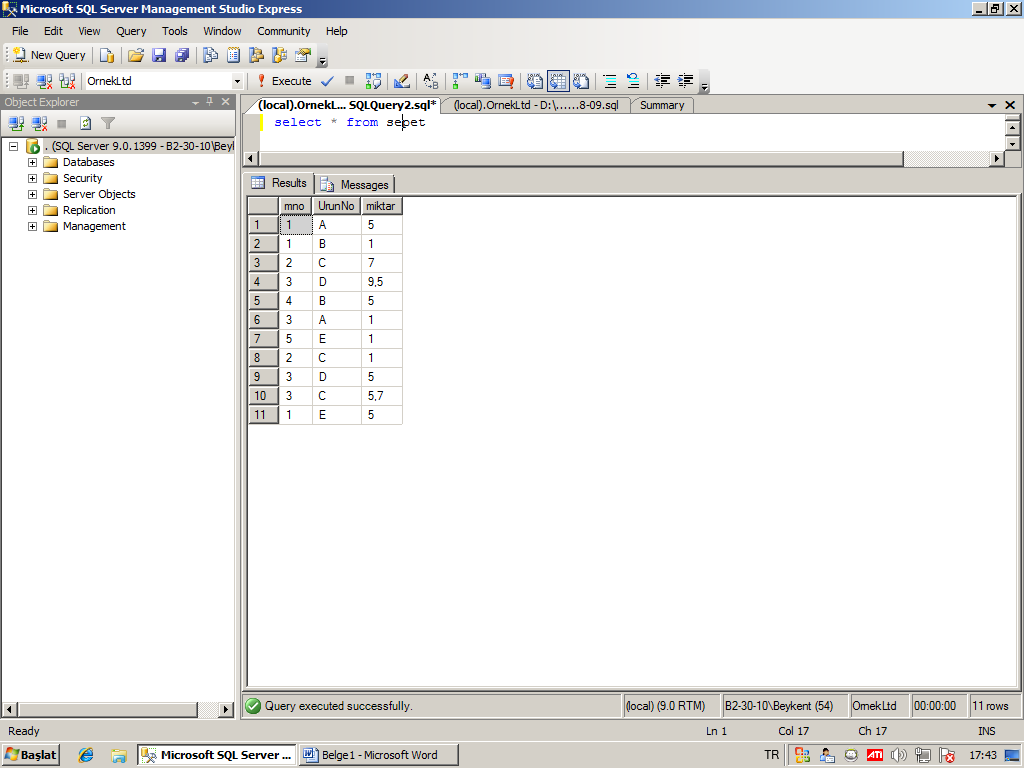
**Önceki Dersin Özeti**

Tablo Tipi Değişkenleri ve Bu değişkenlere veri girmeyi görmüştük.

Genel Yapısı:

Bu değişkenler tablo tipinde olduğu için, tabloya yapılan tüm işlemler, bu değişkenlere de yapılır. (insert, update ve delete işlemleri) Tablo gibi satır ve sütunlardan oluşur.

Declare @isim TABLE (kolon tanımlaması)



Sepet tablosuna benzeyen bir tablo tipi değişken tanımlayalım.

Declare @sepet TABLE (mno int, urunno varchar(10), miktar float)

Yukarıda @sepet isimli bir değişken tanımı yaptık. Bu değişkene veri girilebilir ve tablo gibi sorgulanabilir.

Select \* from @sepet

Yukarıda @sepet değişkeni tablo gibi sorgulandı.

Geçen derste bir tablodaki verinin başka bir tabloya aktarılmasını gördük. Bu amaçla insert into deyimini kullandık.

insert into ifadesine bakalım.

insert into tabloAdı (Kolon ismleri) select sorgusu

yukarıdaki ifadede, select sorgusundan dönem veri tabloya aktarılır. Eğer tablonun tüm kolonlarına veri yazılacaksa kolon isimlerini yazmaya gerek yoktur.

insert into tabloAdı select sorgusu

Yukarıda @sepet değişkenine, sepet tablosunun tümünü aktaralım.

-- @sepet değişkenini tanımla

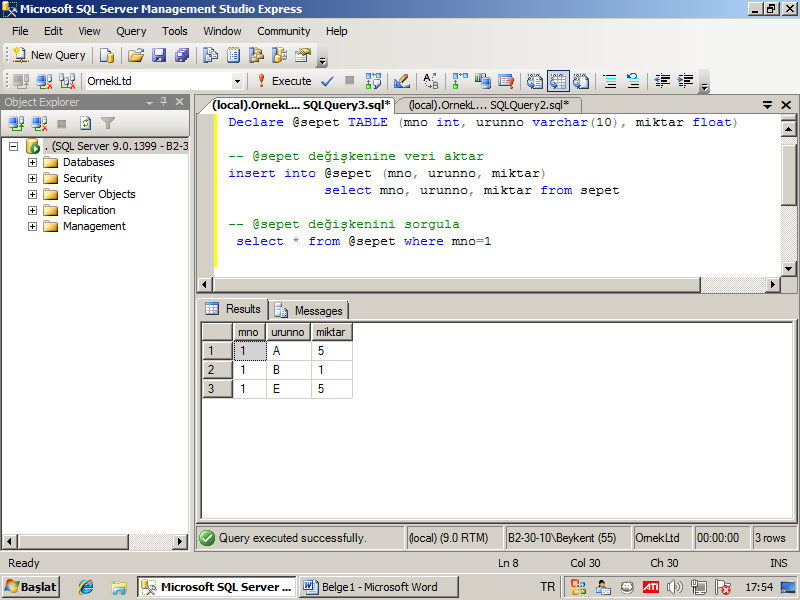
Declare @sepet TABLE (mno int, urunno varchar(10), miktar float)

-- @sepet değişkenine veri aktar

insert into @sepet (mno, urunno, miktar) select mno, urunno, miktar from sepet

-- @sepet değişkenini sorgula

select \* from @sepet where mno=1



KULLANICI TANIMLI FONKSİYONLAR

Biraz view’e, biraz da prosedüre benzerdir. İkisinin özelliklerini taşır.

Üç çeşit kullanıcı tanımlı fonksiyon vardır.

* Skaler değer (yani tek bir değer) veren fonksiyonlar
* Tablo döndüren fonksiyonlar, ki bunlar view’e benzerler
* Çok deyimli (multi-statement) tablo döndüren fonksiyonlar.

Fonksiyonları veritabanı programlamada nereye koyacağımızı anlamak biraz zor olabilir. Bu nedenle fonksiyonları view ve prosedürle karşılaştıralım.

\* Dışarıdan parametre alan bir view oluşturamayız. Bu türden ihtiyaçlar için kullanıcı tanımlı fonksiyonlar kullanılır.

\* Stored procedurleri bir sorgunun parçası olarak kullanamayız. Yani prosedürleri sorgulayamayız. Bir select deyiminin FROM tarafında prosedür olamaz.

\* Bir tek select ifadesi ile view oluşturamadığımız zaman kullanıcı tanımlı fonksiyonları kullanılır. (View’ler tekbir select ifadesinden oluşur.)

\* sql server’da bulunmayan, örneğin replace() gibi fonksiyonları tanımlamak için kullanıcı tanımlı fonksiyonlar kullanılır.

