**VERİTABANI –II**

# 3.TRANSACT SQL (T-SQL)

## 3.1 OUTPUT İŞLEMİ

SQL server üzerinde ekleme, silme ve güncelleme gibi işlemler yapıldığında veri tabanında bulunan kayıtlar üzerinde değişiklikler meydana gelir. Bazı durumlarda programlayıcılar hangi veriler üzerinde değişildiğin meydan geldiğini görmek istemektedir. Veri tabanı programlama yapılırken etkilenen kayıtların görülmesi işlem yapılırken çoğunlukla output ifadesi kullanılmaktadır. Output ifadesi ile yapılan değişiklikler sonuncunda etkilenen satırlar geri dönmekte ve bu değerler istendiği taktirde bir tablo tipi değişkene veya fiziksel bir tabloya kayıt edilebilmektedirler. Bir örnek ile konuyu daha iyi anlatmak gerekirse, Öğrencilerin ders notlarının bulunduğu bir tablo düşünüldüğünde ve bütün öğrencilerin notlarına %5 puan ekleme yapılırsa bu değişikliler sonrasında hangi öğrencinin notunda nasıl bir değişiklik yapıldığını gözlemlemek amacı ile output ifadesi kullanılabilir. Output ifadesini kullanmadığımız durumda bütün değerlerin ilk ve değişimden sonraki hallerinin tutulması gerekmekte ve işlem sonucunda bire bir karşılaştırılma yapılarak değişikliklerin görülmesi sağlanacaktı. Bu işlemde gerek yazılımsal olarak gerekse zaman olarak performans düşüşlerini beraberinde getirmektedir.

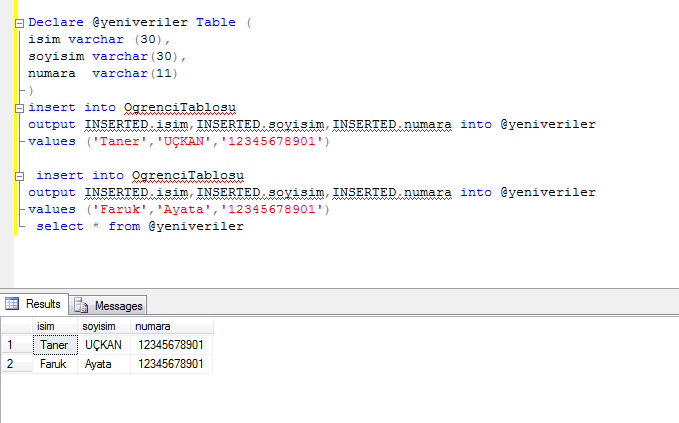
Output ifadesi ile değişiklikten etkilenen satırları bulabilmek için bu ifade ile birlikte INSERTED ve DELETED ifadeleri kullanılmaktadır. Hangi ifadenin hangi durumlarda kullanıldığına dair tablo aşağıdaki gibidir.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Instert işlemi** | **Update İşlemi** | **Delete İşlemi** |
| **Deleted** | Kullanılmaz | Kayıtların Eski Durumu | Silinen Kayıtlar |
| **Inserted** | Yeni Kayıt Eklendiğinde | Değişen Kayıtlar | Kullanılmaz |

Output ifadesi kullanılırken öncelikle hangi alanların kayıt altına alınacağı ve nereye aktarılacakları belirtilmelidir.Daha önce belirttiğimiz gibi bu işlemde aktarılacak yer olaraka tablo tipinde geçici tablolar kullanılabilmektedir.

OUTPUT alınacak\_alanlar INTO aktarılacak\_alan

**Örnek;** Elimizde bulunan öğrenci tablosuna yeni bir kayıt eklediğimizi düşünelim. Yeni eklenen öğrenci bilgilerinin bir değişkende tutularak son olarak hangi değerlerin eklendiğini görmek istediğimizi düşünelim.



Yukarıdaki işlemde adım adım aşağıdaki işlemler yapılmıştır.

