



# VERİTABANI YÖNETİM SİSTEMLERİ

Dr. Öğr. üyesi  
Ender Şahinaslan

# BÖLÜM -9-

---

## T-SQL & View



# GENEL BAKIŞ...

---

## T-SQL

- **Değişken Türleri,**
- **Değişken Tanımlama ve Kuralları,**
- **Değişkenlere Değer Atama,**
- **T-SQL ile Yığın İşlemleri,**
- **T-SQL'de Kontrol Deyimleri,**
  - **If-Else** Deyimi,
  - **Case** Deyimi,
  - **While** Döngüsü,
  - **Break** Deyimi,
  - **Continue** Deyimi.

# GENEL BAKIŞ... (DEVAM...)

---

## **View (Görünüm) Nesnesi**

- View oluşturmak,
- View sütunlarına isim vermek,
- Birden fazla tablo kullanmak,
- View'larda değişiklik yapmak,
- View'ları silmek,
- View'ları şifrelemek.

## 9.1. T-SQL KAVRAMI

---

Microsoft veritabanı sorgulama dilinin gelişmiş bir versiyonudur.

**T-SQL:** Transact - Structured Query Language kelimelerinin kısaltmasıdır.

**SQL Server içerisinde bulunmayan döngü vb. işlemler** T-SQL aracılığı ile yapılabilmektedir.

**Bu işlemleri yapmak için herhangi bir derleyiciye** ihtiyaç yoktur.

**T-SQL ile bütün SQL komutlarını kullanabilir.**

## 9.1.1. DEĞİŞKEN TÜRLERİ

---

**Değişkenler**, verilerin bellekte geçici olarak tutulmalarını ve gerektiğinde kullanılmasını sağlayan yapılardır.

**T-SQL**, diğer programlama dillerinde olduğu gibi değişken tanımlamaya olanak sağlamaktadır.

**T-SQL'de tanımlanan değişken** diğer programlama dillerinde olduğu gibi **bir veri tipi ile sınırlandırılmıştır** ve oluşturulmasının ardından **bellekte belli bir yer** kaplamaktadır.

- Üzerine veri yazılmasına izin vermekte ve daha sonrada program içerisinden çağrılabilmektedir.

## 9.1.1. DEĞİŞKEN TÜRLERİ (DEVAM...)

SQL server'da değişkenler ikiye ayrılmaktadır:

- Yerel değişkenler
- Global değişkenler

Yerel değişkenler, önlerine **@** işareti alarak tanımlanırlar.

- Örneğin; **@Deger**

Global değişkenler ise önlerine **@@** işareti alırlar. Bu değişkenler **SQL Server** tarafından tanımlanmıştır ve kullanıcılar tarafından tanımlanamazlar.

- Örneğin; **@@CONNECTION**

## 9.1.2. DEĞİŞKEN TANIMLAMA KURALLARI

---

Değişken tanımlamaları yapılırken her dilde olduğu gibi T-SQL'de de uyulması gereken kurallar vardır. Bu kurallar şunlardır:

- Değişkenler harf ile başlamalıdır.
- Boşluk karakteri veya Türkçe karakterler kullanılmamalıdır.
- Değişkenler SQL Server için özel anlamı olan ifadeler ile başlamamalıdır. (örneğin @, @@, #, \$ gibi)
- SQL komutları değişkenlere isim olarak verilmemelidir.
  - Örneğin INSERT, UPDATE v.b.



## 9.1.2. DEĞİŞKEN TANIMLAMA KURALLARI (DEVAM...)

---

Nesne isimleri kısa ve anlaşılır olmalıdır.

Tablolara isim verilirken daha kısa ve anlamlı isimler tercih edilmelidir.

- Örneğin TabloUrun yerine tblUrun tercih edilebilir.

Tablolardaki NULL ifadesi hiçbir şey girilmemiş anlamına gelmektedir. Klavyeden SPACE tuşuna basılarak bırakılan boşluk ifadesi ile aynı anlamda değildir.

- NULL ifadesi daha önceden değeri bilinmeyen ve bu nedenle boş bırakılan ifadeler için tercih edilmelidir.

### 9.1.3. DEĞİŞKEN TANIMLAMA

---

T-SQL'de bir değişken tanımlamak için **DECLARE** komutu kullanılmaktadır. DECLARE komutunun genel biçimi aşağıdaki gibidir.

**DECLARE @Degiskenin\_Adi Veri\_Tipi**

Örnekler:

**DECLARE @Adi varchar (20)**

**DECLARE @Numarasi int**

**DECLARE @Tarih smalldatetime**

### 9.1.3. DEĞİŞKEN TANIMLAMA

---

**T-SQL** de araya virgül koyarak birden fazla değişken tanımlamak mümkündür. Böyle bir durumda sadece tek bir **DECLARE** komutu kullanılmış olmaktadır.

#### Örnek:

```
DECLARE @Adi varchar(20),  
          @Numarasi int,  
          @Tarih smalldatetime
```

## 9.1.4. DEĞİŞKENLERE DEĞER ATAMA

---

- T-SQL'de değişkenlere değer atamanın 2  
yolu vardır:
  - **SET** komutu ile değer atama,
  - **SELECT** komutu ile değer atama.
- Bu iki kullanıma ait örnekler ve genel kullanımları sonraki slaytlarda verilmiştir.

## 9.1.4. DEĞİŞKENLERE DEĞER ATAMA (DEVAM...)

---

**SET** komutu kullanılarak değer atamanın genel ifadesi aşağıdaki gibidir:

**SET @degiskenin\_adi = atanan\_deger**

### Örnekler:

- **SET @Adi = 'MURAT'**
- **SET @Soyad = 'ŞAHİN'**
- **SET @Numarasi = 23403016**
- **SET @Tarih = '25.04.2024'**

## 9.1.4. DEĞİŞKENLERE DEĞER ATAMA (DEVAM...)

---

**SELECT** komutu ile de değişkenlere değer atanabilmektedir. Eğer tek başına değişken ile kullanılırsa değişkenin içerisindeki değer ekran görüntülenmesini de sağlamaktadır.

### Örnek:

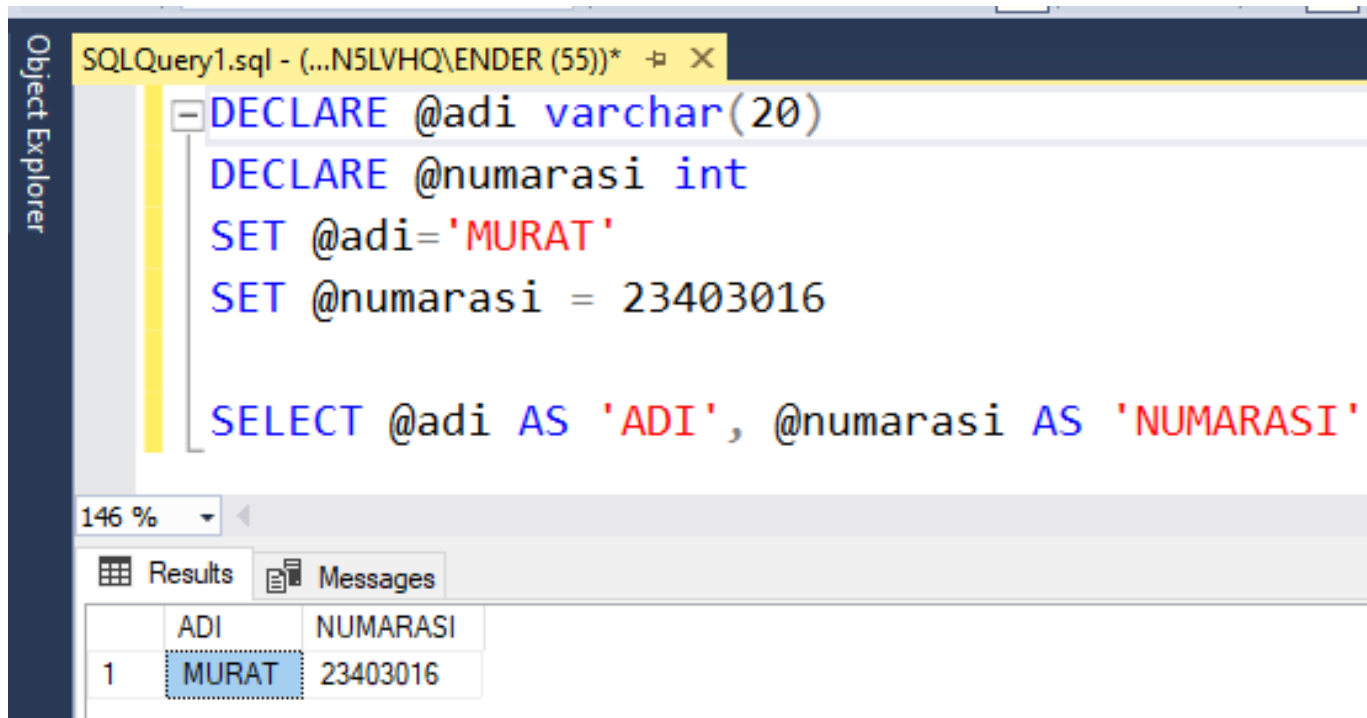
- **DECLARE @Adi varchar (20)**

**SELECT @Adi = 'Mehmet'**

Böyle bir kullanımda değişkene değer atanabilir fakat ekranda gösterilmemektedir.

## 9.1.4. DEĞİŞKENLERE DEĞER ATAMA (DEVAM...)

- **SELECT** komutu ile ekranda değişkenlerin değerleri gösterilmek istenirse;
- Örnek:



The screenshot shows a SQL query window titled "SQLQuery1.sql - (...N5LVHQ\ENDER (55))\*". The query contains the following T-SQL code:

```
DECLARE @adi varchar(20)
DECLARE @numarasi int
SET @adi='MURAT'
SET @numarasi = 23403016

SELECT @adi AS 'ADI', @numarasi AS 'NUMARASI'
```

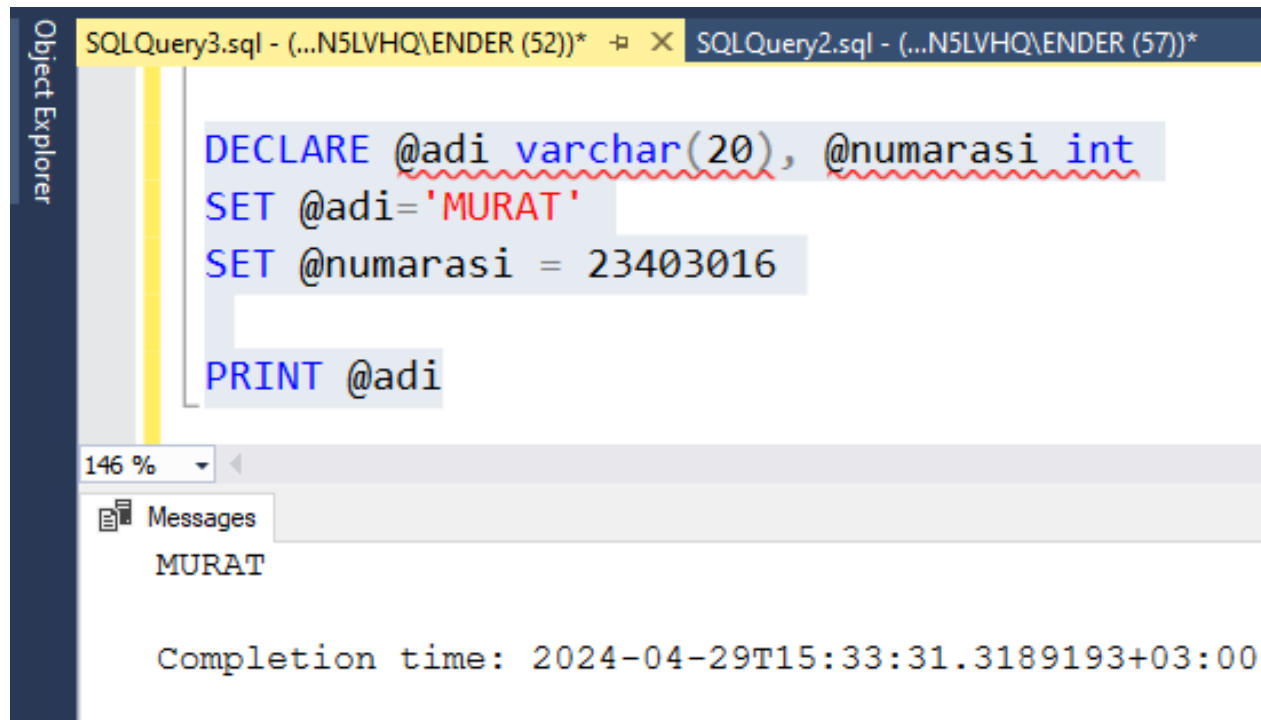
Below the query window, the "Results" tab is active, displaying a grid with the following data:

	ADI	NUMARASI
1	MURAT	23403016

## 9.1.4. DEĞİŞKENLERE DEĞER ATAMA (DEVAM...)

- SELECT dışında ekrana bilgileri yazmaya yarayan bir diğer T-SQL komutu da **PRINT** komutudur. PRINT komutu, değişkenlerin değerlerini, ortaya çıkan hataları vb. durumları ekrana yazdırmak için kullanılmaktadır.

- Örnek:



The screenshot shows a SQL Server Enterprise Manager interface. On the left is the 'Object Explorer' pane. The main window displays a query titled 'SQLQuery3.sql - (...N5LVHQ\ENDER (52))\*'. The query text is as follows:

```
DECLARE @adi varchar(20), @numarasi int
SET @adi='MURAT'
SET @numarasi = 23403016
PRINT @adi
```

Below the query text, the 'Messages' pane shows the output of the PRINT statement:

```
MURAT
```

At the bottom, the 'Completion time' is displayed as: 2024-04-29T15:33:31.3189193+03:00.



## 9.1.5. T-SQL İLE YIĞIN İŞLEMLERİ

---

**Yığın**, belli sorguların arka arkaya yapılması anlamına gelmektedir.

**SQL Server**, arka arkaya gelen sorguları yığın olarak ele almaktadır.

Herhangi bir yığının sonunu belirlemek için **GO** komutu kullanılmaktadır.

SQL Server'da bir yığın çalışmaya başlamadan önce ayrıştırılır (parse edilir). Daha sonra ayrıştırılan bu sorgular derlenerek çalıştırılır.

## 9.1.5. T-SQL İLE YIĞIN İŞLEMLERİ (DEVAM...)

Genel kullanım biçimi şöyledir:

Komutlar

GO

Komutlar

GO

....