SKALER DEĞERLİ KULLANICI TANIMLI FONKSİYONLAR

Skaler fonksiyonlar, bir tek değer döndüren fonksiyonlardır. Örnek getdate() bir skaler fonksiyondur çünkü tek bir değer yani şu anın tarih ve saatini döndürür.

Bazı durumlarda buna benzer fonksiyonlara ihtiyaç duyulur. Örneğin bir müşterinin sepetinde kaç ürün bulunduğu, kaç sipariş verdiğini döndüren fonksiyonlar tanımlanabilir.

Genel İfade

CREATE FUNCTION fonkAdı (varsaParametreler)

RETURNS geriDönüşTipi

AS

Begin

Sql deyimi

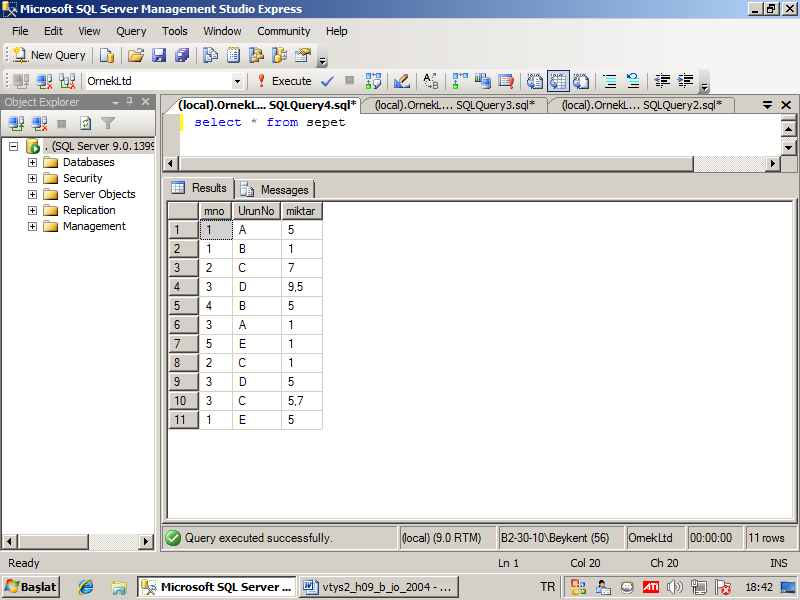
RETURN geriDönüşDeğeri

End

Örnek:

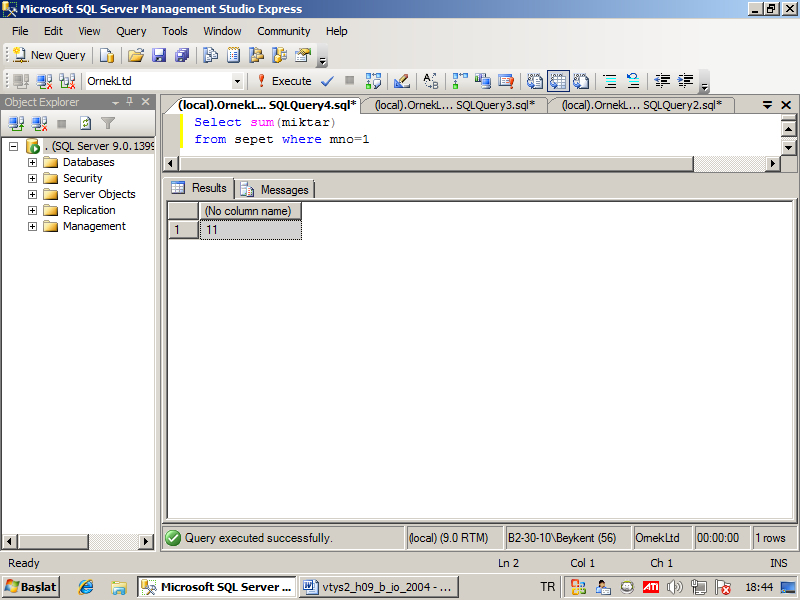
Bir müşteri no girildiğinde, bu müşterinin sepetinde toplam kaç ürün olduğunu bulan bir kullanıcı tanımlı fonksiyon oluşturunuz ve fonksiyonu çalıştırınız.

Çözüm:



Örneğin 1 nolu müşterinin sepetindeki ürün toplamını bulalım.

Select sum(miktar) from sepet where mno=1



Yukarıdaki sorguyu fonksiyon içine yerleştirelim.

Create function fn\_urun\_sayi (@mno int)

Returns float

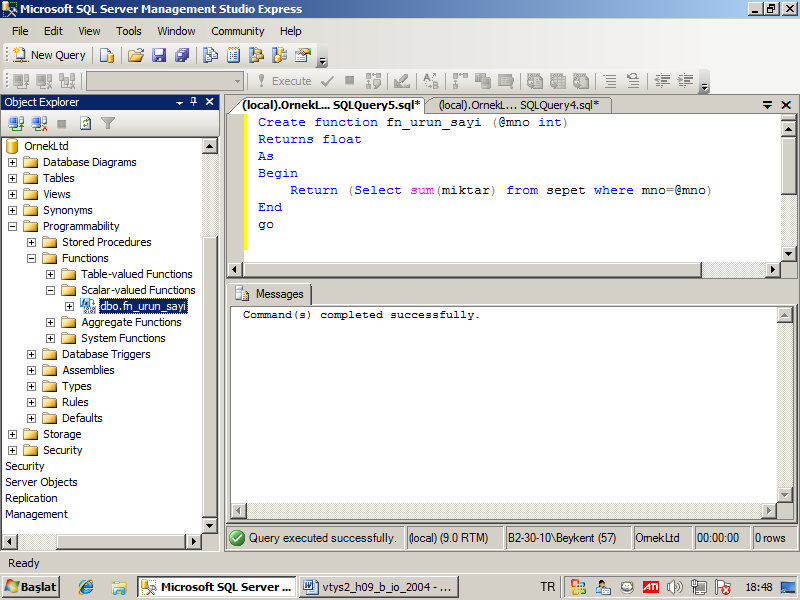
As

Begin

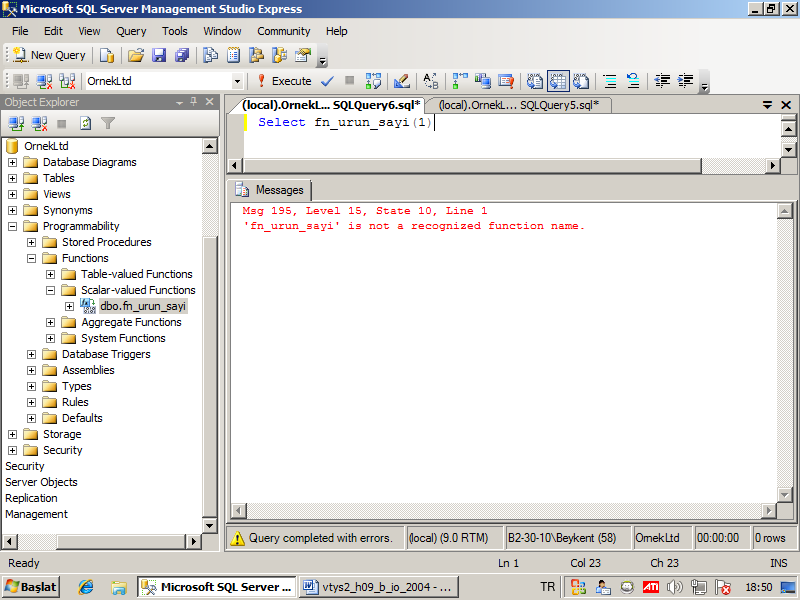
Return (Select sum(miktar) from sepet where mno=@mno)