* Verilerin tutulacağı alanlar oluşturuldu
* Ekleme işlemini gerçekleştirecek sql kodları yazıldı
* Etkilenecek satırlar öncesinde tanımlanan alanlara INSERTED fadesi kullanılarak eklendi
* Son olarak geçici değişkenin içeriği listelenerek etkilenen satırlar ve içerikleri listelendi

## 3.2 T-SQL AKIŞ KONTROL İFADELERİ

Herhangi bir programlama dili ile programlama yapılırken çoğu durumda döngü ifadeleri, koşul ifadeleri ve sonlandırma gibi akış kontrol ifadeleri kullanılmaktadır. Programalma dillerinin söz dizimlerine bağlı olarak değişkenlik gösterebilen bu yapılar T-sql programlama dilinde kullanıldıklarında ifadenin etki alanı BEGIN ve END ifadeleri kullanılarak belirlenmektedirler.

### 3.2.1 IF….ELSE ifadesi

Programlama yapılırken işlem akışı gereği bir koşul ifadesi gerektiği durumlarda if-else yapısı kullanılmaktadır. Bu yapının anlamı koşul sağlandığı durumda if bloğuna ait kodların çalışması eğer koşul sağlanmıyorsa else bloğuna ait kodların çalışması gerekmektedir.

if koşul ifadeleri

begin

--Şartın sağlanması durumunda çalışacak kısım

end

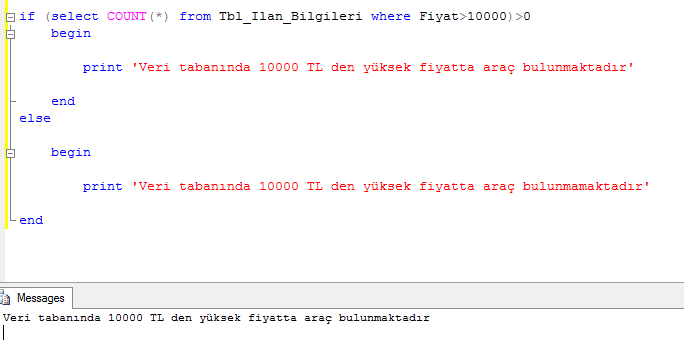
else

begin

--Şartın sağlanmaması durumunda çalışacak kısım

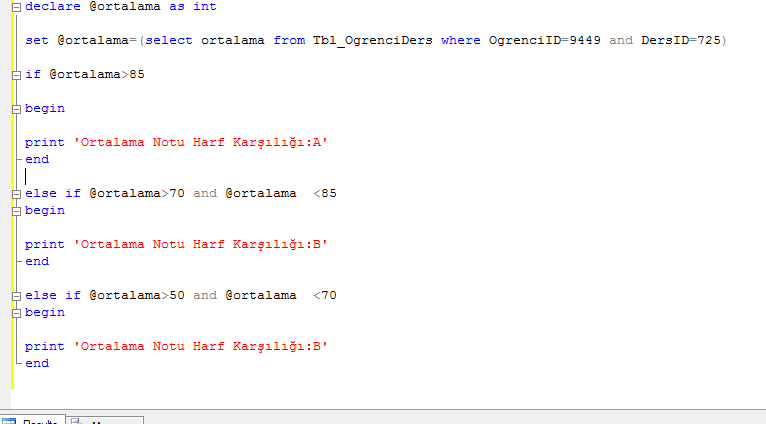
End

Örnek;Elimizde bulunan araç ilan tablosunda fiyatı 10000’den büyük olan aracın olup olmadığını kontrol eden t-sql ifadesini yazalım.



Count ifadesi ile sorgu sonucunda dönen kayıt adedi kontrol edilmiş ve sıfırdan büyük olması durumda o koşulu sağlayan kayıtların olup olmadığı kontrolü yapılmıştır.