SQLQuery1.sql - (...N5LVHQ\ENDER (55))\*

```
USE dukkan
GO
SELECT * FROM tblUrun
GO
SELECT * FROM tblMarka
```

146 %

Results Messages

	urunKod	Barkod	bolgeKod	markaKod	urunAd	bayiFiyat	listeFiyat	Indirim	KDVoran	un
1	3505	NULL	NULL	NULL	Staj Defteri	NULL	2700.0000	NULL	NULL	N
2	3391	3CP015098	NULL	1	HomeConnect PC Digital Web Kamera	20.0000	28.1697	0.0000	0.1800	N
3	3026	KBS-720-F	NULL	2	PS/2 F TR Klavye (Anti-RSI)	4.3000	6.0461	0.0000	0.1800	N
4	3027	KBS-720	NULL	2	PS/2 Q TR Anti-RSI Standart Klavye	4.5000	6.3565	0.0000	0.1800	N
5	3028	KBS-21	NULL	2	PS/2 Q Anti-Rsi MM Klavye (Siyah/Gri)	7.5000	10.5776	0.0000	0.1800	N
6	3029	KBS-8	NULL	2	PS/2 Q TR Multimedya Klavye (Anti-RSI)	7.5000	10.5776	0.0000	0.1800	N
7	3030	KBS-827	NULL	2	PS/2 Q TR Multimedya Klavye + Kablosuz Mouse (An...	23.0000	32.3908	0.0000	0.1800	N
8	3031	KB-827	NULL	2	PS/2 Q TR Multimedya Klavye + Kablosuz Mouse	23.0000	32.3908	0.0000	0.1800	N

	markaKod	Marka	MarkaWeb	MarkaMail	MarkaTel	MarkaResim
1	1	3COM	NULL			
2	2	A4 TECH	NULL			
3	3	ADAPTEC	NULL			
4	4	AIRFORCE	NULL			
5	5	ALCATEL	NULL			
6	6	ALPS	NULL			
7	7	ALTEC	NULL			
8	8	AMP	NULL			

Query executed successfully.

## 9.1.6. T-SQL' DE KONTROL DEYİMLERİ

---

Diğer programlama dillerinde olan ve T-SQL'de yer alan bu kontrol deyimleri şunlardır:

- **IF... ELSE** Yapısı
- **CASE** Yapısı
- **WHILE** Döngüsü

Bu üç yapı yapılan SQL sorgusunun herhangi bir noktasında bazı değerlerin kontrol edilerek çalıştırılmasını sağlamaktadırlar.

## *IF-ELSE DEYİMİ*

---

**IF – ELSE** karar yapısı diğer programlama dillerinden de hatırlanacağı gibi gelen **değer yada değerlere göre bir kararın verilmesini sağlayan** bir yapıdır. Bu yapının T-SQL deki genel kullanım biçimi şöyledir.

**IF (şartlar)**

**BEGIN**

**Komutlar1**

**END**

**ELSE**

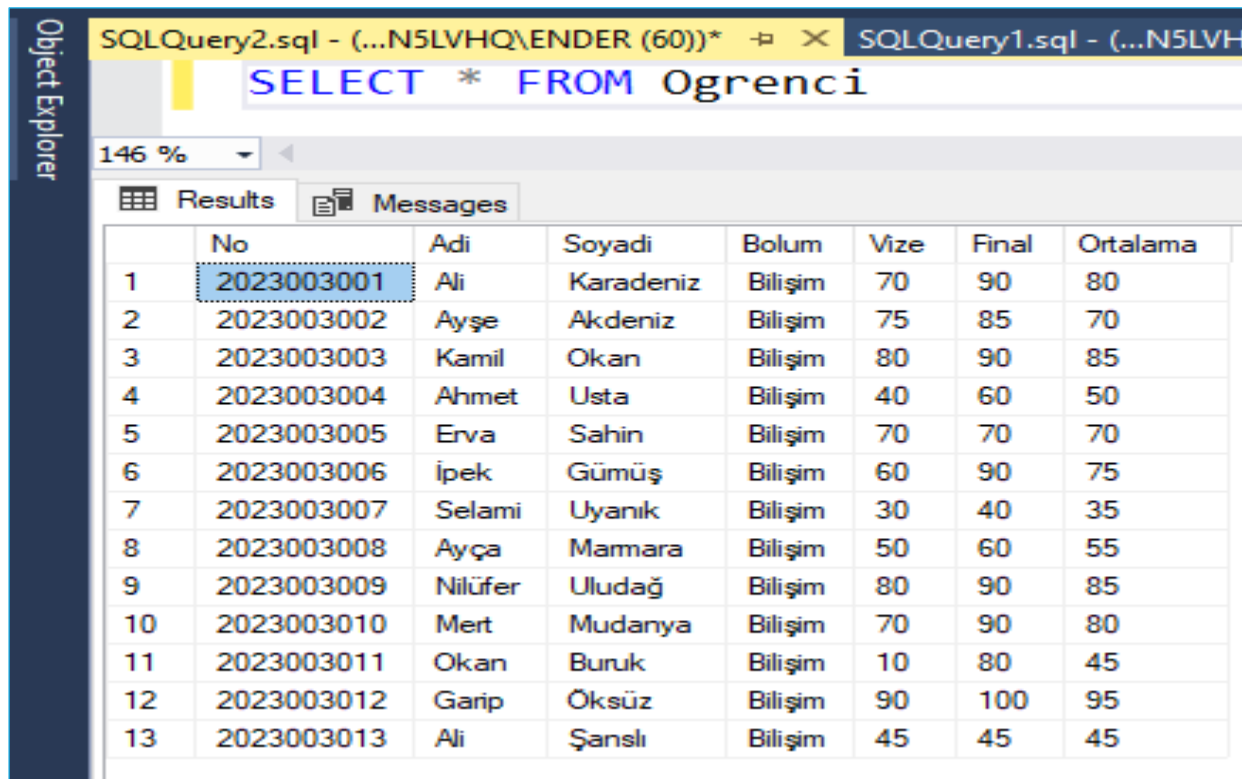
**BEGIN**

**Komutlar2**

**END**

## IF-ELSE DEYİMİ

**Örnek:** Aşağıda verilen Öğrenci tablosunda öğrencilerin ders geçme notlarının **ortalaması** 70'den büyük ise “Sınıfın durumu İYİ”, düşük ise “sınıfın durumu İYİ DEĞİL” olarak uyarı veren bir T-SQL programı yazalım.

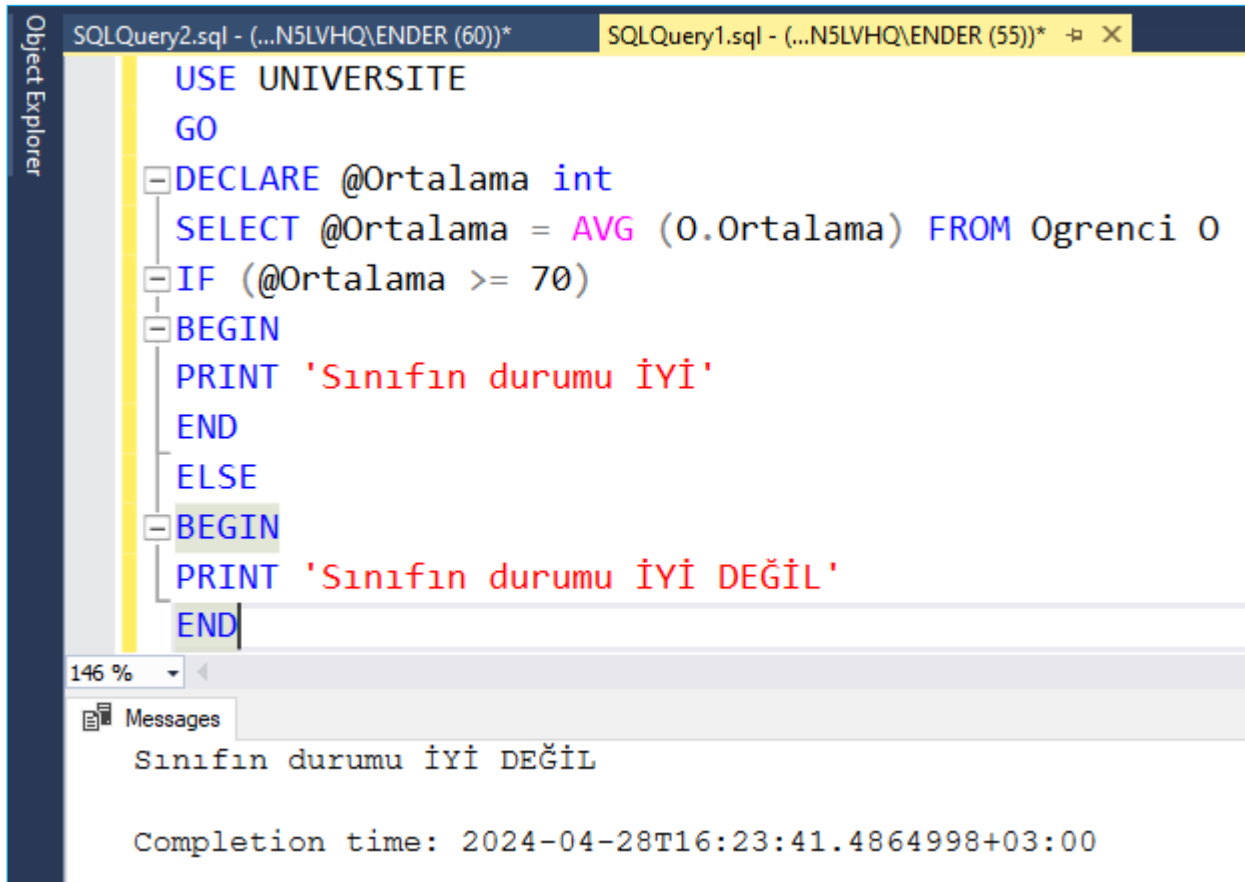


The screenshot shows a SQL Server Enterprise Manager interface. On the left is the 'Object Explorer' pane. The main window displays two SQL query files: 'SQLQuery2.sql' and 'SQLQuery1.sql'. The 'SQLQuery2.sql' file is active, showing the query: `SELECT * FROM Oğrenci`. Below the query editor, there are tabs for 'Results' and 'Messages'. The 'Results' tab is selected, showing a grid of data from the 'Oğrenci' table. The grid has 13 rows and 8 columns: 'No', 'Adi', 'Soyadi', 'Bolum', 'Vize', 'Final', and 'Ortalama'. The first row is highlighted with a blue border.

	No	Adi	Soyadi	Bolum	Vize	Final	Ortalama
1	2023003001	Ali	Karadeniz	Bilişim	70	90	80
2	2023003002	Ayşe	Akdeniz	Bilişim	75	85	70
3	2023003003	Kamil	Okan	Bilişim	80	90	85
4	2023003004	Ahmet	Usta	Bilişim	40	60	50
5	2023003005	Erva	Sahin	Bilişim	70	70	70
6	2023003006	İpek	Gümüş	Bilişim	60	90	75
7	2023003007	Selami	Uyanık	Bilişim	30	40	35
8	2023003008	Ayça	Mamara	Bilişim	50	60	55
9	2023003009	Nilüfer	Uludağ	Bilişim	80	90	85
10	2023003010	Mert	Mudanya	Bilişim	70	90	80
11	2023003011	Okan	Buruk	Bilişim	10	80	45
12	2023003012	Garip	Öksüz	Bilişim	90	100	95
13	2023003013	Ali	Şanslı	Bilişim	45	45	45

## IF-ELSE DEYİMİ

**Örnek:** Aşağıda verilen Öğrenci tablosunda öğrencilerin ders geçme notlarının **ortalaması** 70'den büyük ise “Sınıfın durumu İYİ”, düşük ise “sınıfın durumu İYİ DEĞİL” olarak uyarı veren bir T-SQL programı yazalım.



```
USE UNIVERSITE
GO
DECLARE @Ortalama int
SELECT @Ortalama = AVG (O.Ortalama) FROM Ogrrenci O
IF (@Ortalama >= 70)
BEGIN
PRINT 'Sınıfın durumu İYİ'
END
ELSE
BEGIN
PRINT 'Sınıfın durumu İYİ DEĞİL'
END
```

146 %

Messages

Sınıfın durumu İYİ DEĞİL

Completion time: 2024-04-28T16:23:41.4864998+03:00

## *CASE DEYİMİ*

---

**CASE** yapısı herhangi bir değişkenin birden fazla durumla karşılaştırmak için kullanılmaktadır. Genel yapısı şöyledir.

**CASE**

**WHEN** Şart\_ifadesi1 **THEN** Değer1

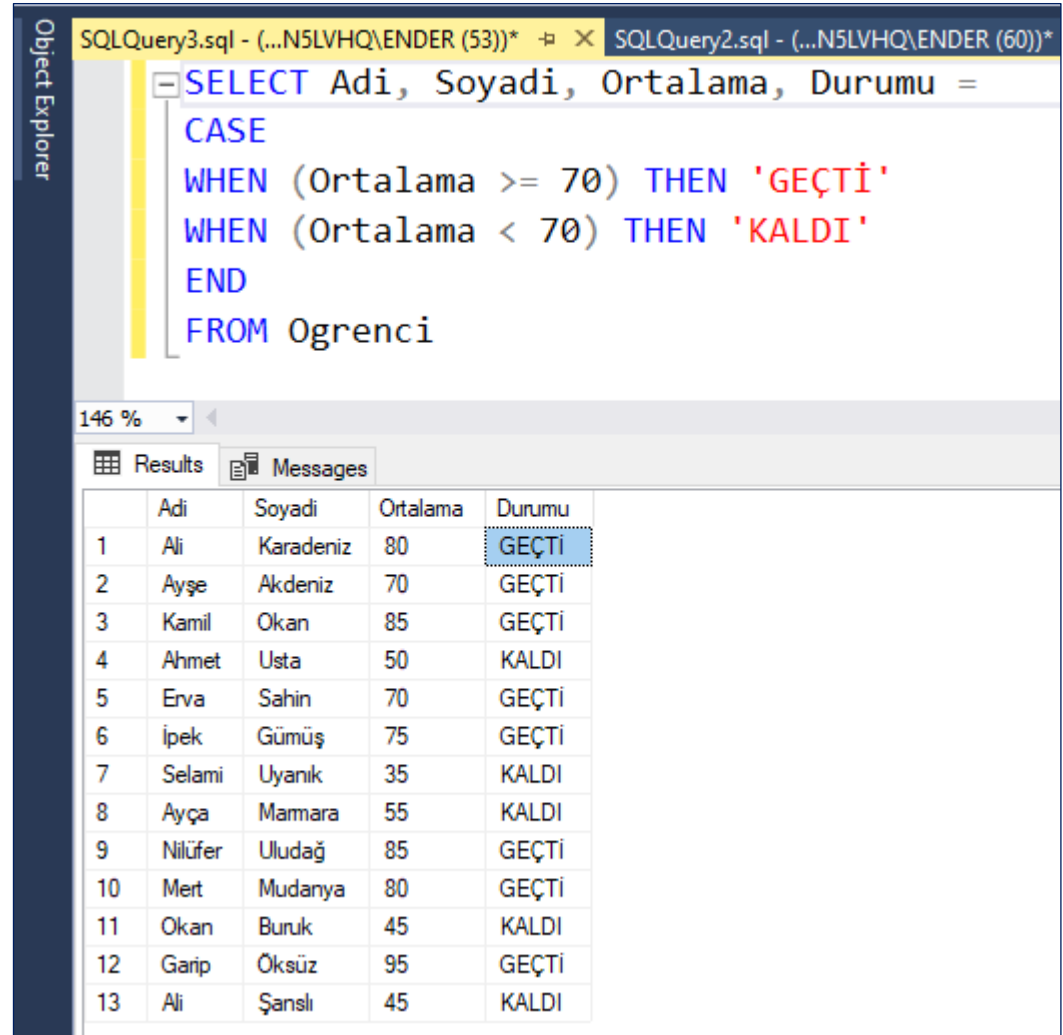
**WHEN** Şart\_ifadesi2 **THEN** Değer2

**ELSE** Değer3

**END**

## CASE DEYİMİ (DEVAM....)

Örnek: sınıftaki öğrencilerin not ortalamalarına göre **70** ve üzeri olanlarda “**GEÇTİ**” altında olanların durumunu “**KALDI**” şeklinde ekrana listeleyen T-SQL cümlecği ve elde edilen sonuç liste:



The screenshot displays the SQL Server Enterprise Manager interface. The left pane shows the 'Object Explorer' with a tree view containing 'SQLQuery3.sql' and 'SQLQuery2.sql'. The right pane shows the SQL query editor with the following T-SQL code:

```
SELECT Adi, Soyadi, Ortalama, Durumu =  
CASE  
WHEN (Ortalama >= 70) THEN 'GEÇTİ'  
WHEN (Ortalama < 70) THEN 'KALDI'  
END  
FROM Ogrenci
```

Below the query editor, the 'Results' tab is active, showing a table with 13 rows and 5 columns: 'Adi', 'Soyadi', 'Ortalama', and 'Durumu'. The first row is highlighted with a blue border.