End

go



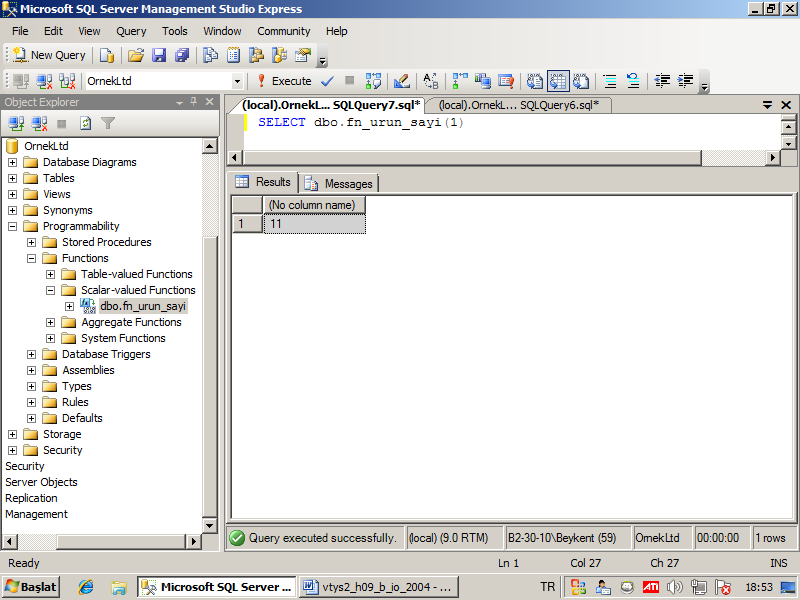
Select fn\_urun\_sayi(1)

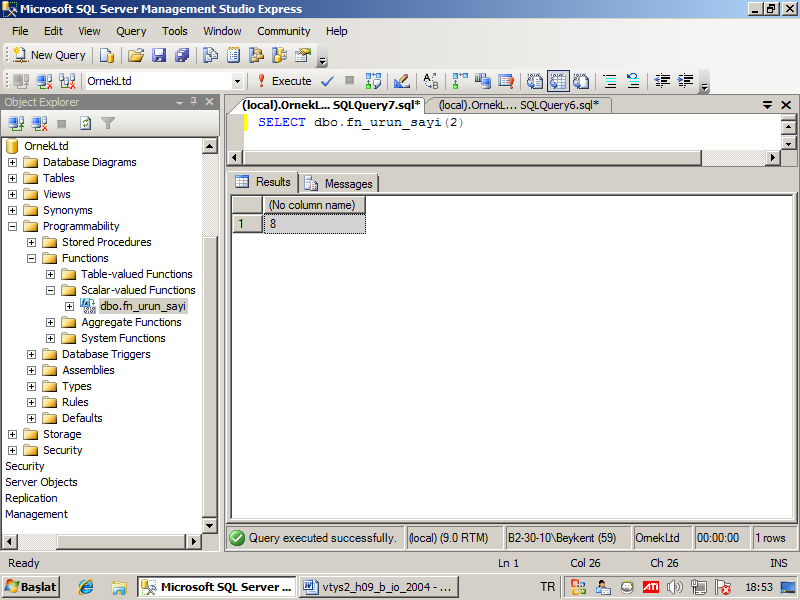


Skaler değerli fonksiyon isminin başında şema adı yazılı olmalı. Yazılı olmayınca kendi standart arıyor. Böyle bir fonksiyon olmadığı mesajını veriyor. Aşağıdaki kod çalışır.

Varsayılan şema adı dbo ‘dur.

SELECT dbo.fn\_urun\_sayi(1)

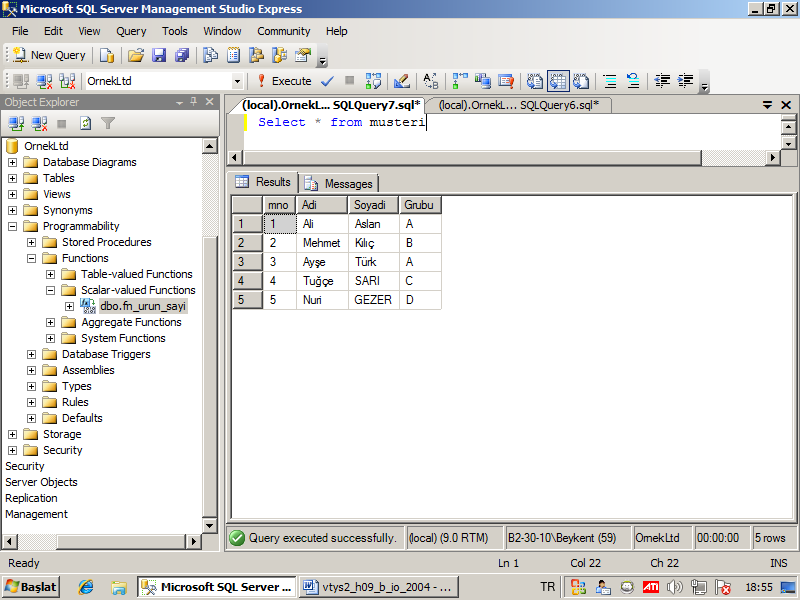




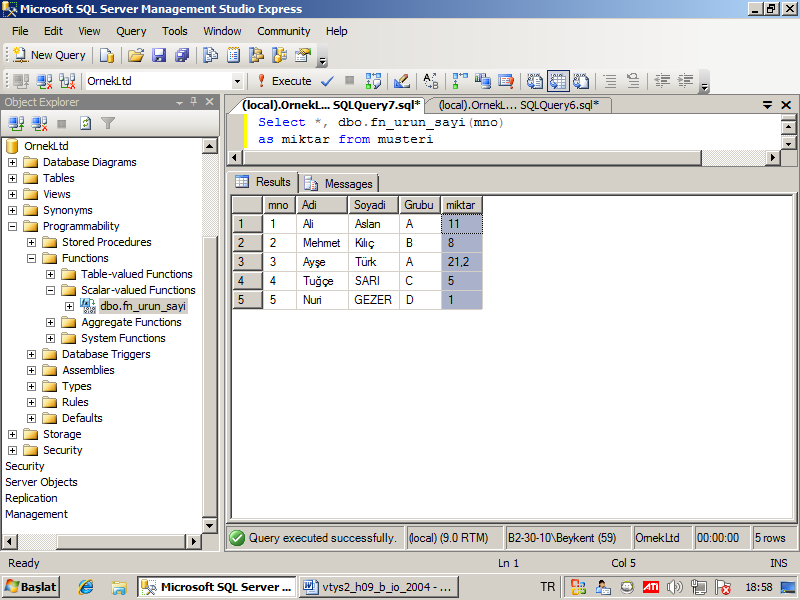
Örnek:

Yukarıdaki fonksiyonu kullanarak, müşterileri ve sepetindeki ürün miktarını veren sorguyu yazınız.