Örnek:



### 3.2.2 CASE İfadesi

Bir önceki adımda da anlatıldığı üzere program içerisinde şart –koşul ifadeleri bulunuyorsa if-else ifadesiden faydalanılmaktadır. Koşul ifadeleri birden fazla olduğu durumlarda daha basit ve daha az kod yapısına sahip olan CASE ifadesi de kullanılabilmektedir. Mantık olarak if –else ifadesi ile aynı görevi yapmakta fakat programlama esnasında kullanım kolaylığı bulunmaktadır. Öğrenci Notları tablosunda bulunan öğrencilerin almış oldukları notların bulundukları aralaıklara göre sınıflandırılması işlemini Case ifadesi ile yapmak mümkündür.Her bir koşulu bir Case ifadesi olarak belirttikten sonra yapılması gerek işlem aynı satırda belirtilebilmektedir.

Kullanım şekli;

Case Kullanılacak\_Kontrol \_Değeri

When Alacağı\_değer Veya Karşılaştırma ifadesi THEN Yapılacak işlem

When Alacağı\_değer Veya Karşılaştırma ifadesi THEN Yapılacak işlem

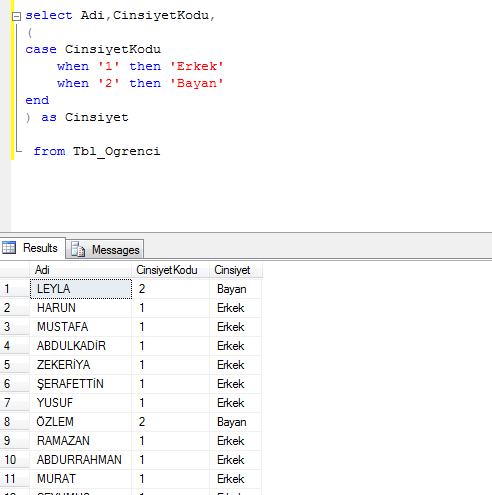
…..

Else Yapılacak işlem

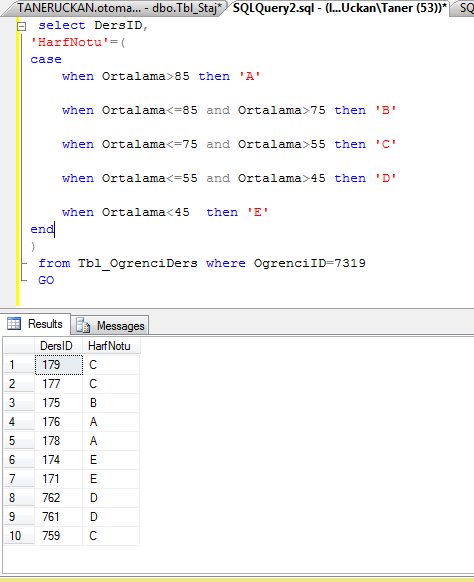
END

Kullanım şeklinden de anlaşılacağı üzere öncelikle case ifadesi ile kontrol edilecek değer belirtilmektedir. Bu değer belirtildikten sonra when ifadesi ile belirtilen değerin alacağı herhangib bir değer veya oluşturulacak bir koşul kontrol edilmektedir. When ifadesinde belirtilen değer veya koşul ifadesi sağlandığı zaman Then ifadesi ile istenilen atama işlemi veya istenilen başka bir işlem yapılabilmektedir.

Örnek: Öğrenci Tablosunda bulunan öğrencilerin cinsiyetlerinin kod olarak tutulduğu bir alanda kodların karşılığında cinsiyetlerinin ne olduğunu bulmamızı sağlayacak sql sorgusu aşağıdaki şekilde yapılmaktadır.



Örnek; Daha önce if-else ifadesinde örnek olarak yapmış olduğumuz öğrenci notları tablosundaki bilgilerin not aralığına göre harf notunun belirlenmesi işlemini Case ifadesi ile aşağıdaki şekilde yapmak mümkündür.