	Adi	Soyadi	Ortalama	Durumu
1	Ali	Karadeniz	80	GEÇTİ
2	Ayşe	Akdeniz	70	GEÇTİ
3	Kamil	Okan	85	GEÇTİ
4	Ahmet	Usta	50	KALDI
5	Erva	Sahin	70	GEÇTİ
6	İpek	Gümüş	75	GEÇTİ
7	Selami	Uyanık	35	KALDI
8	Ayça	Mamara	55	KALDI
9	Nilüfer	Uludağ	85	GEÇTİ
10	Mert	Mudanya	80	GEÇTİ
11	Okan	Buruk	45	KALDI
12	Garip	Öksüz	95	GEÇTİ
13	Ali	Şanslı	45	KALDI



## *WHILE DÖNGÜSÜ*

---

Döngüler bir işlemde tekrar gerektiren durumlarda kullanılmaktadır. Döngüler belirlenen şart sağlanıncaya kadar işlemin yapılmasını sağlamaktadır. Diğer dillerde olduğu gibi T-SQL içerisinde de tekrarlı işlemler için bir döngü mevcuttur.

Bu döngünün genel yapısı aşağıdaki gibidir.

**WHILE Şart**

**BEGIN**

**Tekrar gerektiren kodlar**

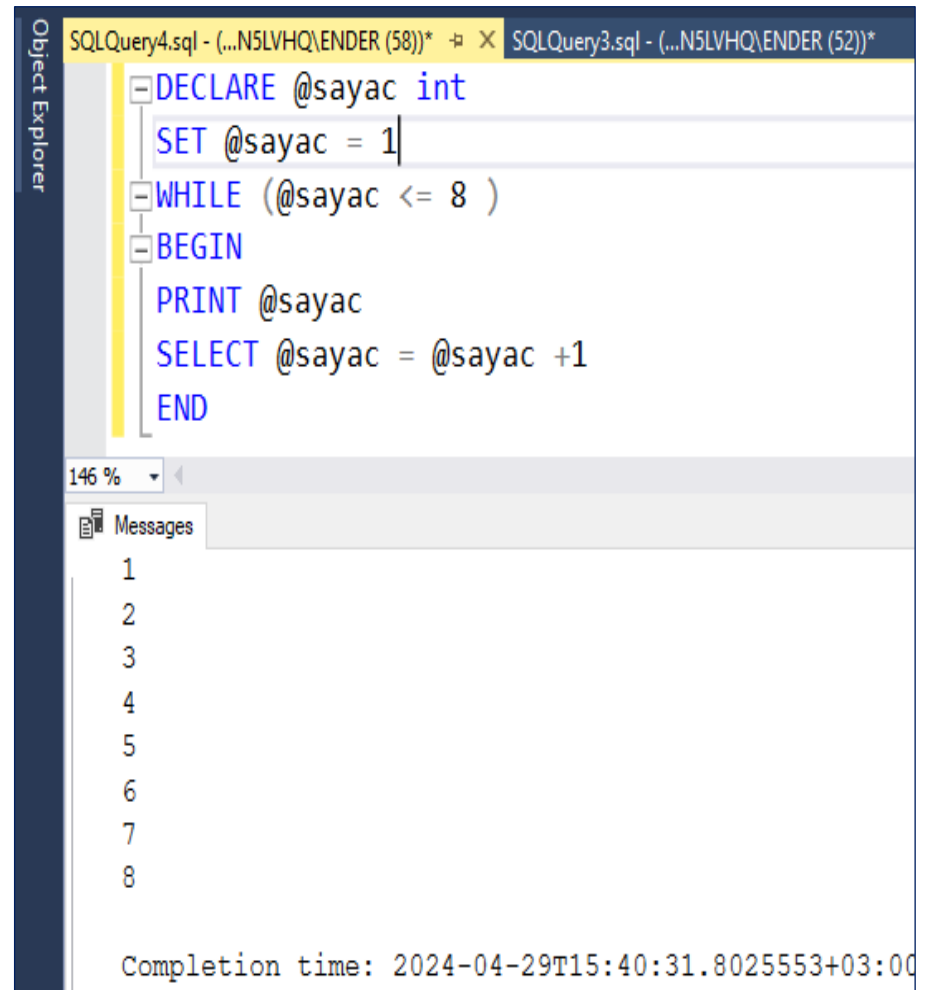
.....

**END**

## WHILE DÖNGÜSÜ (DEVAM...)

### Örnek 1:

Basit bir  
döngü ile  
8'e kadar  
sayıları  
yazdıralım



The screenshot shows a SQL Server Enterprise Manager window with two tabs: 'SQLQuery4.sql - (...N5LVHQ\ENDER (58))\*' and 'SQLQuery3.sql - (...N5LVHQ\ENDER (52))\*'. The active tab displays a SQL script for a WHILE loop. The script declares an integer variable @sayac, sets it to 1, and enters a loop that prints the value and increments it until it reaches 8. The output pane shows the numbers 1 through 8. The completion time is 2024-04-29T15:40:31.8025553+03:00.

```
SQLQuery4.sql - (...N5LVHQ\ENDER (58))*  SQLQuery3.sql - (...N5LVHQ\ENDER (52))*
--DECLARE @sayac int
--SET @sayac = 1
--WHILE (@sayac <= 8 )
--BEGIN
--    PRINT @sayac
--    SELECT @sayac = @sayac +1
--END
```

146 %

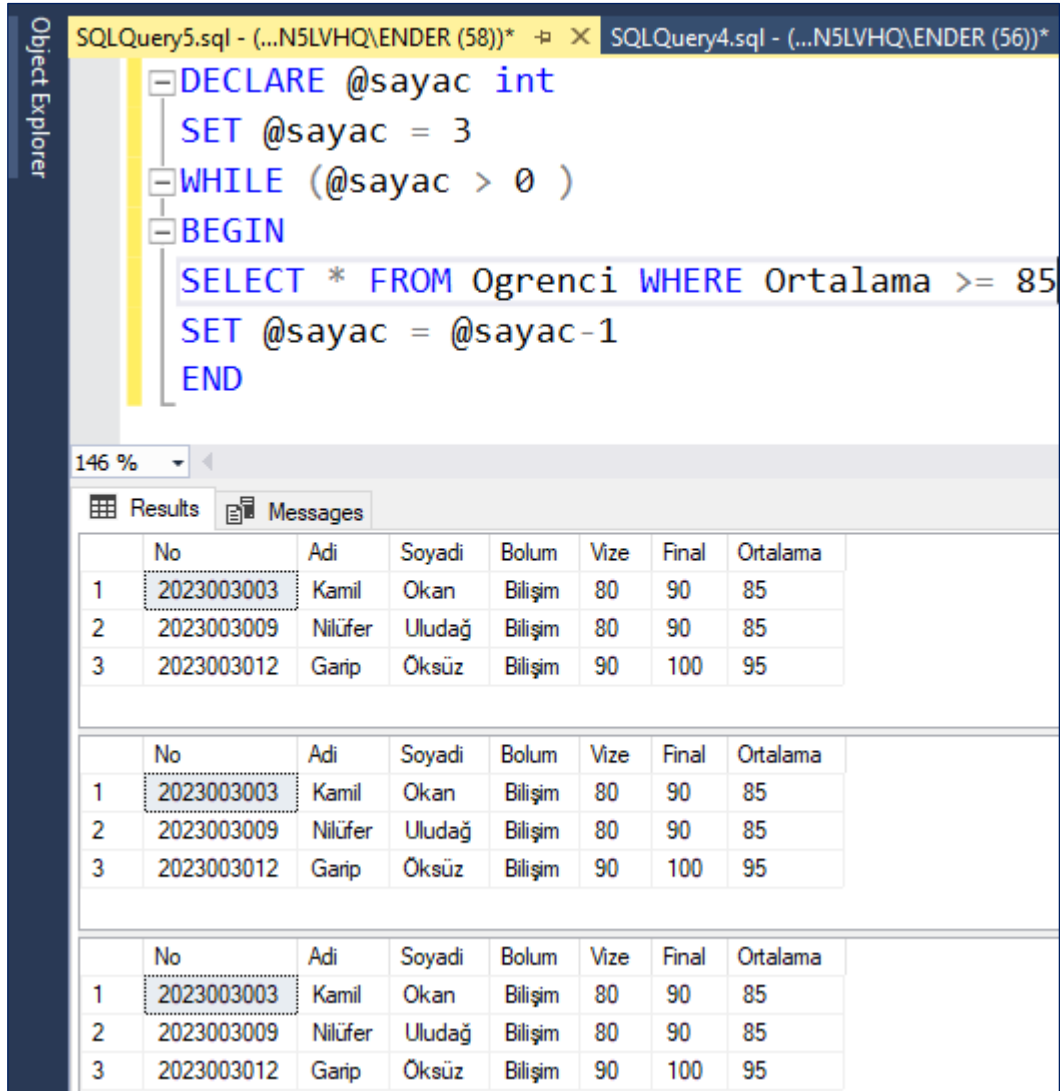
Messages

1  
2  
3  
4  
5  
6  
7  
8

Completion time: 2024-04-29T15:40:31.8025553+03:00

## WHILE DÖNGÜSÜ (DEVAM...)

Örnek2: Bazı durumlarda belli bir SQL cümleciğinin birden fazla çalıştırmak istenebilir. Bu durumda **döngü** kullanılabilir. Yandaki örnekte SQL ifadesini **3 defa** çalıştıralım.



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a SQL query in a file named 'SQLQuery5.sql'. The query is a WHILE loop that declares a counter variable @sayac, sets it to 3, and then enters a loop that runs as long as @sayac is greater than 0. Inside the loop, it selects all records from the 'Ogrenci' table where the average score (Ortalama) is greater than or equal to 85, and then decrements the counter by 1. The bottom pane shows the results of the query, which are displayed in three identical tables. Each table has 8 columns: No, Adi, Soyadi, Bolum, Vize, Final, and Ortalama. The data in each table is as follows:

	No	Adi	Soyadi	Bolum	Vize	Final	Ortalama
1	2023003003	Kamil	Okan	Bilişim	80	90	85
2	2023003009	Nilüfer	Uludağ	Bilişim	80	90	85
3	2023003012	Garip	Öksüz	Bilişim	90	100	95

	No	Adi	Soyadi	Bolum	Vize	Final	Ortalama
1	2023003003	Kamil	Okan	Bilişim	80	90	85
2	2023003009	Nilüfer	Uludağ	Bilişim	80	90	85
3	2023003012	Garip	Öksüz	Bilişim	90	100	95

	No	Adi	Soyadi	Bolum	Vize	Final	Ortalama
1	2023003003	Kamil	Okan	Bilişim	80	90	85
2	2023003009	Nilüfer	Uludağ	Bilişim	80	90	85
3	2023003012	Garip	Öksüz	Bilişim	90	100	95

## ***BREAK DEYİMİ***

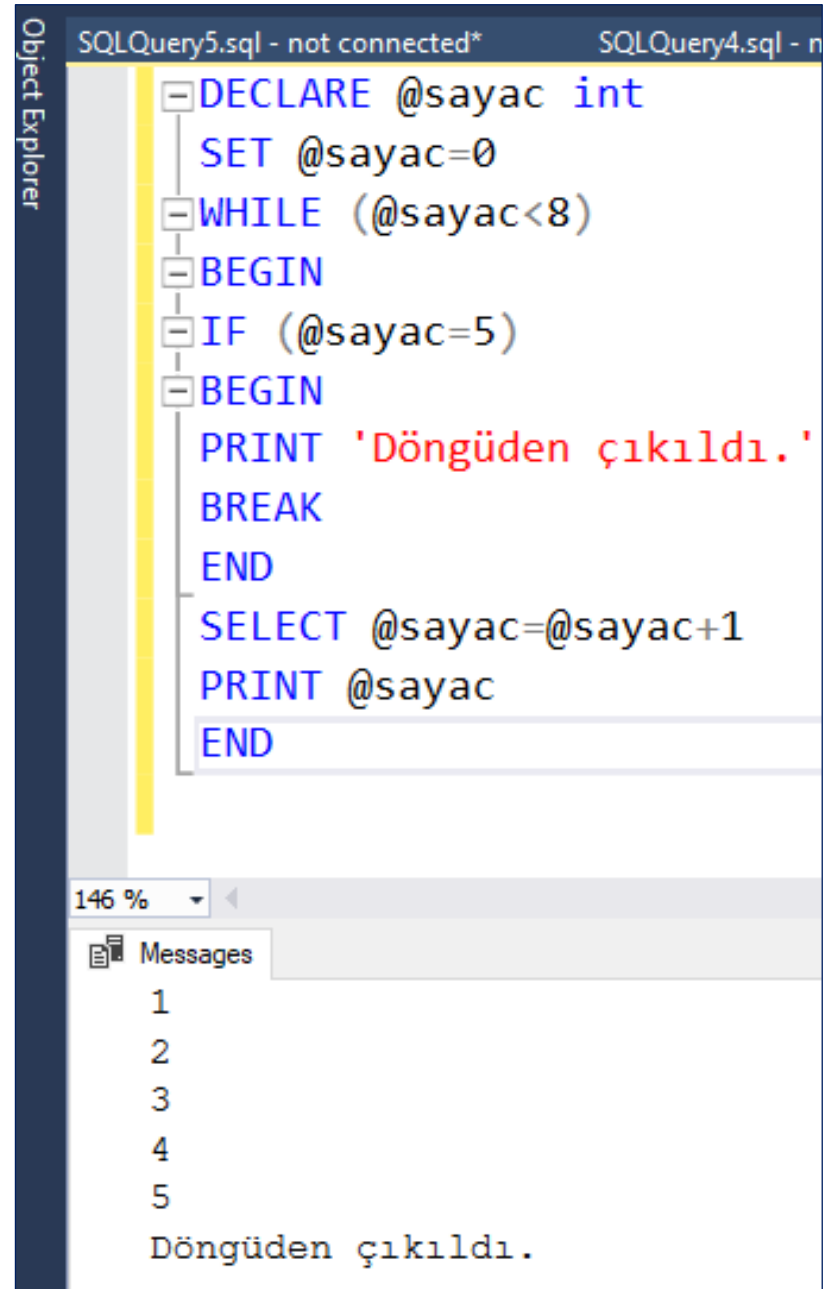
---

**BREAK** komutu, bir döngü içerisinde belirlenen amaca ulaşıldığında döngünün kırılarak sonlandırılmasını sağlamaktadır.