Select \* from musteri



Select \*, dbo.fn\_urun\_sayi(mno) as miktar from musteri



Örnek:

1,2 gibi ay numarası girilecek, Ocak, şubat gibi ay adını döndüren bir kullanıcı tanımlı fonksiyon oluşturunuz.

Create function fn\_ay (@ay tinyint)

Returns varchar(30)

As

Begin

Return (

Select case @ay

When 1 then ‘Ocak’

When 2 then ‘Şubat’

When 3 then ‘Mart’

When 4 then ‘Nisan’

When 5 then ‘Mayıs’

When 6 then ‘Haziran’

When 7 then ‘Temmuz’

When 8 then ‘Ağustos’

When 9 then ‘Eylül’

When 10 then ‘Ekim’

When 11 then ‘Kasım’

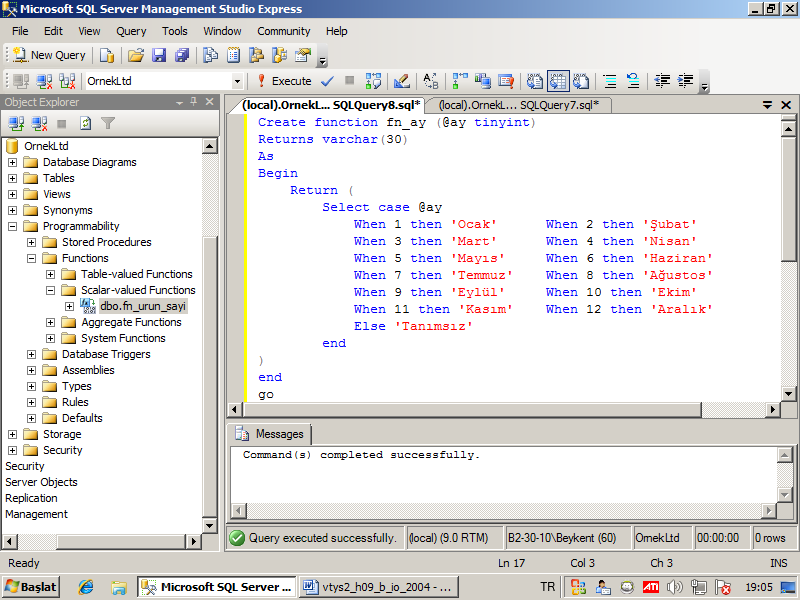
When 12 then ‘Aralık’

Else ‘Tanımsız’

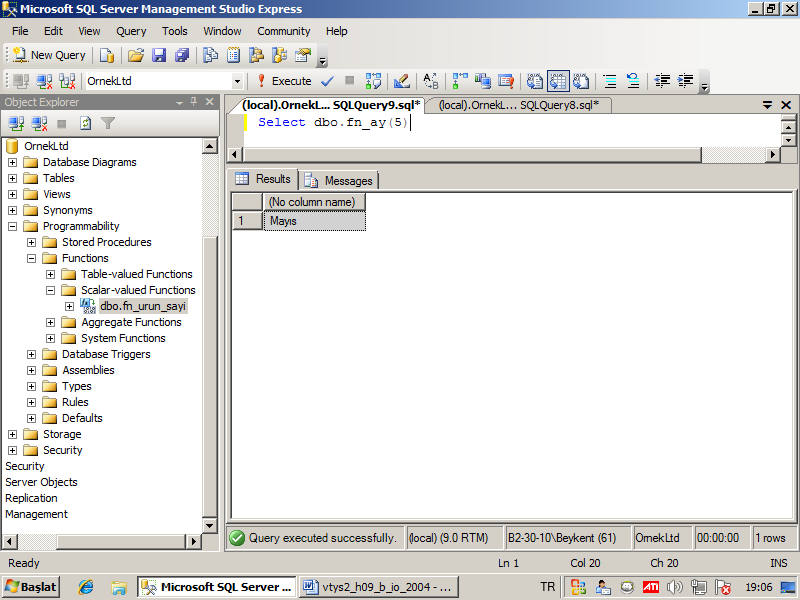
end

)

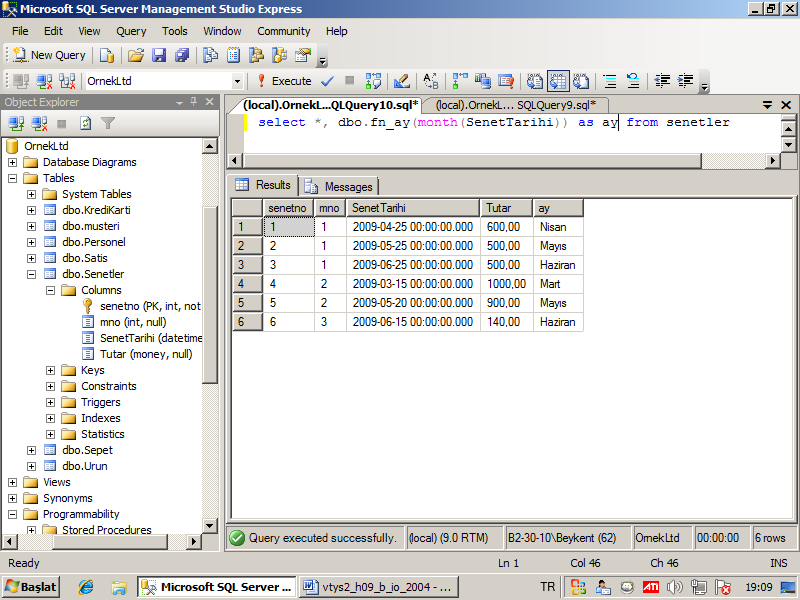
End



Select dbo.fn\_ay(5)







Örnek:

Herhangi bir gruba ait müşteri sayısını veren sorguyu yazınız.

Create function fn\_musteri\_sayisi

(@grup varchar(10))

Returns int

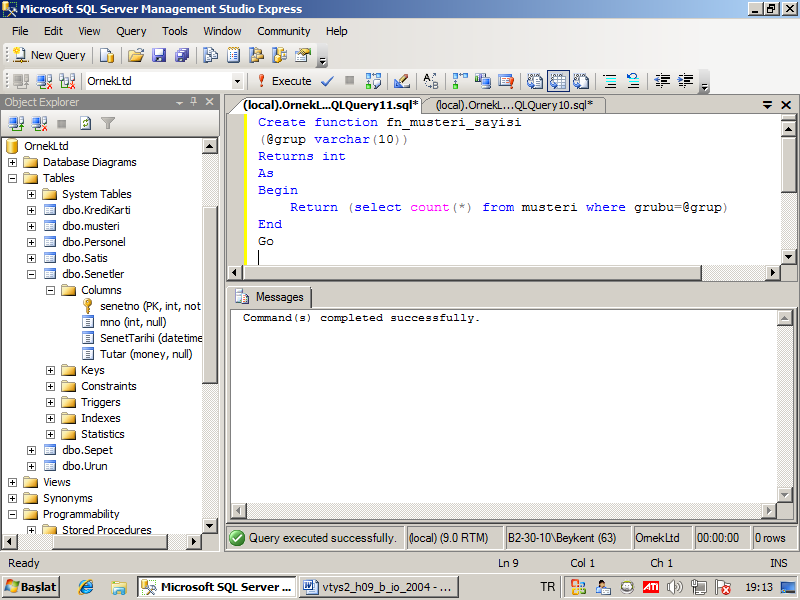
As

Begin

Return (select count(\*) from musteri where grubu=@grup)