### 3.2.3 WHILE İfadesi

Bir Kod bloğunun belli bir koşul gerçekleştiği sürece aynı işlemi tekrar tekrar yapmasını istediğimiz zamanlarda döngü ifadeleri kullanılmaktadır. Sql’de döngü işlemleri çoğunlukla While ifadesi kullanılarak gerçekleştirilmektedir. Bu ifadede belirtilen koşul sağlandığı sürece yani TRUE değeri döndürdüğü sürece döngü blokları arasında bulunan kodlar tekrarlı bir şekilde çalışır. Kullanım şekli aşağıdaki şekildedir;

While İstenilen\_Şart\_ifadesi

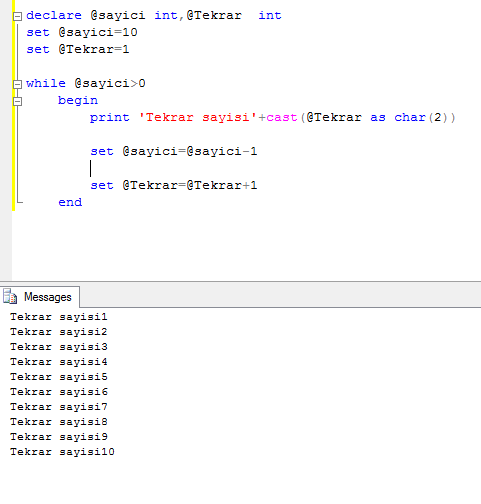
Begin

Döngü işlemine tabi tutulacak sql kodları

End

Begin .. End ifadesi çoğunlukla tekrarlanacak kod satırı sayısı birden fazla olduğu durumlarda kullanılmaktadır aksi durumda kullanılmaması bir hata doğurmayacağı gibi kullanılmasında da bir sakınca bulunmamaktadır.

Örnek; Bir döngünün nasıl işlediğini anlayabilmek için sayıcı mantığında çalışan bir program parçaçıcığı aşağıdaki şekilde yazılabilir.



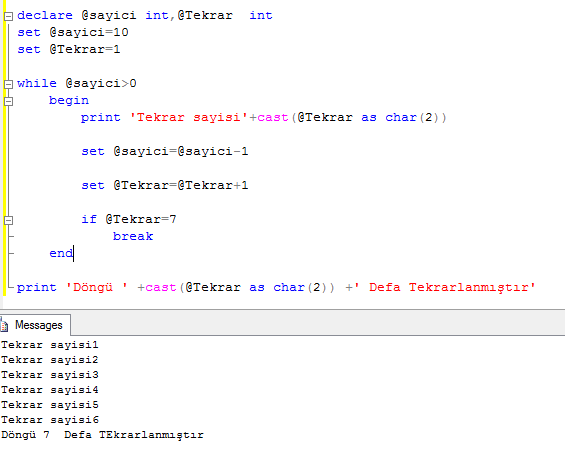
Yukarıdaki kod bloğunda görüldüğü üzere öncelikle döngünün kaç kez tekrarlanacağını belirleyen sayıcı değişkeni oluşturulmakta ve bu değere rastgele bir değer atılmaktadır. Döngünün koşul ifadesinde bu değişkenin değeri sıfır olana kadar tekrar etmesi istenmekte ve aynı zamanda Tekrar adındaki değişkenin değeride döngü sayısını belirlemek için adım adım arttırılıp print ifadesi ile ekrana yazılmıştır.

Sorgu bloğunda kullanılan Cast ifadesinin amacı tip dönüşümü yapmaktır. Daha önceden integer tipinde tanımlanan değişkenin bir string tipine dönüştürülmek istenmiştir.