Döngüden çıkıldığında **WHILE**'in **END**'ini takip eden kodlardan program çalışmaya devam eder.

## ***BREAK DEYİMİ***

- Örnek: 1'den 8'e kadar çalışan bir döngüde 5'e gelindiğinde döngüden çıkmasını sağlayalım.



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a SQL query in 'SQLQuery5.sql - not connected\*'. The query is a T-SQL script that declares a variable @sayac as an integer, sets it to 0, and enters a WHILE loop that runs as long as @sayac is less than 8. Inside the loop, there is an IF statement that checks if @sayac equals 5. If true, it prints 'Döngüden çıkıldı.' and uses the BREAK statement to exit the loop. After the IF statement, it increments @sayac by 1 and prints its current value. The bottom pane shows the 'Messages' window with the output of the query execution. It lists the numbers 1 through 5, followed by the message 'Döngüden çıkıldı.', which confirms that the loop was successfully broken when @sayac reached 5.

```
SQLQuery5.sql - not connected*  SQLQuery4.sql - n
Object Explorer

-- DECLARE @sayac int
-- SET @sayac=0
-- WHILE (@sayac<8)
-- BEGIN
-- IF (@sayac=5)
-- BEGIN
-- PRINT 'Döngüden çıkıldı.'
-- BREAK
-- END
-- SELECT @sayac=@sayac+1
-- PRINT @sayac
-- END

146 %
Messages
1
2
3
4
5
Döngüden çıkıldı.
```

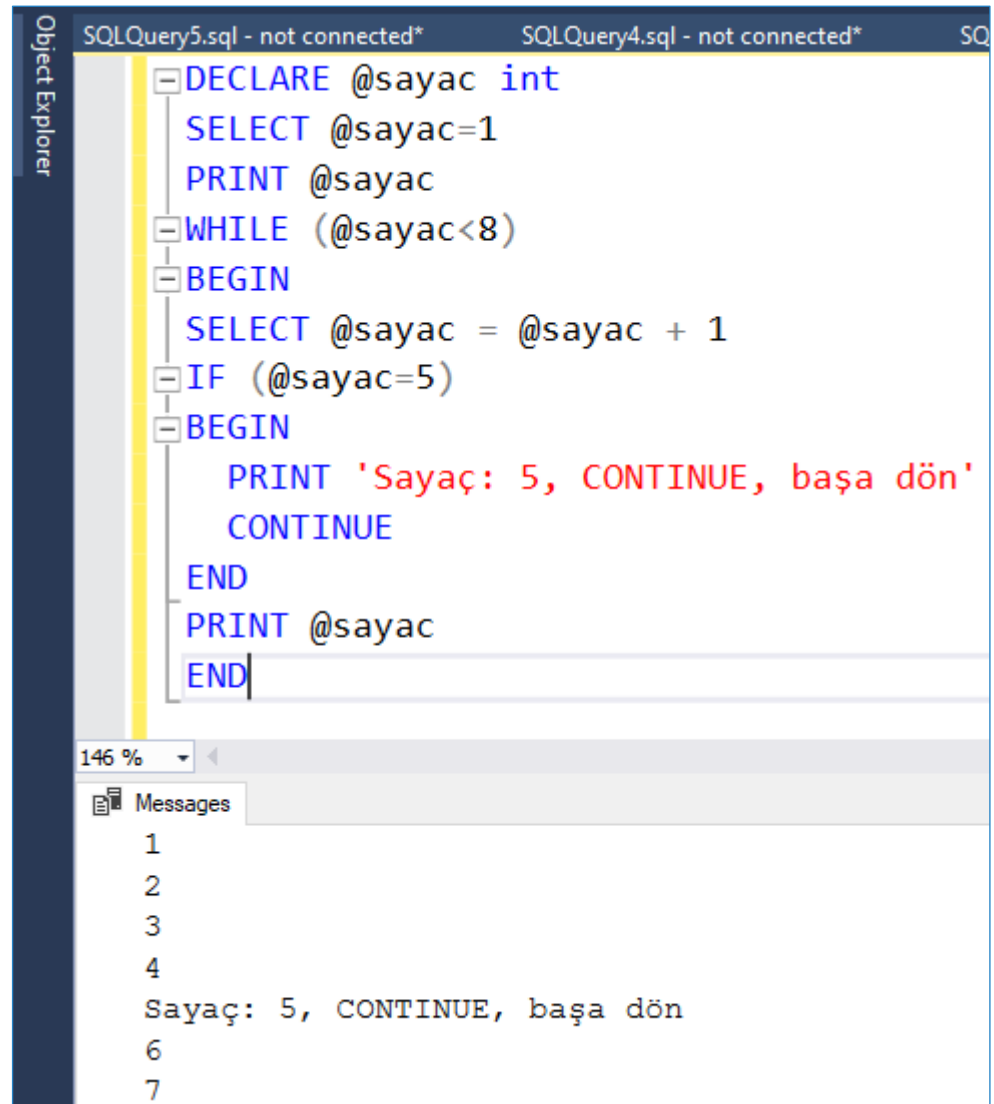
## *CONTINUE DEYİ Mİ*

---

- Bu komut BREAK komutunun tam tersi durumunu gerçekleştirir.
- BREAK komutunun aksine **CONTINUE** komutu **WHILE** yapısının başına gitmeyi sağlar. Bir defaya mahsus komutundan sonraki kısımlar çalıştırılmaz.

## CONTINUE DEYİMİ

Örnek: 1'den 8'e kadar saymaya çalışan bir döngüde 5'e gelindiğinde **CONTINUE** kullanılması nedeniyle **döngü başına dönen** bir örnek.



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a SQL query in 'SQLQuery5.sql - not connected\*'. The query is a WHILE loop that increments a counter @sayac from 1 to 8. It includes a PRINT statement to show the counter value. A nested IF statement checks if the counter is 5; if true, it executes a BEGIN block containing a PRINT statement with the text 'Sayaç: 5, CONTINUE, başa dön' followed by a CONTINUE statement to skip the rest of the loop body and restart the loop. The bottom pane shows the 'Messages' window with the execution results. The output shows the counter values 1 through 4, followed by the message 'Sayaç: 5, CONTINUE, başa dön', and then 6 and 7. The counter value 8 is not shown, indicating the loop restarted at 1 after the CONTINUE statement was executed.

```
SQLQuery5.sql - not connected* SQLQuery4.sql - not connected* SQ
Object Explorer
-- DECLARE @sayac int
-- SELECT @sayac=1
-- PRINT @sayac
-- WHILE (@sayac<8)
-- BEGIN
--   SELECT @sayac = @sayac + 1
--   IF (@sayac=5)
--   BEGIN
--     PRINT 'Sayaç: 5, CONTINUE, başa dön'
--     CONTINUE
--   END
--   PRINT @sayac
-- END

146 %
Messages
1
2
3
4
Sayaç: 5, CONTINUE, başa dön
6
7
```

## 9.2. **VIEW** (GÖRÜNÜM) NESNESİ

---

- **Görünümler**, veritabanı içerisinde önceden tanımlanmış tabloları kullanarak elde edilen geçici sanal tablolardır.
- Bu tablolar **veritabanında fiziksel herhangi bir yer kaplamazlar**.
- Görünümler, **herhangi bir tablodaki belli alanların listelenmesini sağlarlar**.
- Bir veritabanı üzerinde **eğer sık sorgulanan veriler var ise**, bu verilerin **hızlı ve basit bir şekilde sorgulanmalarını** sağlamaktadırlar.



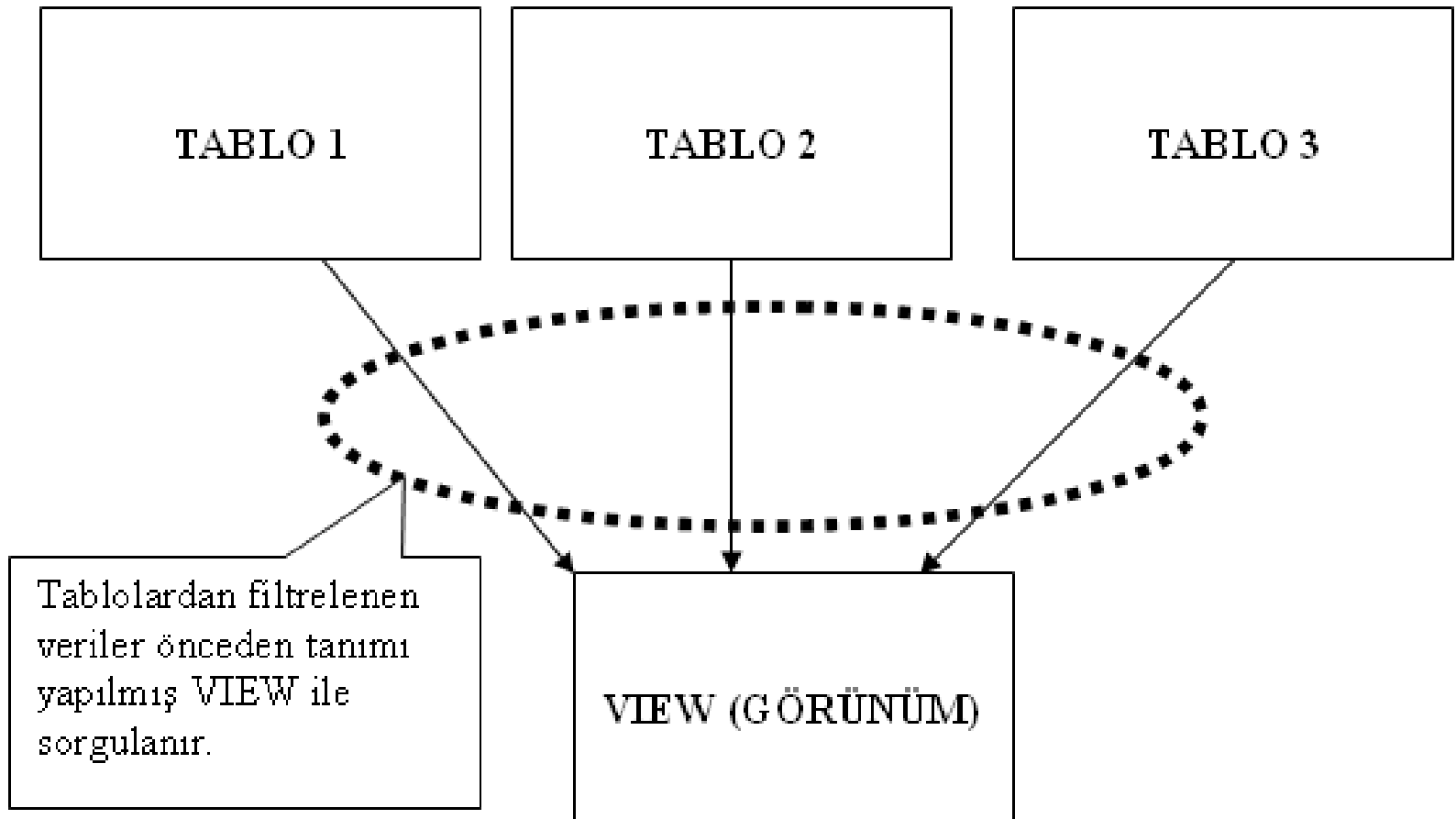
## 9.2. VIEW (GÖRÜNÜM) NESNESİ (DEVAM...)

---

- **Görünümler** **kaydedilmiş** **sorgulardan** oluşmaktadır.
- **Bir görünüm ile birden fazla tablo üzerinde sorgulama** yapılabilir. Bu haliyle karmaşık SQL ifadelerini daha da basitleştirmiş olur.
- **Oluşturulan bir görünüm ile veriler üzerinde herhangi bir **güncelleme işlemi** **yapılamaz**.**
- **Bu yapılar MS SQL Server, Oracle** gibi ileri seviye veritabanı yönetim sistemlerinde bulunmaktadır.

## 9.2. VIEW (GÖRÜNÜM) NESNESİ (DEVAM...)

- Bir görünümün genel yapısı şekildeki gibidir.



## 9.2. VIEW (GÖRÜNÜM) NESNESİ (DEVAM...)

---

- Genel olarak **görünümler aşağıdaki amaçlar için de kullanılmaktadırlar:**
  - Kullanıcıların, bir yada birden fazla tabloda yer alan sütun veya satırlarında yer alan **önemli verilerini görmek.**
  - Tabloda bulunan **verileri istenilen bir tablo formatında göstermek.**
  - **Karmaşık sorguları basitleştirmek.**

## 9.2.1. VIEW OLUŞTURMAK

---

- Görünümler tablolarla aynı özelliklere sahip olup **en fazla 1024 tane sütundan** oluşabilmektedir. Görünüm oluşturmak için **CREATE VIEW** komutu kullanılmaktadır.
- Görünümün genel yapısı şöyledir:

**CREATE VIEW** view\_adı

**AS**

**SELECT** sütunlar **FROM** tablo\_adı

## 9.2.1. VIEW OLUŞTURMAK (DEVAM...)

- Örnek:** tblUrun tablosunda urunKod, urunAd ve listeFiyat bilgilerini gösteren bir View oluşturalım.

Object Explorer

SQLQuery8.sql - (...N5LVHQ\ENDER (55))\* SQLQuery7.sql - (...N5LVHQ\ENDER (62))\* SQLQuery5.sql - not connected\* SQLQuery4.sql - not