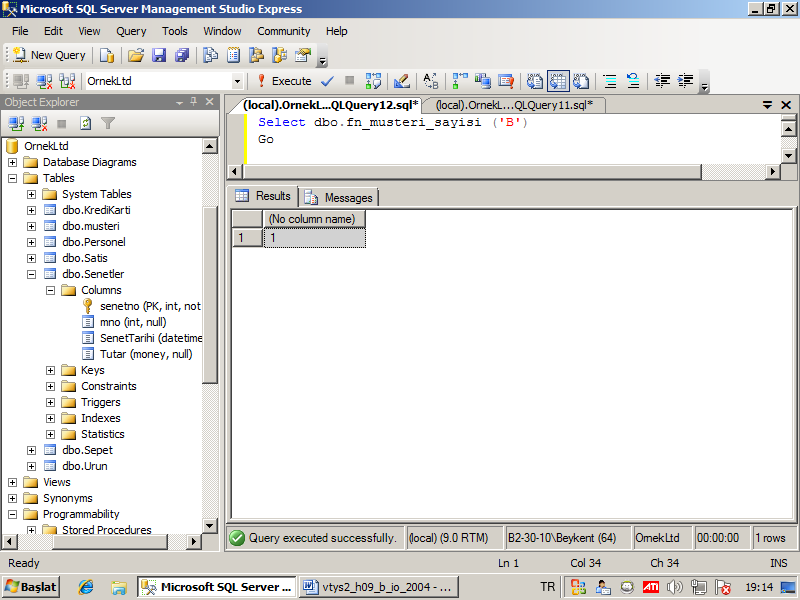
End

Go



Select dbo.fn\_musteri\_sayisi (‘B’)

Go



Yukarıda B grubundaki müşteri sayısını bulduk.

Müşteri grubu parametre değeri NULL ise tüm müşteri sayısını, değilse ilgili gruptaki müşteri sayısını veren sorguyu yazınız.

alter function fn\_musteri\_sayisi

(@grup varchar(10))

Returns int

As

Begin

Declare @sayi int

IF @grup is null

Select @sayi = (select count(\*) from musteri)

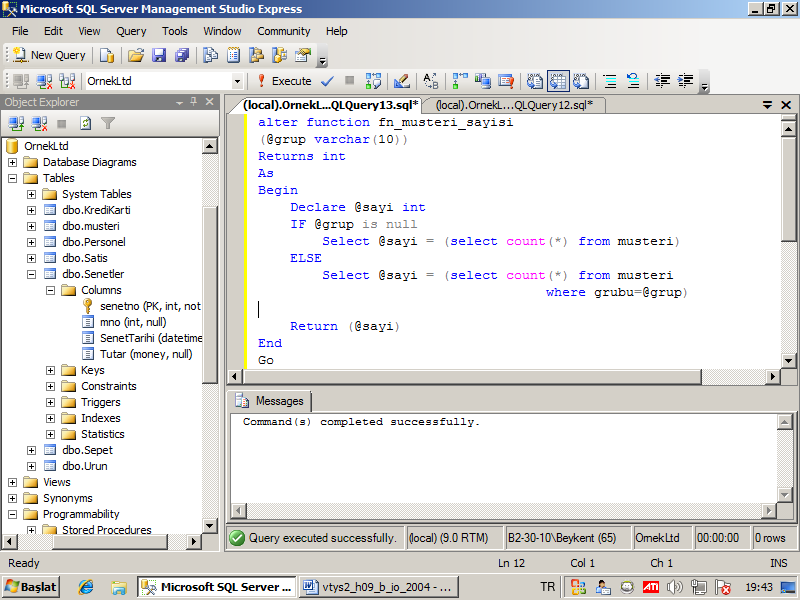
ELSE

Select @sayi = (select count(\*) from musteri where grubu=@grup)

Return (@sayi)

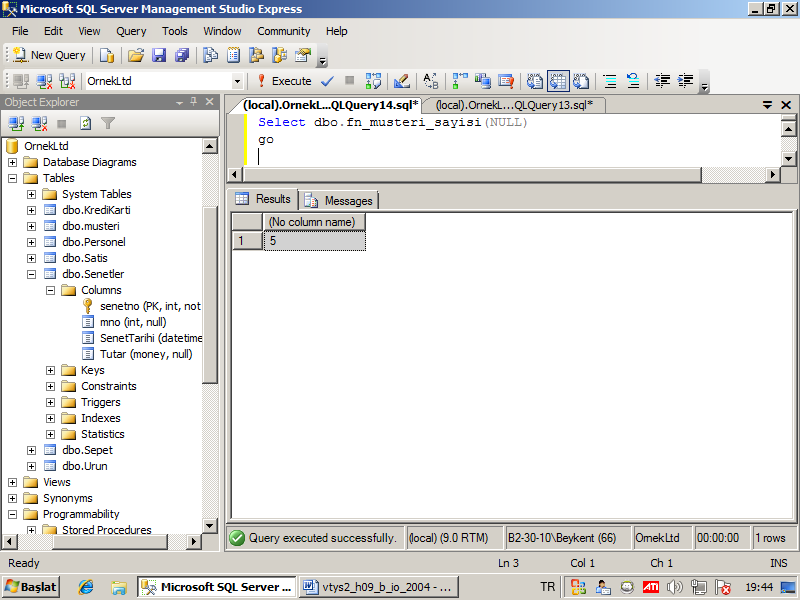
End

Go



Select dbo.fn\_musteri\_sayisi(NULL)

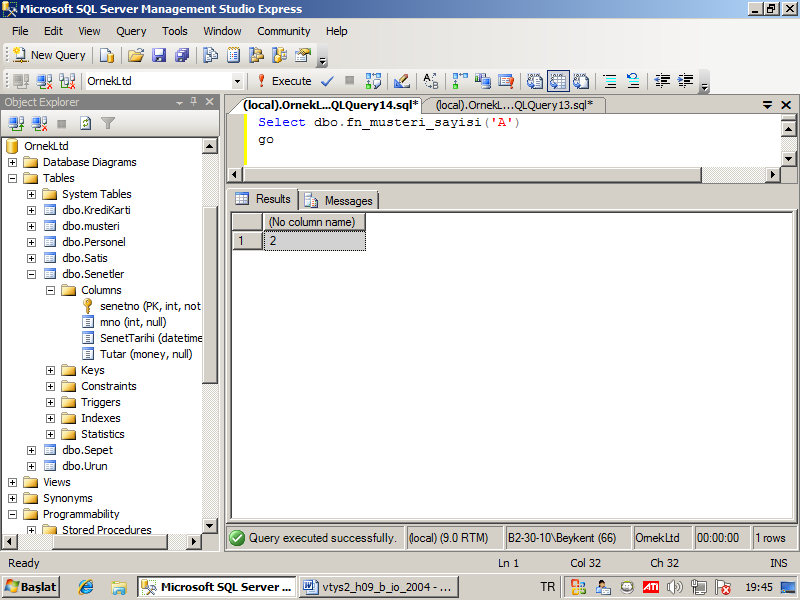
go



Yukarıda parametrede NULL girdik, bu bize tüm müşteri sayını verir.

