡪**Metni ters çevirir.

**SUBSTRING ()🡪**Bir metinde belirtilen karakterden itibaren belirli sayıda karakter almak için kullanılır.

**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

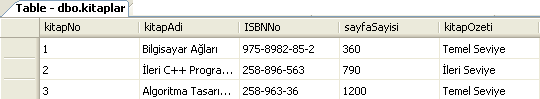
else if(şartlar)

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:

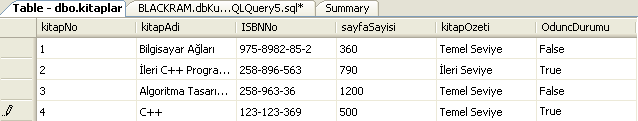
CASE

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

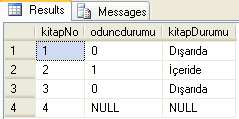
WHEN oduncdurumu = 0 THEN 'Dışarıda'

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.