SELECT \* FROM Urun

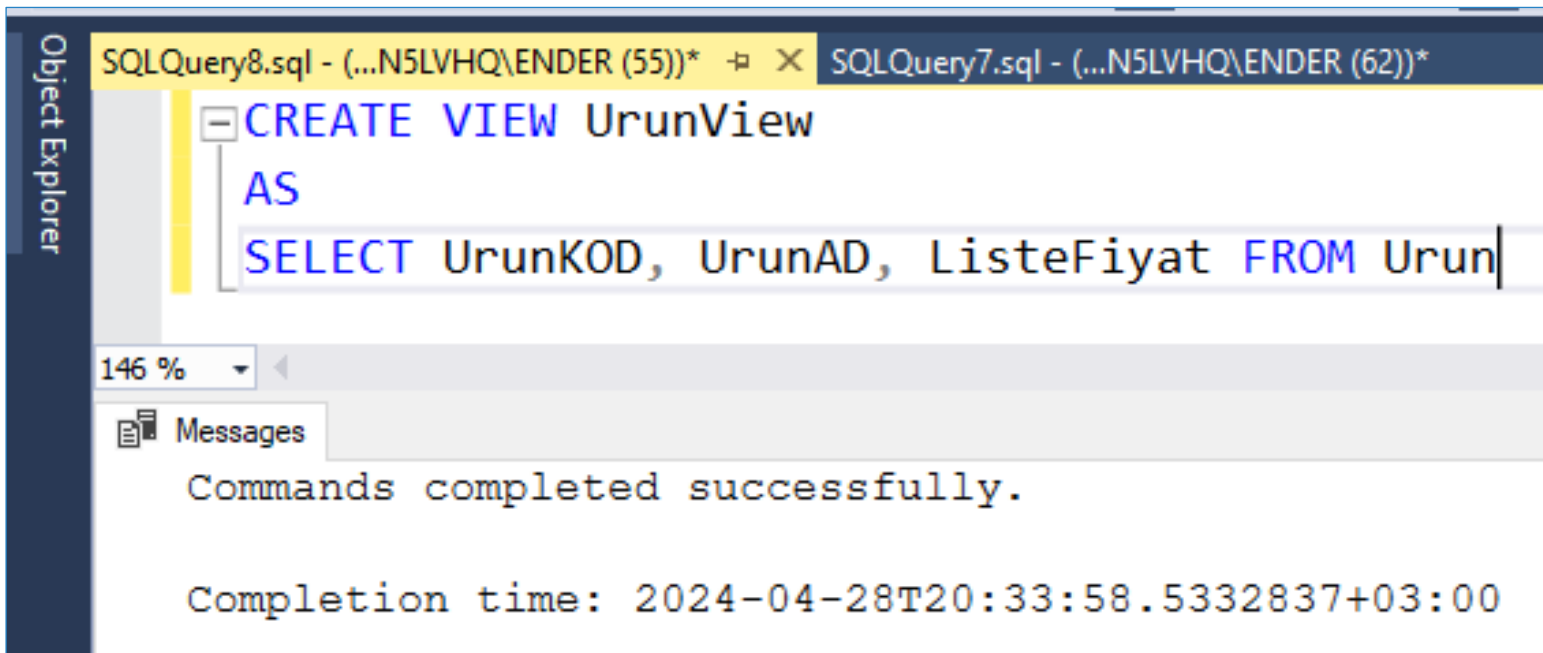
146 %

Results Messages

	urunKod	Barkod	bolgeKod	markaKod	urunAd	bayiFiyat	listeFiyat	Indirim	KDVoran	urunRe
1	3505	NULL	NULL	NULL	Staj Defteri	NULL	2700.0000	NULL	NULL	NULL
2	3391	3CP015098	NULL	1	HomeConnect PC Digital Web Kamera	20.0000	28.1697	0.0000	0.1800	NULL
3	3026	KBS-720-F	NULL	2	PS/2 F TR Klavye (Anti-RSI)	4.3000	6.0461	0.0000	0.1800	NULL
4	3027	KBS-720	NULL	2	PS/2 Q TR Anti-RSI Standart Klavye	4.5000	6.3565	0.0000	0.1800	NULL
5	3028	KBS-21	NULL	2	PS/2 Q Anti-Rsi MM Klavye (Siyah/Gri)	7.5000	10.5776	0.0000	0.1800	NULL
6	3029	KBS-8	NULL	2	PS/2 Q TR Multimedya Klavye (Anti-RSI)	7.5000	10.5776	0.0000	0.1800	NULL
7	3030	KBS-827	NULL	2	PS/2 Q TR Multimedya Klavye + Kablosuz Mouse (An...	23.0000	32.3908	0.0000	0.1800	NULL
8	3031	KB-827	NULL	2	PS/2 Q TR Multimedya Klavye + Kablosuz Mouse	23.0000	32.3908	0.0000	0.1800	NULL
9	3032	KBA-527RA	NULL	2	PS/2 Q TR Klavye + Mouse (Anti-RSI)	25.5000	35.9291	0.0000	0.1800	NULL
10	3033	KBS-827R	NULL	2	PS/2 Q TR Multimedya Kablosuz Klavye + Kablosuz ...	26.0000	36.6243	0.0000	0.1800	NULL
11	3034	KBS-853RPCA	NULL	2	KBD Q PS/2 A4TECH A TIPI MM RF KLV+MOU	32.0000	45.0790	0.0000	0.1800	NULL
12	3035	KBS-2348RP	NULL	2	PS/2 Q TR Klavye + Mouse (Anti-RSI)	35.0000	49.3001	0.0000	0.1800	NULL
13	3036	KBS-2548RPA	NULL	2	PS/2 Q TR Klavye + Mouse (Anti-RSI)	35.0000	49.3001	0.0000	0.1800	NULL
14	3061	MOA-RP680	NULL	2	USB 2+2+2+1+1 Tuslu ve 1 Tekerlek	24.5000	34.5262	0.0000	0.1800	NULL
15	3062	MOA-SWW48P	NULL	2	PS/2 2 Tus, 1 Scroll Mouse	3.0000	4.2211	0.0000	0.1800	NULL
16	3063	MOA-RP648	NULL	2	USB Internet mouse, 2+1 tuslu ve 1 tekerlek	20.0000	28.1697	0.0000	0.1800	NULL
17	3064	MOA-RP649	NULL	2	USB 2+2+1 Tuslu ve 2 Tekerlek	21.0000	29.5850	0.0000	0.1800	NULL

## 9.2.1. VIEW OLUŞTURMAK (DEVAM...)

- MS SQL Server üzerinde yeni bir Query açarak oluşturacağımız «UrunView» adlı View'a ait T-SQL kodlarını yazarak çalıştıralım.



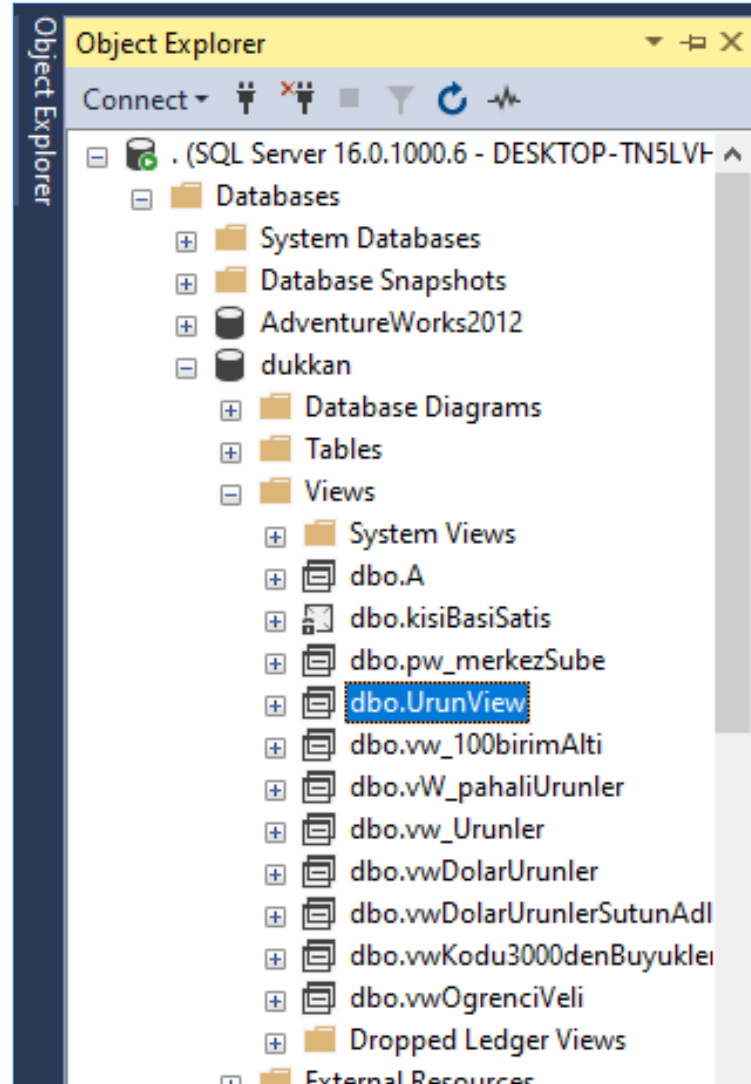
The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane is visible. The main window shows a query editor with two tabs: 'SQLQuery8.sql - (...N5LVHQ\ENDER (55))\*' and 'SQLQuery7.sql - (...N5LVHQ\ENDER (62))\*'. The active tab contains the following T-SQL code:

```
CREATE VIEW UrunView  
AS  
SELECT UrunKOD, UrunAD, ListeFiyat FROM Urun
```

Below the query editor, the 'Messages' pane shows the execution result: 'Commands completed successfully.' and the completion time: '2024-04-28T20:33:58.5332837+03:00'.

## 9.2.1. VIEW OLUŞTURMAK (DEVAM...)

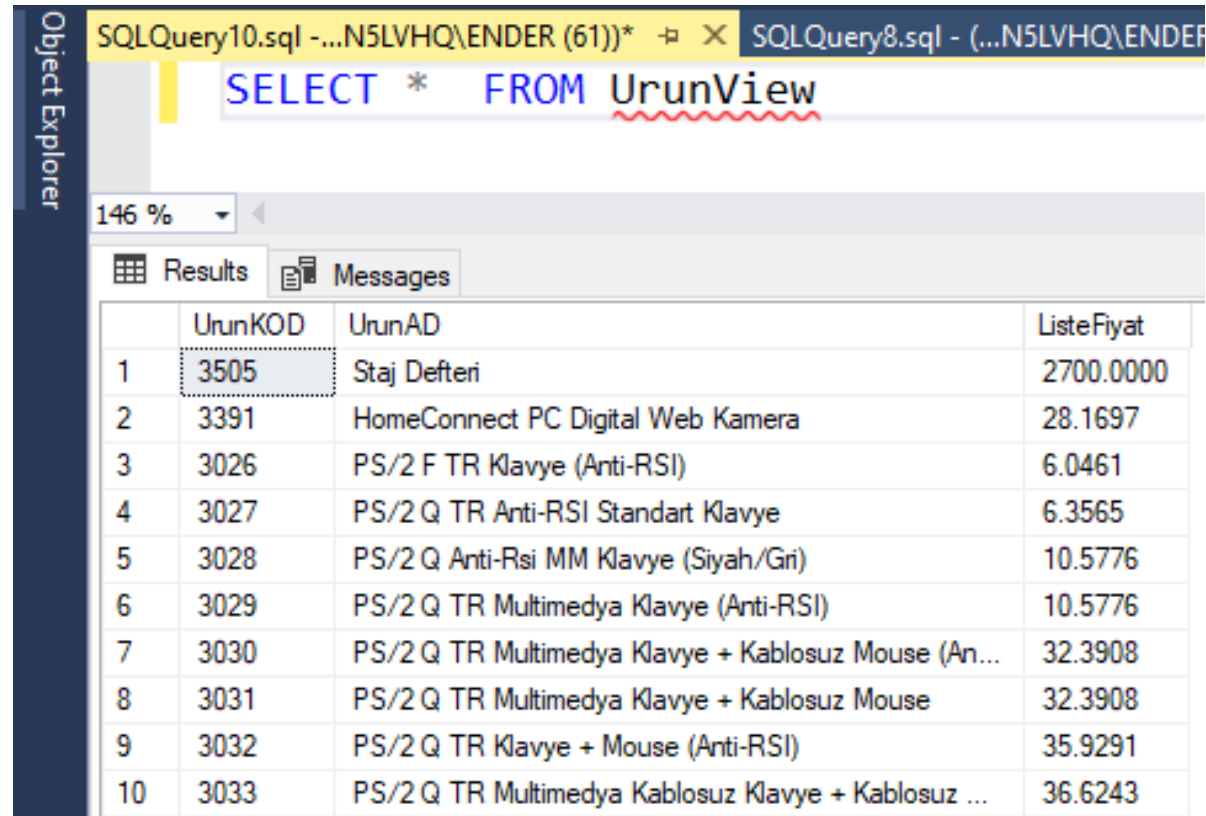
- Oluşturulan View' lar ilgili veritabanının «**views**» sekmesinde yer almaktadır ve **dbo.UrunView** şeklinde görüntülenmektedir.



## 9.2.1. VIEW LİSTELEME (DEVAM...)

Oluşturulan View standart SQL komutlarına benzer şekilde **SELECT** çalıştırılır.

Sorgu sonucu yandaki gibi olacaktır.



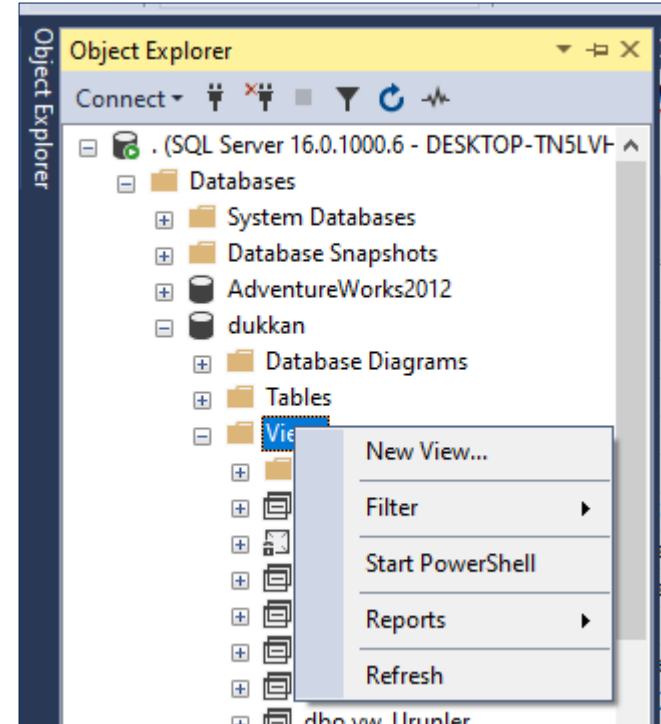
The screenshot shows the SQL Server Enterprise Manager interface. The 'Object Explorer' pane on the left is partially visible. The main window displays a query window titled 'SQLQuery10.sql - (...N5LVHQ\ENDER (61))' and 'SQLQuery8.sql - (...N5LVHQ\ENDER (61))'. The query text is 'SELECT \* FROM UrunView'. Below the query window, the 'Results' tab is active, showing a table with 10 rows and 4 columns: 'UrunKOD', 'UrunAD', and 'ListeFiyat'. The first row is highlighted. The zoom level is set to 146%.