Select dbo.fn\_musteri\_sayisi(‘A’)

go



Yukarıdaki kod bize A grubundaki müşteri sayısını verir.

TABLO DÖNDÜREN FONKSİYONLAR

Bu tip fonksiyonlar viewlere çok benzerdir. Bir tek select ifadesi içerir. Bu ifadenin sonucunu gösterir. View’den farklı olarak dışarıdan parametre alır.

View gibi tek bir select yazılabilir.

View gibi sorgulanabilir.

Genel İfade

CREATE FUNCTION fonkAdı (varsaParametreler)

RETURNS TABLE

AS

RETURN selectİfadesi

Go

View, fonksiyon ve prosedürleri karşılaştırmak için kod yazalım ve çalıştıralım.

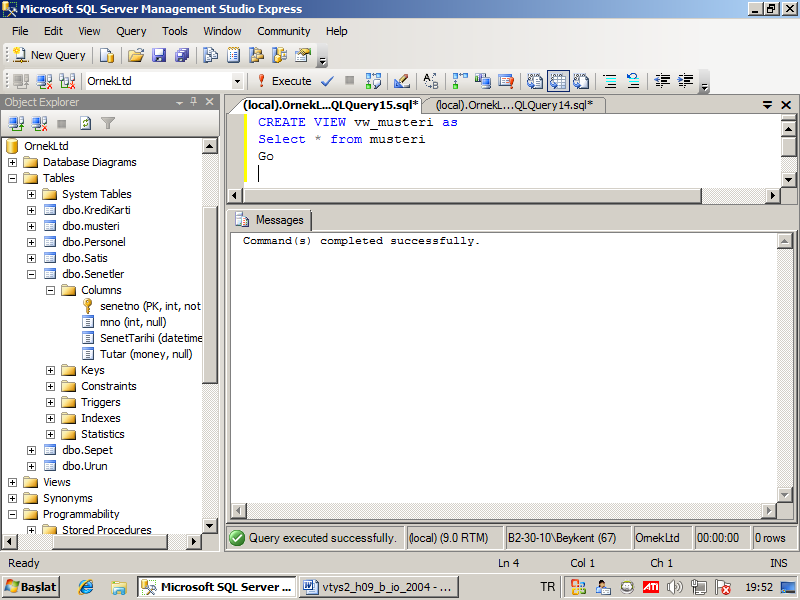
Örnek:

Tüm müşterileri listeleyen bir view oluşturunuz ve bu viewi sorgulayınız.

CREATE VIEW vw\_musteri as

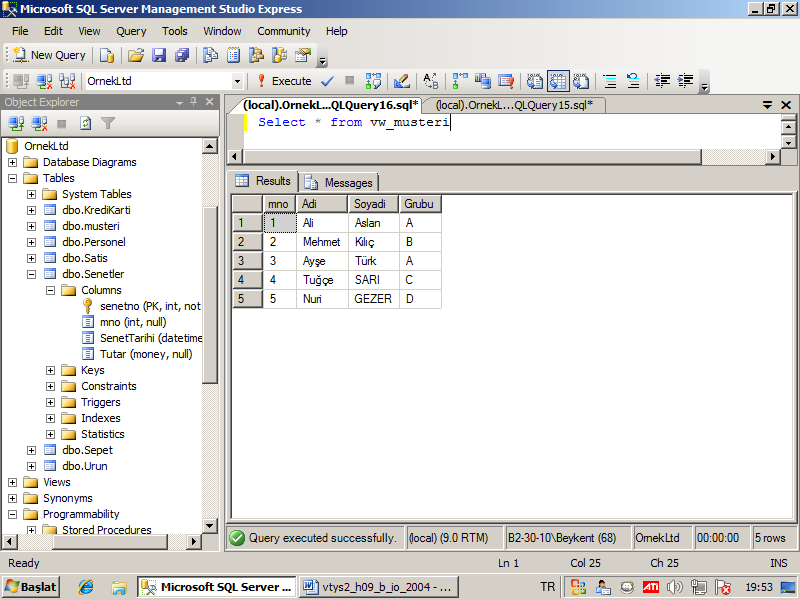
Select \* from musteri

Go



Yukarıdaki view’i sorgulayalım.

Select \* from vw\_musteri



Aynı işi yapan bir fonksiyon oluşturalım.

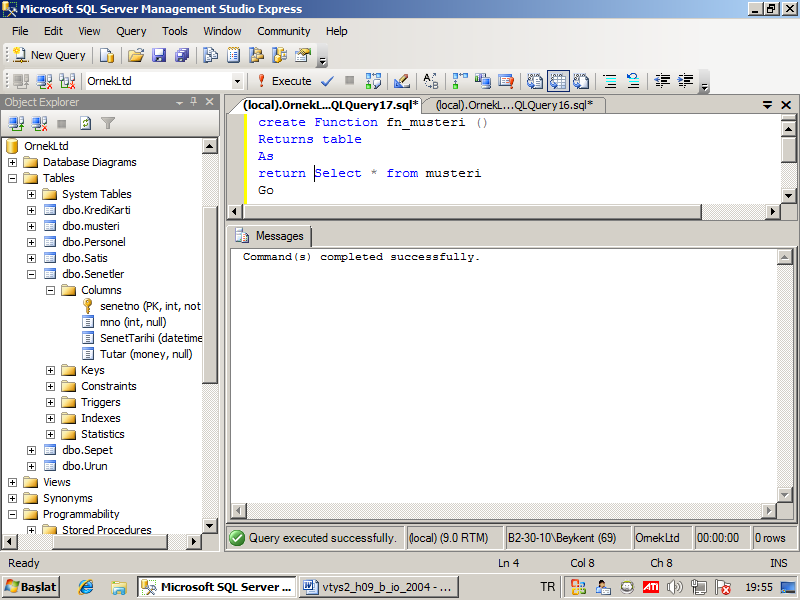
Create Function fn\_musteri ()

Returns table

As

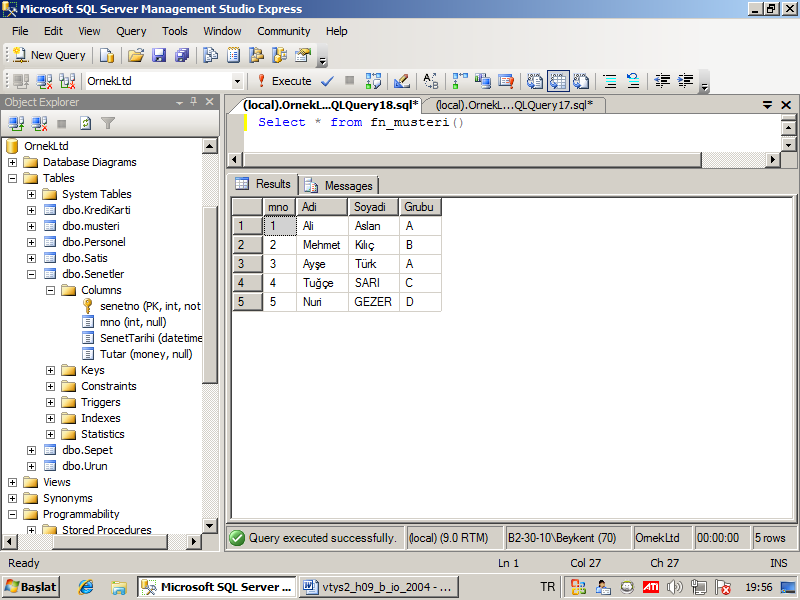
Return Select \* from musteri

Go



Yukarıdaki fonksiyonu sorgulayalım.