### 3.2.4 Break İfadesi

Program içerisinde Döngü yapıları kullanılırken bu yapılar çoğunlukla döngü ifadesine verilen koşul sağlanana kadar tekrarlı olarak çalışmaktadır. Benzer şekilde bazı durumlarda da döngünün doğal çevrim sayısını tamamlamadan bitmesi ve programın döngü dışına çıkarak devam etmesi istenmektedir. İşte bu tip durumlarda Break ifadesi kullanılmaktadır. Break sayesinde döngü içerisinde istenilen değerlerler sağlandığı zaman döngüden çıkılabilmekte ve döngü bloğunun sonrasındaki ilk kod satırından program devam edebilmektedir. Aşağıdaki kod yapısında döngü devam ederken istenilen değere ulaşıldığında döngüden çıkılacaktır.

Örnek; Bir önceki While örneğinden yola çıkarak tekrar değişkeninin değerinin 7 olduğu duruma kadar döngünün devam etmesi istenmekte ve bu değer 7 değerini bulduğunda döngünden çıkıp programın devamındaki kodlara geçmesi sağlanmaktadır.

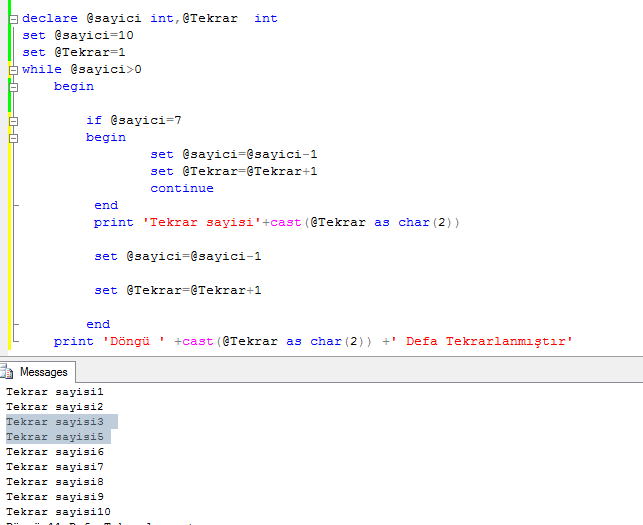


### 3.2.5 Continue İfadesi

Break ifadesine benzer bir kullanıma sahip olan bu ifade de döngü içerisinde belli bir noktada işlemlerin atlanması istenildiğinde kullanılmaktadır.Örneğin döngü devam ederken istenilen bir koşul sağladığında döngünün o adımdaki işleri atlaması ve diğer döngü adımına geçmesi istenebilir . Bu gibi durumlarda Continue ifadesi kullanılmaktadır.

Örnek; Break ifadesinde kullandığımız örneği Continue ifadesi için yeniden düzenleyebiliriz. Buradaki amacımız sayici değişkenimizin 7 değerine geldiğinde sayici ve tekrar değerlerinin içeriğinin arttırılmasını devam ettirmesi ve ekrana yazma işlemini atlamasıdır. Sonuç ekranındanda görüldüğü üzere 4.adımda yani sayici değeri 7 ye düştüğünde ekrana yazma işlemini atlamıştır.

Burada dikkat edilmesi gereken ve break komutundan farklı olan tarafı döngünün sonlanmaması ve bir sonraki adımdan rutin işlemlerini yapmaya devam etmesidir.Hatırlanacağı üzere break komutunda döngü tamamen sonlandırılıp döngü dışından program çalışmaya devam etmekteydi.



### 3.2.6 GOTO İfadesi

Programcılar tarafından kodlama esnasında bazı durumlarla karşılaşıldığında programın belirlenen başka bir noktadan devam edilmesi yani dallanması istenebilir. Bu durumlarda GOTO ifadesi kullanılmaktadır. Bu ifade ile daha önceden belirlenmiş bir etikete program içerisinden dallanma yapılabilmekte ve programın etiketin bulunduğu yerden çalışması sağlanabilmektedir. Goto ifadesinin kullanım şekli aşağıda gösterilmektedir.

Dallanılacak\_Bölge:

…

Goto Dallanılacak\_Bölge;