	UrunKOD	UrunAD	ListeFiyat
1	3505	Staj Defteri	2700.0000
2	3391	HomeConnect PC Digital Web Kamera	28.1697
3	3026	PS/2 F TR Klavye (Anti-RSI)	6.0461
4	3027	PS/2 Q TR Anti-RSI Standart Klavye	6.3565
5	3028	PS/2 Q Anti-Rsi MM Klavye (Siyah/Gri)	10.5776
6	3029	PS/2 Q TR Multimedya Klavye (Anti-RSI)	10.5776
7	3030	PS/2 Q TR Multimedya Klavye + Kablosuz Mouse (An...	32.3908
8	3031	PS/2 Q TR Multimedya Klavye + Kablosuz Mouse	32.3908
9	3032	PS/2 Q TR Klavye + Mouse (Anti-RSI)	35.9291
10	3033	PS/2 Q TR Multimedya Kablosuz Klavye + Kablosuz ...	36.6243



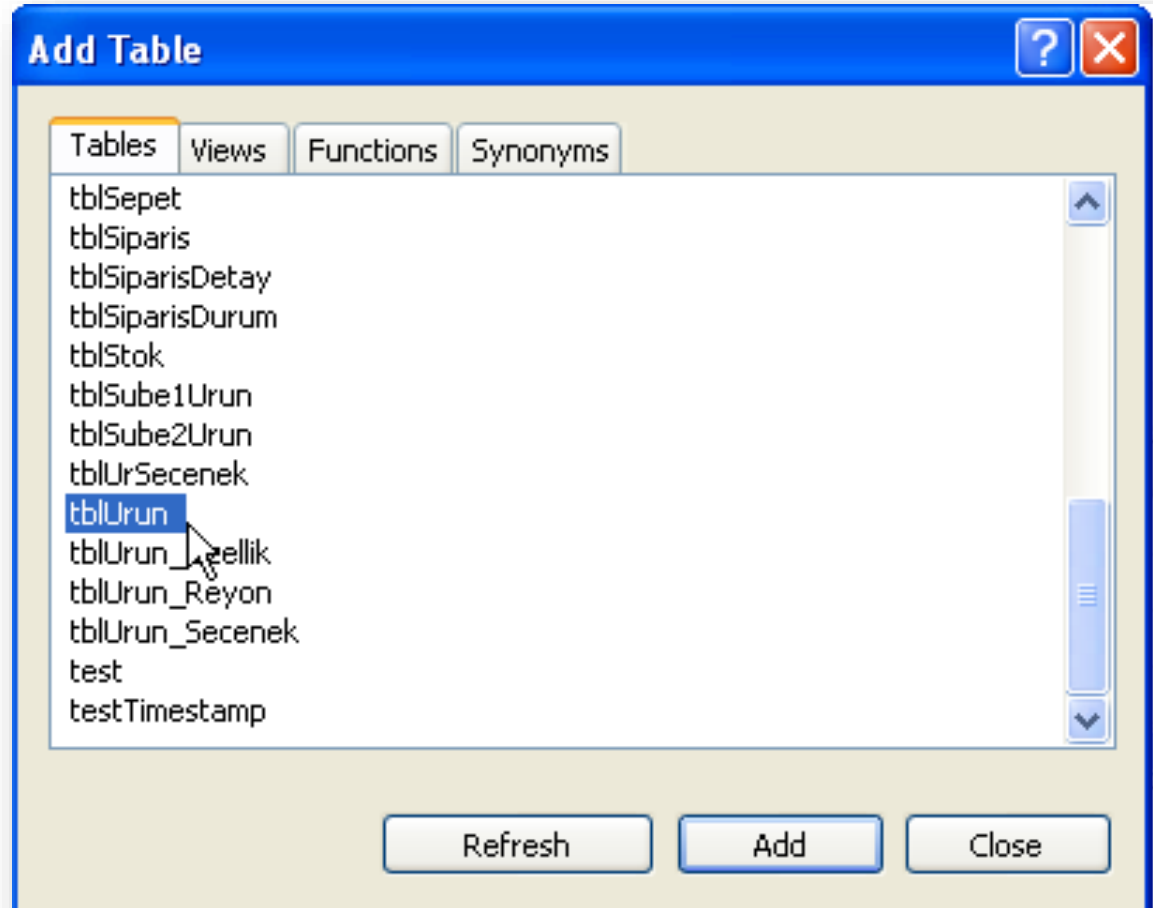
## 9.2.1. VIEW OLUŞTURMAK (DEVAM...)

**View oluşturma işlemini** dilerseniz MS SQL Server’da ilgili veritabanındaki «views» seçeneğini farenin sağ tuşuna basarak açılan menüden «**New View**» diyerek te oluşturabilirsiniz.



## 9.2.1. VIEW OLUŞTURMAK (DEVAM...)

«**New View**» seçeneğini tıkladığınızda karşınıza **View**'ı hangi tablo yada tablolardan oluşturacağınızı soran bir ekran gelecektir. Örneğimizde **tblUrun** tablosunu kullanarak yeniden yapalım.



## 9.2.1. VIEW OLUŞTURMAK (DEVAM...)

- tblUrun tablosu seçildiğinde View'da sorgulanması gereken alanların tek tek seçilmesi istenecektir. Seçilen bu alanlar View sorgulandığında ekranda gösterilecektir.

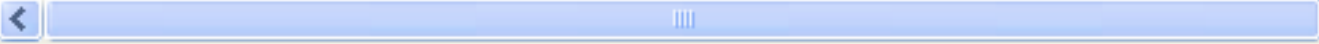
The screenshot shows a database software interface. On the left, a list of columns from the 'tblUrun' table is displayed. The columns are: \* (All Columns), urunKod, Barkod, bolgeKod, markaKod, urunAd, bayiFiyat, listeFiyat, Indirim, KDVorun, and urunResim. The columns 'urunKod', 'urunAd', and 'listeFiyat' are selected with checkmarks. Below the column list, a table shows the selected columns and their corresponding table names, output status, and sort options.

Column	Alias	Table	Output	Sort Type	Sort Order	Filter
urunKod		tblUrun	✓			
urunAd		tblUrun	✓			
listeFiyat		tblUrun	✓			
			■			

## 9.2.1. VIEW OLUŞTURMAK (DEVAM...)

Her seçilen alan bir **SELECT** ifadesi içerisine eklenir. Eğer bir filtreleme yapılacaksa **Filter** seçeneği içerisinde belirtilir.

	Column	Alias	Table	Output	Sort Type	Sort Order	Filter
▶	urunKod		tblUrun	<input checked="" type="checkbox"/>			= 2160
	urunAd		tblUrun	<input checked="" type="checkbox"/>			
	listeFiyat		tblUrun	<input checked="" type="checkbox"/>			
				<input type="checkbox"/>			



```
SELECT  urunKod, urunAd, listeFiyat
FROM    dbo.tblUrun
WHERE   (urunKod = 2160)
```

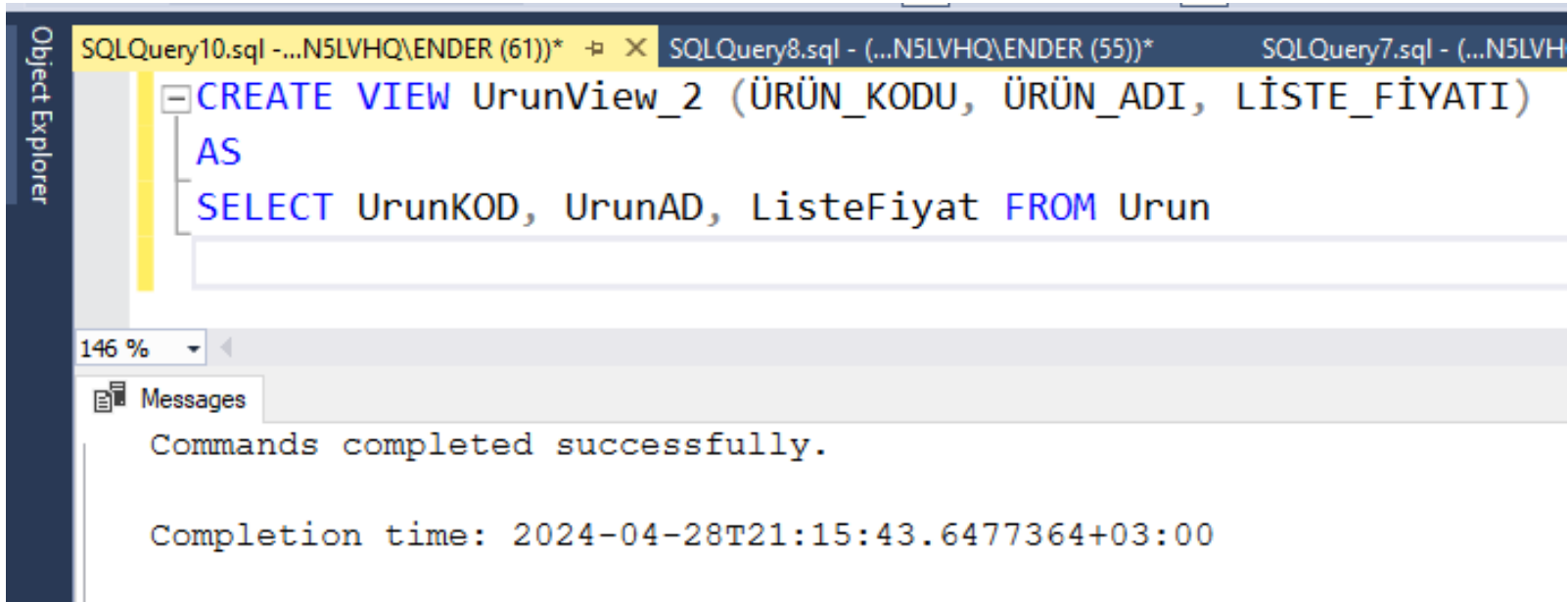
## 9.2.2. VIEW SÜTUNLARINA İSİM VERMEK

---

- View'lar da **sütun adları isteğe göre değiştirilebilir**. Eğer değiştirilmek istenmiyorsa filtrelenen ana tablodaki sütun adları ile aynı olacaktır.
- Sütun adları ihtiyaca göre değiştirilebilir. **View'larda** **sütun adları verilirken herhangi bir veri tipi belirtilmez**. Bunun nedeni sorgulanan tablo baz alındığı için veri tipleri yine asıl tablodaki veri tipleri ile aynı olacaktır.

## 9.2.2. VIEW SÜTUNLARINA İSİM VERMEK

Aynı örneğimizi sütunlara isim vererek tekrar yapalım.



The screenshot displays the SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane is visible. The main window shows a SQL query editor with the following code:

```
CREATE VIEW UrunView_2 (ÜRÜN_KODU, ÜRÜN_ADI, LİSTE_FİYATI)
AS
SELECT UrunKOD, UrunAD, ListeFiyat FROM Urun
```

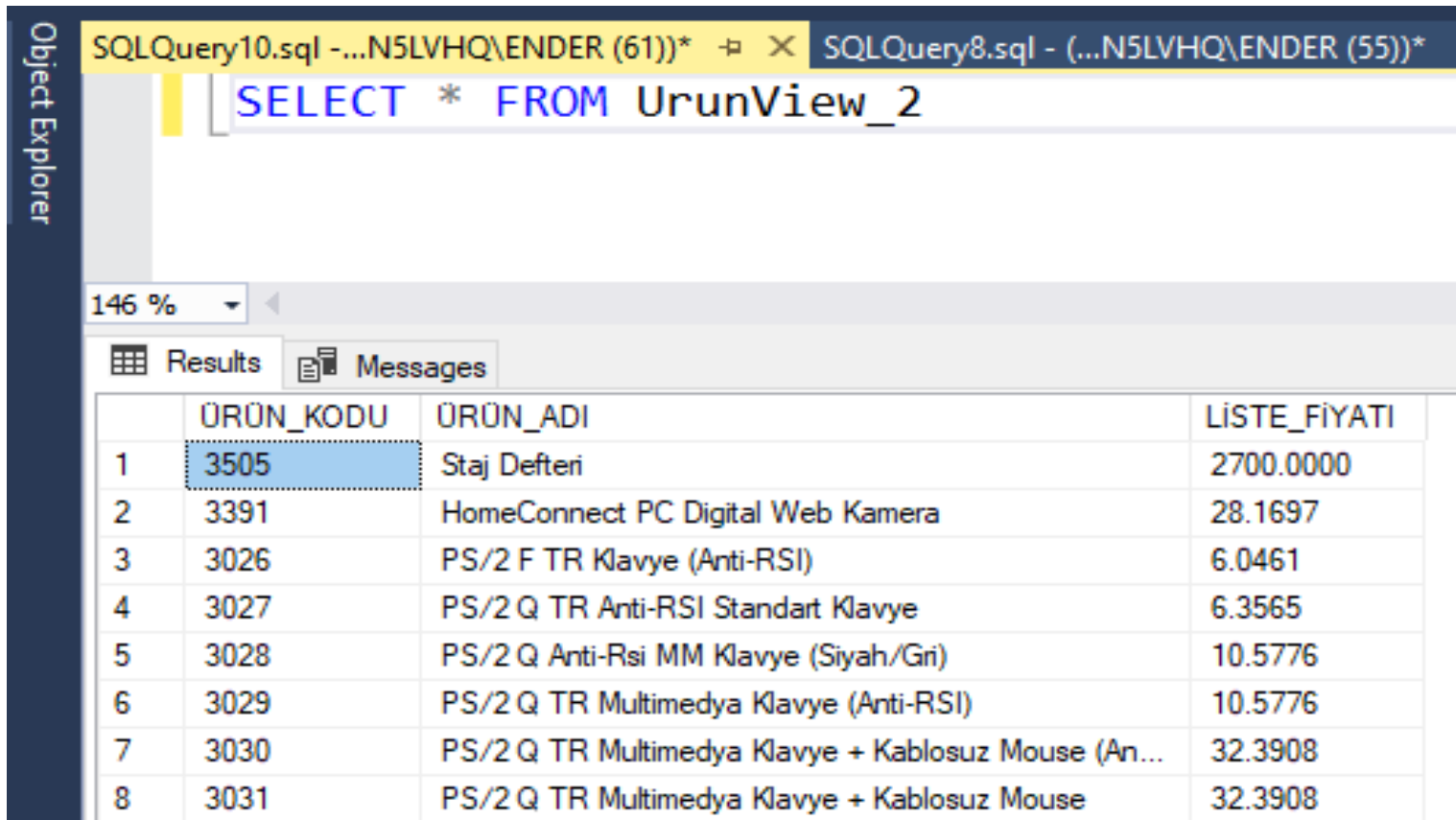
Below the query editor, the 'Messages' pane shows the following output:

```
Commands completed successfully.

Completion time: 2024-04-28T21:15:43.6477364+03:00
```

## 9.2.2. VIEW SÜTUNLARINA İSİM VERMEK

UrunView\_2 isimli yeni oluşturulan View'e ait kayıtların bir kısmı



The screenshot shows the SQL Server Enterprise Manager interface. The 'Object Explorer' on the left shows the 'SQLQuery10.sql' file selected. The main window displays the query 'SELECT \* FROM UrunView\_2' and its results. The results are shown in a table with 4 columns: ÜRÜN\_KODU, ÜRÜN\_ADI, and LISTE\_FİYATI. The first row is highlighted in blue.

	ÜRÜN_KODU	ÜRÜN_ADI	LISTE_FİYATI
1	3505	Staj Defteri	2700.0000
2	3391	HomeConnect PC Digital Web Kamera	28.1697
3	3026	PS/2 F TR Klavye (Anti-RSI)	6.0461
4	3027	PS/2 Q TR Anti-RSI Standart Klavye	6.3565
5	3028	PS/2 Q Anti-Rsi MM Klavye (Siyah/Gri)	10.5776
6	3029	PS/2 Q TR Multimedya Klavye (Anti-RSI)	10.5776
7	3030	PS/2 Q TR Multimedya Klavye + Kablosuz Mouse (An...	32.3908
8	3031	PS/2 Q TR Multimedya Klavye + Kablosuz Mouse	32.3908

### 9.2.3. BİRDEN FAZLA TABLO KULLANMAK

---

Bir **JOIN komutu** kullanılarak birden fazla tablodaki veriler bir **VIEW** oluşturularak defalarca sorgulanabilir. Genel olarak bu işlem için şöyle bir yapı kullanılır:

**CREATE VIEW** view\_adı

**AS**

**SELECT** sütun\_adları **FROM** tablo\_1 **JOIN**  
tablo\_2 **ON** birleştirme\_şartı



## 9.2.3. BİRDEN FAZLA TABLO KULLANMAK (DEVAM...)

Örnek: tblUrun ve tblMarka tablolarını kullanarak marka kodları aynı olan ürünleri listeleyen bir view yazalım. Kullanılacak olan tablolar aşağıdaki gibidir.