Select \* from fn\_musteri()



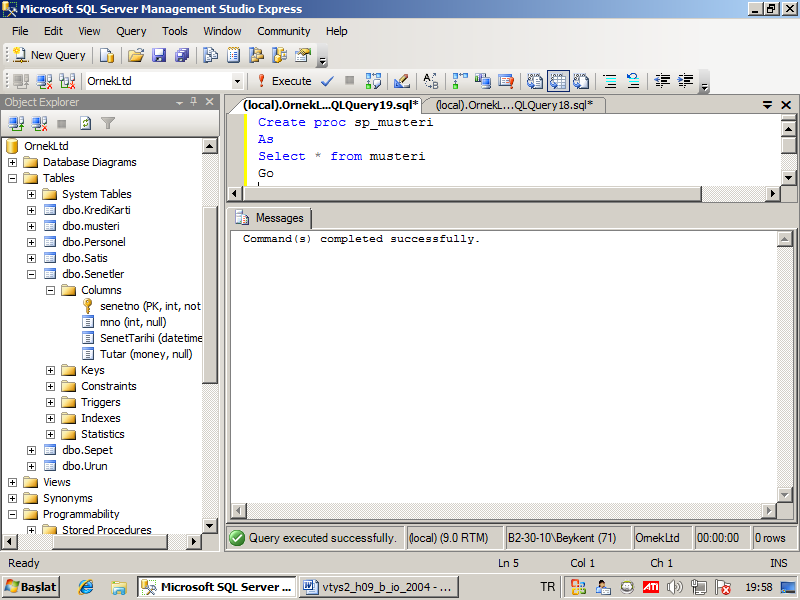
Aynı işi yapan prosedür oluşturalım

Create proc sp\_musteri

As

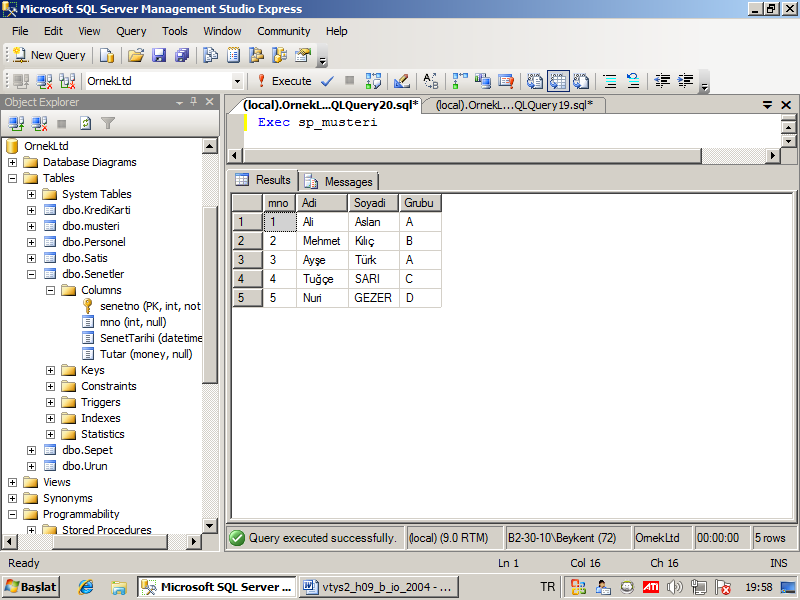
Select \* from musteri

Go



Yukarıdaki prosedürü çalıştıralım.

Exec sp\_musteri

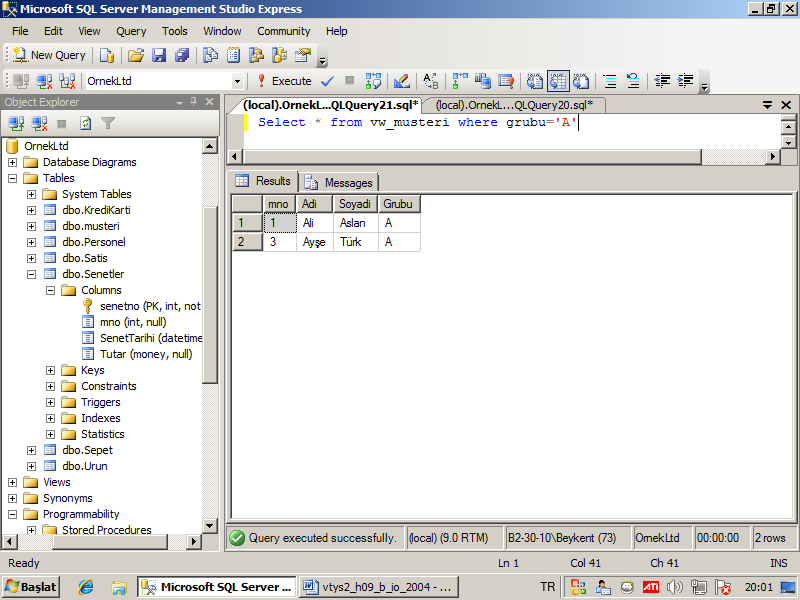


View, fonksiyon ve prosedürün yukarıdaki çıktılarını karşılaştırırsak aynı olduğunu görürüz.

Yukarıdaki fonksiyon, view ve prosedürden faydalanarak A grubundaki müşterileri listeleyiniz.

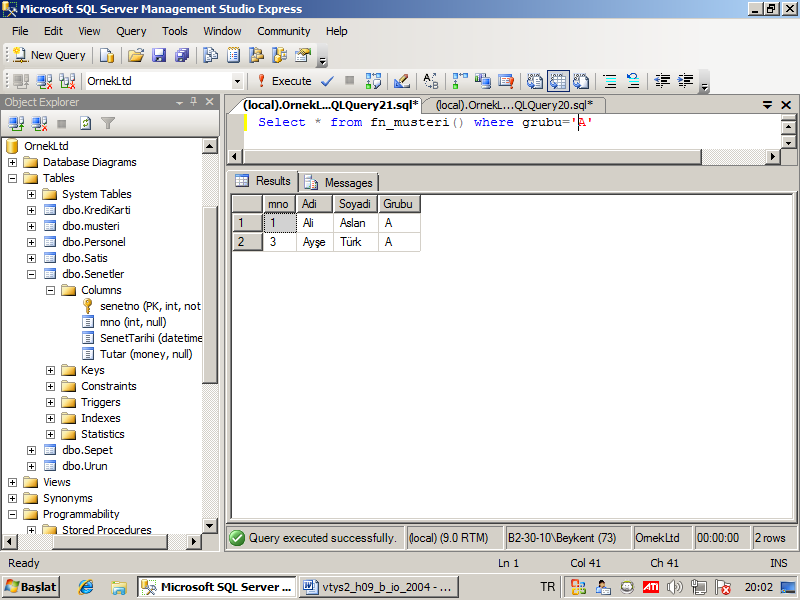
-- view’den faydanalanarak A grubundaki listeleyelim.

Select \* from vw\_musteri where grubu=’A’



Aynı işi fonksiyon ile yapalım

Select \* from fn\_musteri() where grubu=’A’



Aynı işi prosedür ile yapalım.