**tblUrun**

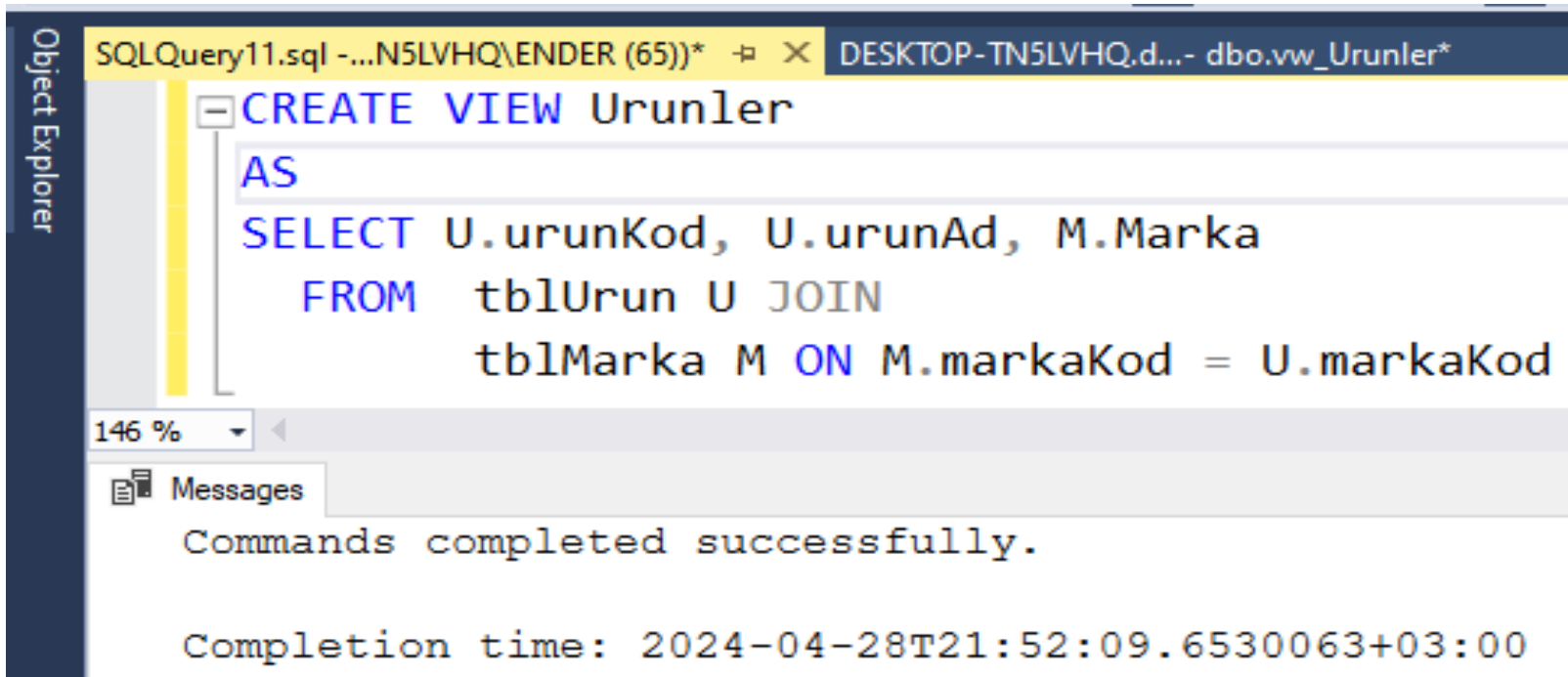
	urunKod	Barkod	bolgeKod	markaKod	urunAd
▶	3505	NULL	NULL	NULL	Staj Defteri
	3391	3CP015098	NULL	1	HomeConnect P...
	3026	KBS-720-F	NULL	2	PS/2 F TR Klavy...
	3027	KBS-720	NULL	2	PS/2 Q TR Anti-...
	3028	KBS-21	NULL	2	PS/2 Q Anti-Rsi ...
	3029	KBS-8	NULL	2	PS/2 Q TR Multi...
	3030	KBS-827	NULL	2	PS/2 Q TR Multi...
	3031	KB-827	NULL	2	PS/2 Q TR Multi...
	3032	KBA-527RA	NULL	2	PS/2 Q TR Klavy...
	3033	KBS-827R	NULL	2	PS/2 Q TR Multi...
	3034	KBS-853RPCA	NULL	2	KBD Q PS/2 A4T...
	3035	KBS-2348RP	NULL	2	PS/2 Q TR Klavy...
	3036	KBS-2548RPA	NULL	2	PS/2 Q TR Klavy...
	3061	MOA-RP680	NULL	2	USB 2+2+2+1+...
	3062	MOA-SWW48P	NULL	2	PS/2 2 Tus, 1 Sc...
	3063	MOA-RP648	NULL	2	USB Internet mo...
	3064	MOA-RP649	NULL	2	USB 2+2+1 Tusl...

**tblMarka**

	markaKod	Marka
	1	3COM
	2	A4 TECH
	3	ADAPTEC
	4	AIRFORCE
▶	5	ALCATEL
	6	ALPS
	7	ALTEC
	8	AMD
	9	AOPEN
	10	APACHE
	11	ASUS
	12	AVERMEDIA
	13	BENQ
	14	CANON
	15	CASPER
	16	CHTC

## 9.2.3. BİRDEN FAZLA TABLO KULLANMAK (DEVAM...)

Dikkat edilecek olursa her iki tabloda ortak olan değerler **markaKod** bilgisidir. Bu değerleri kullanarak her iki tablodan bilgileri elde edecek olan **VIEW** aşağıdaki gibi olacaktır:



```
SQLQuery11.sql - ...N5LVHQ\ENDER (65))* X DESKTOP-TN5LVHQ.d...- dbo.vw_Urunler*

-- CREATE VIEW Urunler
AS
SELECT U.urunKod, U.urunAd, M.Marka
FROM   tblUrun U JOIN
       tblMarka M ON M.markaKod = U.markaKod

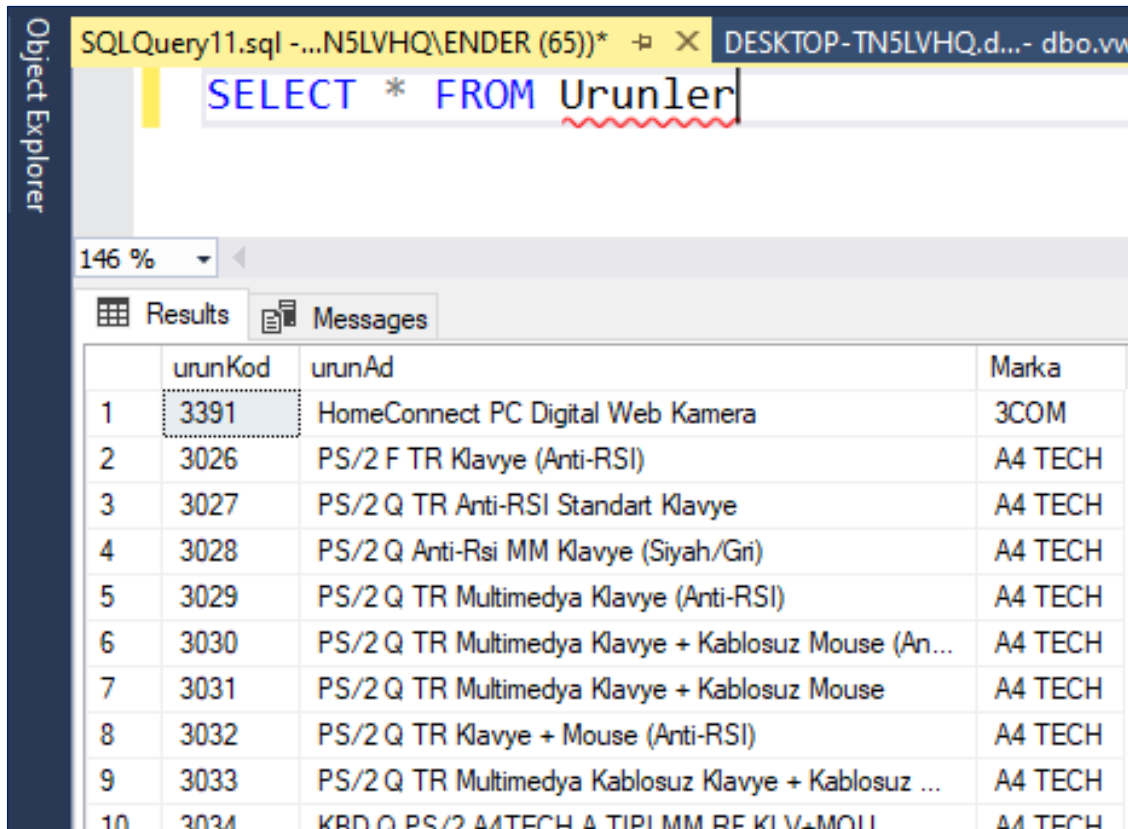
146 %

Messages
Commands completed successfully.

Completion time: 2024-04-28T21:52:09.6530063+03:00
```

## 9.2.3. BİRDEN FAZLA TABLO KULLANMAK (DEVAM...)

Bu VIEW çalıştırıldığında her iki tabloda yer alan veriler birleştirilerek ekrana gösterilecektir.



The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays a SQL query: `SELECT * FROM Urunler`. The bottom pane shows the results of the query in a table format. The table has four columns: `urunKod`, `urunAd`, and `Marka`. The results are listed in 10 rows, with the first row highlighted.

	urunKod	urunAd	Marka
1	3391	HomeConnect PC Digital Web Kamera	3COM
2	3026	PS/2 F TR Klavye (Anti-RSI)	A4 TECH
3	3027	PS/2 Q TR Anti-RSI Standart Klavye	A4 TECH
4	3028	PS/2 Q Anti-Rsi MM Klavye (Siyah/Gri)	A4 TECH
5	3029	PS/2 Q TR Multimedya Klavye (Anti-RSI)	A4 TECH
6	3030	PS/2 Q TR Multimedya Klavye + Kablosuz Mouse (An...	A4 TECH
7	3031	PS/2 Q TR Multimedya Klavye + Kablosuz Mouse	A4 TECH
8	3032	PS/2 Q TR Klavye + Mouse (Anti-RSI)	A4 TECH
9	3033	PS/2 Q TR Multimedya Kablosuz Klavye + Kablosuz ...	A4 TECH
10	3034	KBD Q PS/2 A4TECH A TIPI MM BE KI V+MOU	A4 TECH

## 9.2.4. VIEW' LARDA DEĞİŞİKLİK YAPMAK

---

Bir View üzerinde değişiklik yapılmak istenildiğinde ALTER VIEW komutu kullanılır. ALTER VIEW deyiminin genel yapısı aşağıdaki gibidir.

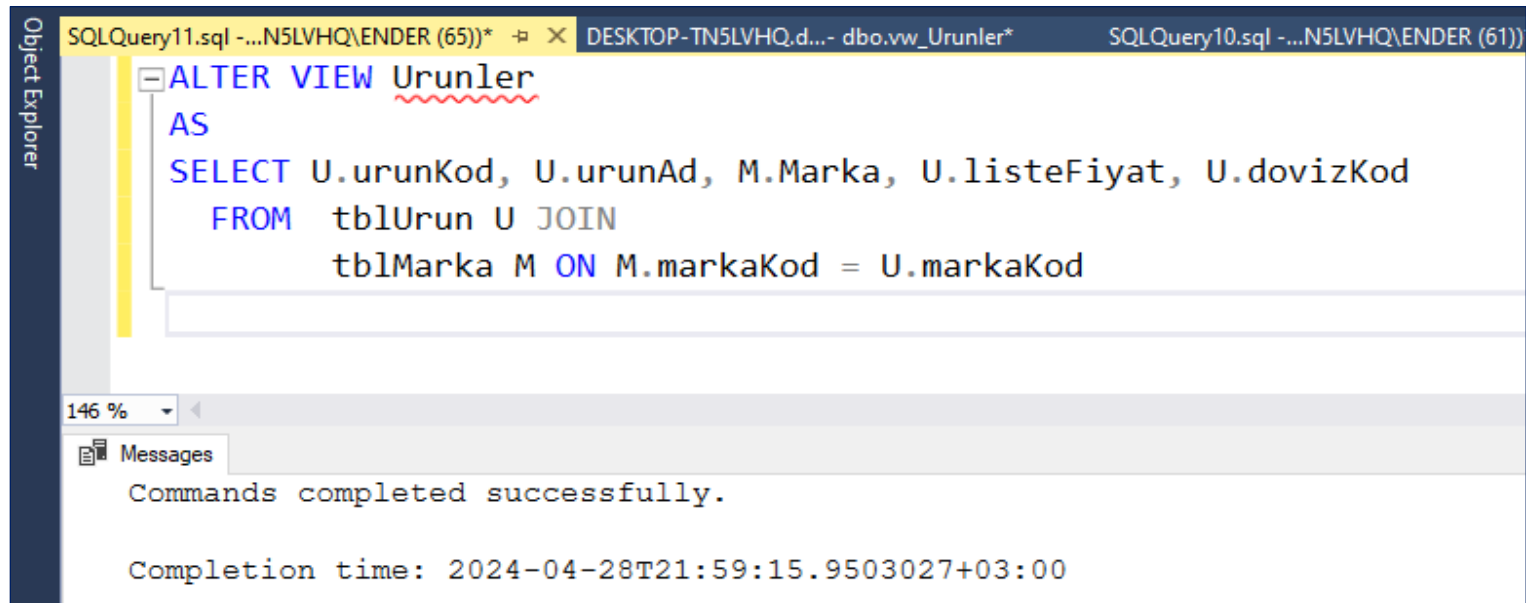
**ALTER VIEW** **view\_adi**

**AS**

**SELECT ifadesi**

## 9.2.4. VIEW' LARDA DEĞİŞİKLİK YAPMAK (DEVAM...)

- Oluşturmuş olduğumuz View üzerinde listefiyat bilgisini de ekleyecek bir değişiklik yapalım. Bu durumda değişiklik yapacak SQL ifadesi aşağıdaki gibi olacaktır.



The screenshot shows the SQL Server Enterprise Manager interface. The main window displays the following SQL query:

```
ALTER VIEW Urunler
AS
SELECT U.urunKod, U.urunAd, M.Marka, U.listeFiyat, U.dovizKod
FROM   tblUrun U JOIN
       tblMarka M ON M.markaKod = U.markaKod
```

Below the query, the Messages pane shows the following output:

```
Commands completed successfully.

Completion time: 2024-04-28T21:59:15.9503027+03:00
```

- Çalıştırıldığında listefiyat ve dovizkod bilgisi de View'e eklenir.

## 9.2.5. VIEW' LARI SİLMEK

---

- Bir View'ı veritabanından kaldırmak(silmek) için **DROP komutu** kullanılmaktadır. Bu komut kullanıldığında view sistemden tamamen silinmiş olacaktır. **DROP komutunun genel kullanımı** aşağıdaki gibidir.

**DROP VIEW** **view\_adı**

- Örnek:

**DROP VIEW** Urunler

## 9.2.6. VIEW' LARI ŞİFRELEMEK

---

Bazı durumlarda View'ları oluşturan **kaynak kodların** **başkaları tarafından görülmemesi** istenebilir. Böyle bir durumda SQL Server'ın kendi içerisinde yer alan **bir şifreleme mekanizması** bu yapıları kullanıcı için şifreler. Burada dikkat edilmesi gereken en önemli nokta **şifrelenmiş bir View'ın içeriğini** biz dahil hiç kimsenin görememesidir. Bu nedenle şifreleme işlemi uygulanmadan önce kaynak kodların bir kopyasının mutlaka saklanması önerilir.

## 9.2.6. VIEW' LARI ŞİFRELEMEK (DEVAM...)

---

View kaynak kodları şifrelemek için **ENCRYPTION** komutu kullanılmaktadır.

ENCRYPTION komutu View oluşturulurken kullanılacağı gibi sonradan da kullanılabilir. Bu komutun genel yapısı şöyledir.

**CREATE VIEW** **view\_adı**

**WITH ENCRYPTION**

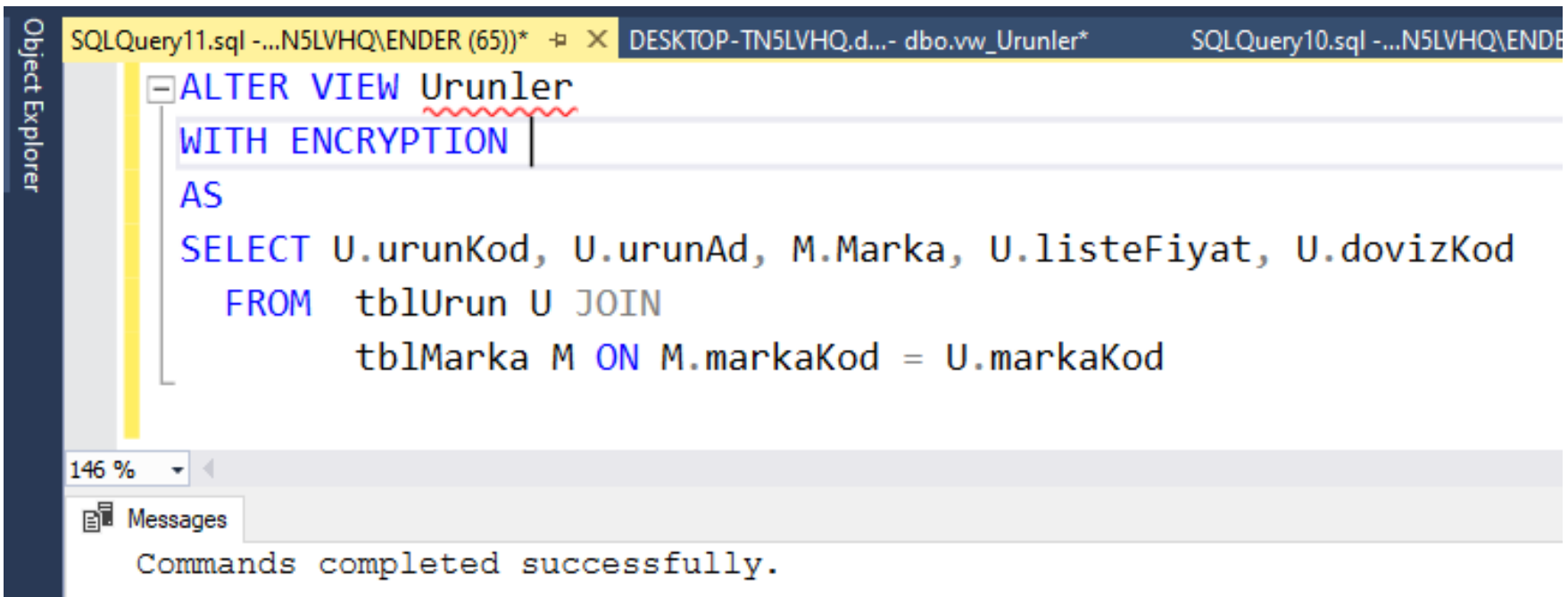
**AS**

**SELECT** **cümlesi**



## 9.2.6. VIEW' LARI ŞİFRELEMEK (DEVAM...)

**Örnek:** Daha önceden oluşturmuş olduğumuz Urunler View'ını şifreli hale getirelim.



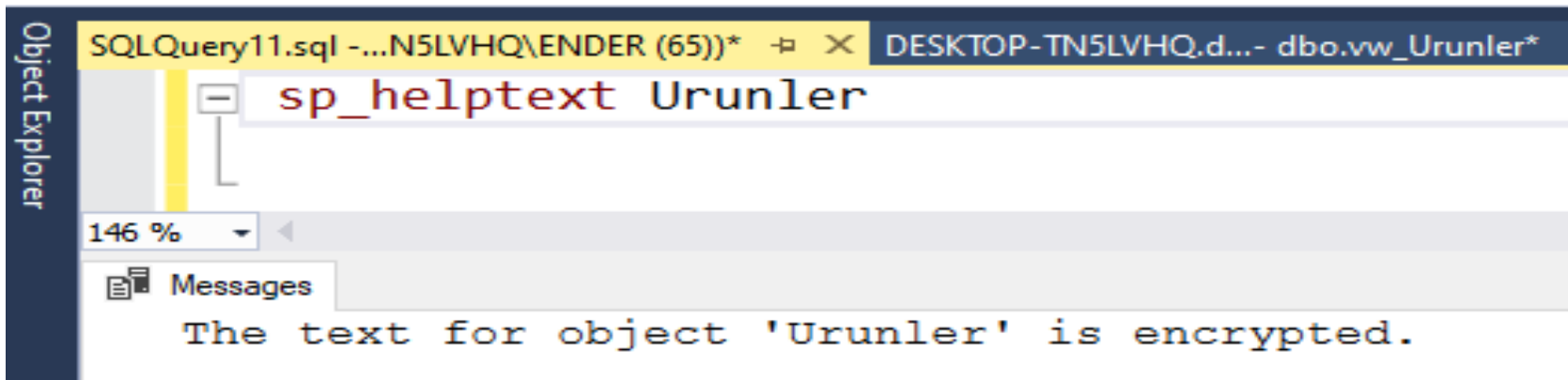
The screenshot shows the SQL Server Enterprise Manager interface. The 'Object Explorer' on the left shows the 'Urunler' view under the 'dbo' schema. The 'SQLQuery11.sql' window displays the following T-SQL code:

```
ALTER VIEW Urunler
WITH ENCRYPTION
AS
SELECT U.urunKod, U.urunAd, M.Marka, U.listeFiyat, U.dovizKod
FROM tblUrun U JOIN
tblMarka M ON M.markaKod = U.markaKod
```

The 'Messages' pane at the bottom shows the message: 'Commands completed successfully.'

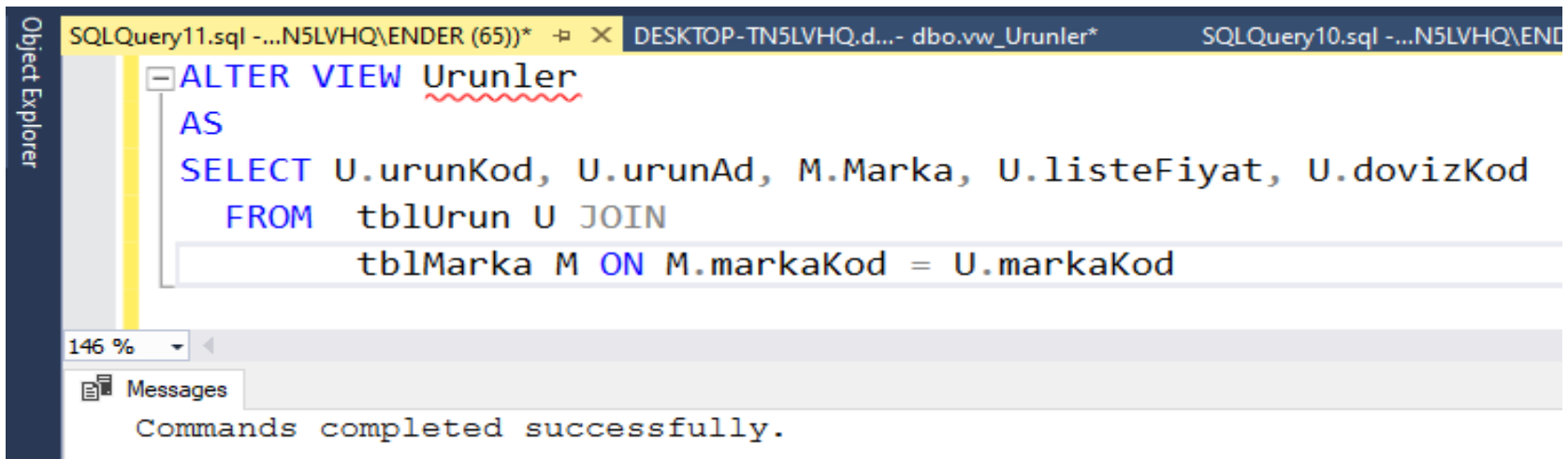
## 9.2.6. ŞİFRELENMİŞ VIEW KAYNAK KODU GÖRME

Burada dikkat edilmesi gereken bir diğer noktada, View çalıştırılabilir ancak kodları görüntülenemez. Kaynak kodunu görmek için aşağıdaki sistem prosedürü çalıştırıldığında ilgili View kaynak kodun şifrelendiğini belirten bir mesaj görüntülenir.



## 9.2.6. VIEW KOD ŞİFRESİNİ KALDIRMAK

Şifreyi kaldırmak isterseniz View üzerinde yine bir değişiklik yaparak bunu yapabilirsiniz. Bunun için **WITH ENCRYPTION** komut satırını kaldırarak View'ı yeniden düzenleyebilirsiniz.



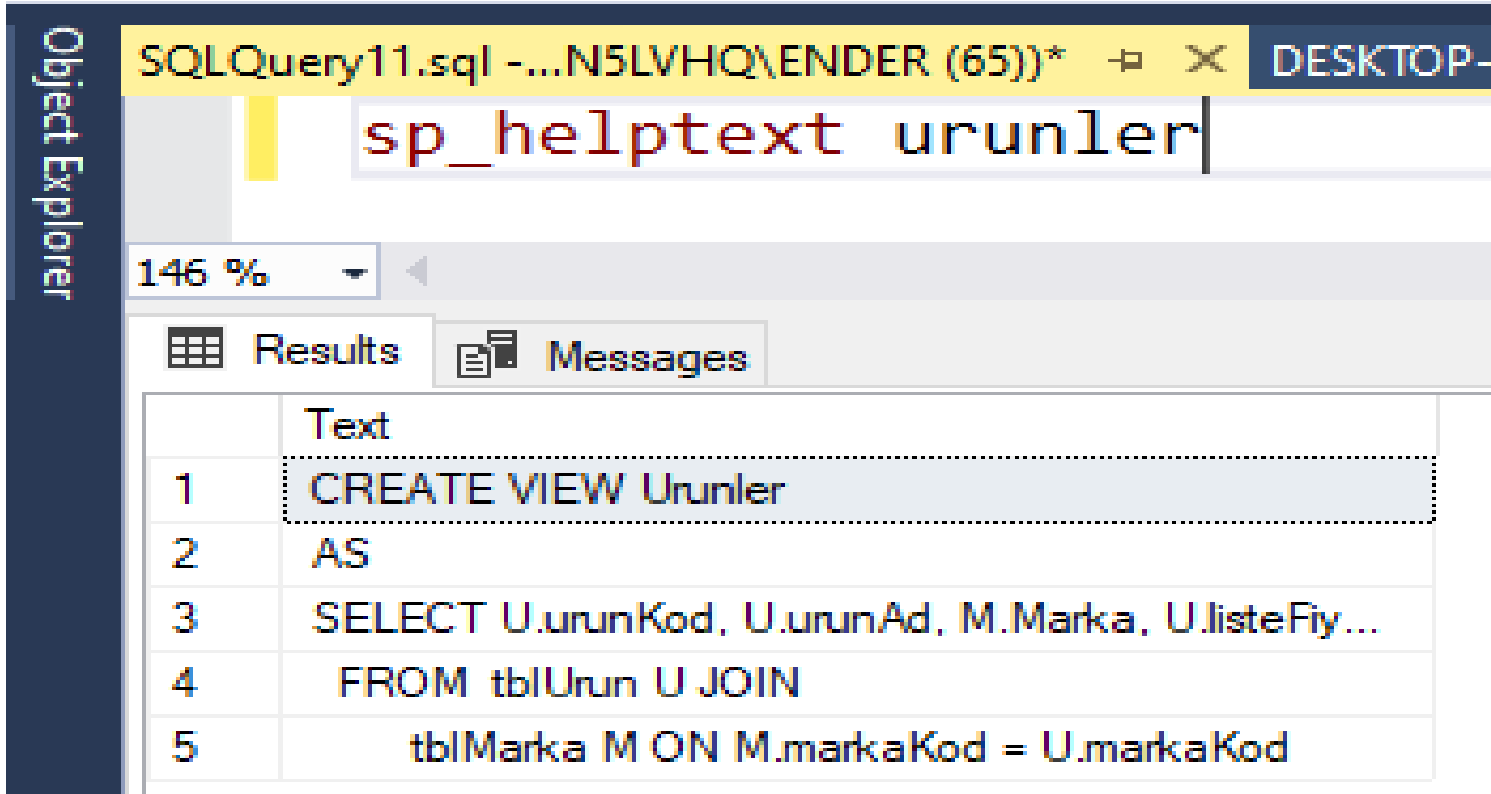
The screenshot shows the SQL Server Enterprise Manager interface. The 'Object Explorer' on the left shows the 'Urunler' view. The 'SQL Query' window displays the following SQL code:

```
ALTER VIEW Urunler
AS
SELECT U.urunKod, U.urunAd, M.Marka, U.listeFiyat, U.dovizKod
FROM tblUrun U JOIN
tblMarka M ON M.markaKod = U.markaKod
```

The 'Messages' pane at the bottom shows the message: 'Commands completed successfully.'

## 9.2.6. VIEW KAYNAK KODU GÖRME

Kaynak kodunu görmek için `sp_helptext` komutu yeniden çalıştırıldığında;



The screenshot shows the SQL Server Enterprise Manager interface. On the left is the 'Object Explorer' pane. The main window displays a query window titled 'SQLQuery11.sql'. The query text is `sp_helptext urunler`. Below the query window, the 'Results' tab is active, showing the source code of the view 'urunler'. The code is as follows:

	Text
1	CREATE VIEW Urunler
2	AS
3	SELECT U.urunKod, U.urunAd, M.Marka, U.listeFiy...
4	FROM tblUrun U JOIN
5	tblMarka M ON M.markaKod = U.markaKod

## 9.2.6. VIEW 'LERİN AVANTAJLARI

---

- **Görünümler** belirli bilgiye ulaşmayı sağlarken belirli bir kısım bilgiye ulaşmayı da engelleyerek daha kısıtlı bir tablo yaratmış olurlar.
- **Kullanıcı** yeni bir **görünüm oluştururken** daha kolay kullanabilmek için **alanlara istediği isimleri verebilir** ve böylece varsa karmaşık ve kullanımı zor alan isimlerinden de kurtulmuş olur.

## 9.2.6. VIEW 'LERİN AVANTAJLARI

---

- Veritabanı sahibi kişi bir tabloda **sadece belli bazı alanlar için istediği kişilere sorgulama hakkı vermek yerine** o tabloyu temel tablo kabul ederek bir **görünüm oluşturabilir** ve o **görünümde istediklerine sorgulama hakkı verebilir**. Böylece hem temel tabloda herhangi bir değişikliğin yapılması önlenmiş olur; hem de sorgulama hakkına sahip kişilerin kesintisiz bir şekilde görünüm üzerinde sorgulama yapmaları sağlanır.
- İki ya da daha fazla tablonun "join" ile birleştirilmesi sonucu görünüm oluşturulabilir ve daha ileriki uygulamalarda kullanılabilecek **bir nesne elde edilir**.

# *TEŞEKKÜRLER...*

## *SORULARINIZ ?*

---