Prosedürler sorgulamaz. Sp\_musteri prosedüründe yararlanarak A grubundaki bulamayız.

View ve fonksiyon sorgulanabilir ancak prosedür sorgulanamaz.

Şimdi view, fonksiyon ve prosedürlere parametre tanımlayalım. (Grubu parametresi)

-- view parametre tanımlayalım.

View lere parametre tanımlanamaz.

-- fonksiyona parametre tanımlayalım.

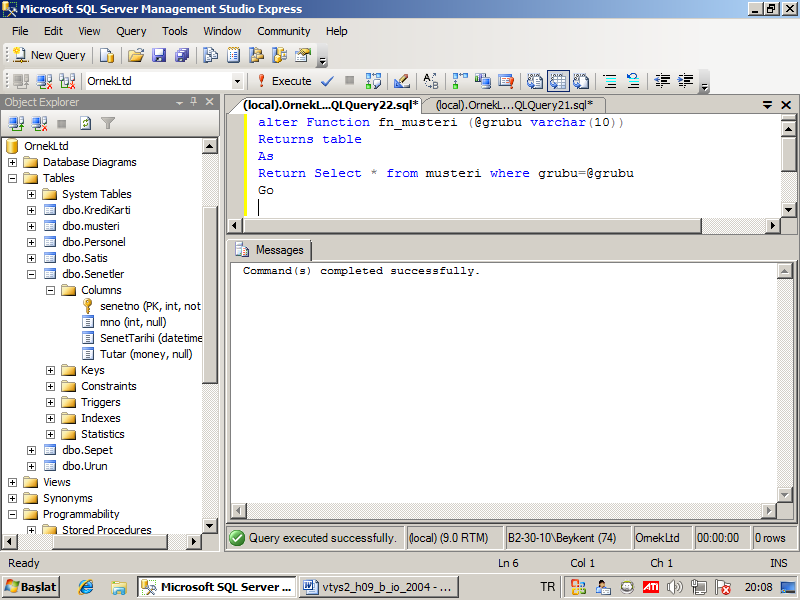
alter Function fn\_musteri (@grubu varchar(10))

Returns table

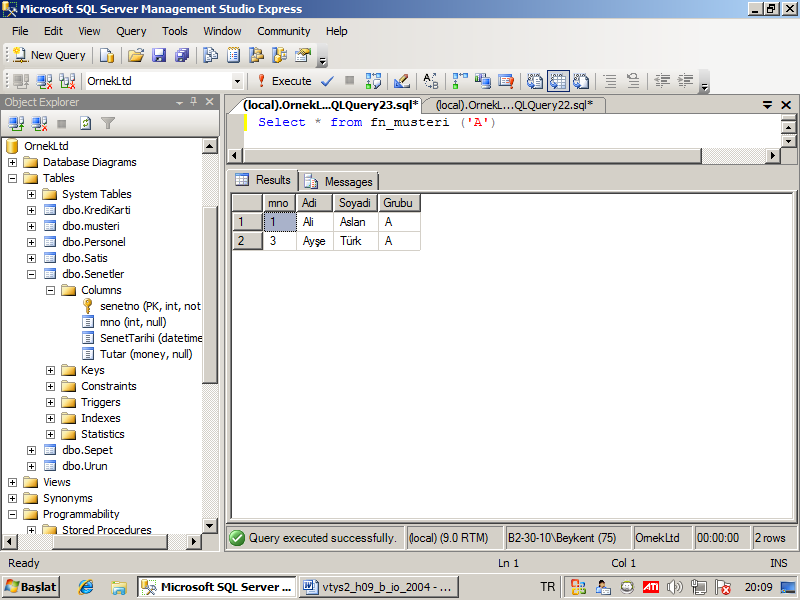
As

Return Select \* from musteri where grubu=@grubu

Go



Select \* from fn\_musteri (‘A’)



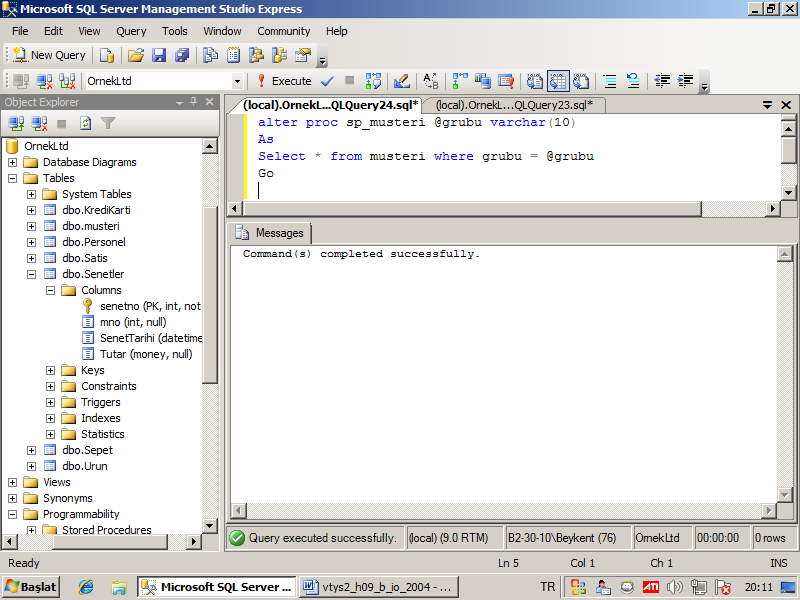
--prosedüre parametre tanımlayalım.

alter proc sp\_musteri @grubu varchar(10)

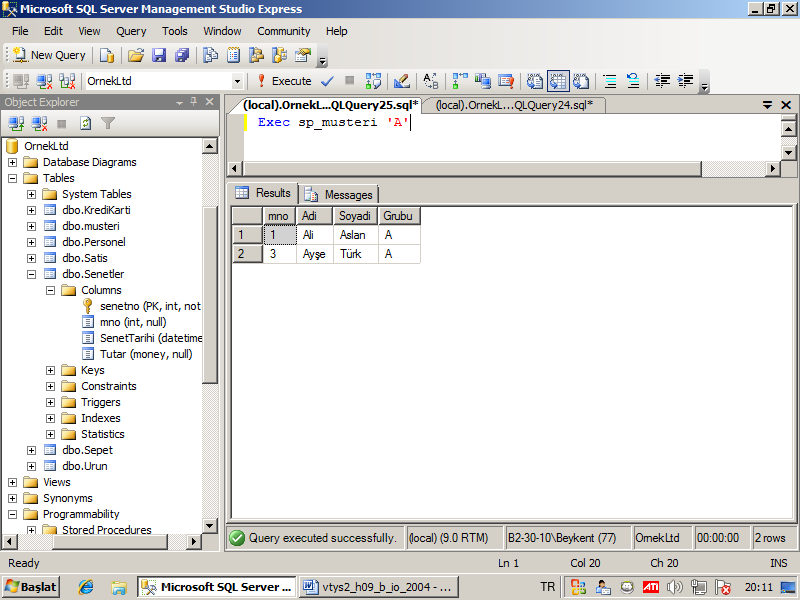
As

Select \* from musteri where grubu = @grubu

Go



Exec sp\_musteri ‘A’



Yukarıdakileri incelersek view’e parametre tanımlanamaz ancak fonksiyon ve prosedüre tanımlanabilir.