

### Veritabanı ve Elektronik Tablo Arasındaki Farklar Nelerdir?

Hem veritabanları hem elektronik tablolar (ör. Microsoft Excel) bilgi depolamak için uygun yöntemlerdir. Aralarındaki temel farklar şu şekilde sıralanabilir:

- Verileri toplama ve manipüle etme yöntemi
- Verilere erişebilen kişiler
- Depolanabilecek veri miktarı

### Veritabanı Türleri

1. İlişkisel Veritabanı
2. Nesne Odaklı Veritabanı
3. Dağıtılmış Veritabanı (HİZ – TERÖR – DOĞAL AFET)
4. Veri Ambarı
5. NoSQL Veritabanı
6. Grafik Veritabanı
7. OLTP Veritabanı

### Yeni Veritabanı Türleri

#### 1) Açık Kaynak Veritabanları

Kaynak kodu açık kaynak olan bir sistemdir. Bu tür veritabanları SQL veya NoSQL veritabanları olabilir.

#### 2) Bulut Veritabanları

İki tür bulut veritabanı modeli bulunur: Geleneksel ve servis olarak veritabanı (DBaaS). DBaaS sayesinde yönetim görevleri ve bakım işlemleri servis sağlayıcı tarafından gerçekleştirilir.

#### 3) Çoklu Model Veritabanı

#### 4) Belge/JSON Veritabanı

Belge odaklı bilgilerin depolanması, alınması ve yönetilmesi için tasarlanan belge veritabanları, verileri satırlar ve sütunlar yerine JSON biçiminde depolamak için modern bir yöntem sunar.

#### 5) Kendi Kendini Yöneten Veritabanları

Veritabanı yöneticileri tarafından gerçekleştirilen ince ayar, güvenlik, yedekleme, güncelleme ve diğer rutin yönetim görevlerini otomatikleştirmek üzere makine öğreniminden yararlanan bulut tabanlı çözümlerdir.

- **SQL**, verileri sorgulamak, manipüle etmek, tanımlamak ve aynı zamanda erişim kontrolü sağlamak üzere neredeyse tüm ilişkisel veritabanlarında kullanılan bir programlama dilidir.
- **Manipüle etmek:** Bilgilerin bir amaç doğrultusunda yönlendirilmesidir.
- SQL, yapılandırılmış sorgu dilidir. Yapılandırılmış derken ne demek istiyoruz? Örneğin TC kimlik numarası için konuşacak olursak, TC kimlik numarası sadece rakam içerir harf içermez.
- Tutulan her türlü veriye **varlık** denir.

- **Sütunlar** bize veritabanına eklenen özellikleri, **satırlar** ise eklenen verileri verir.
- Komutlar, tablo isimleri vs büyük-küçük harfe duyarlı değildir.
- String verileri tek tırnak ( ' ' ) içerisine alınır.

- ✓ Veri tabanımızı oluşturmamızın sonucunda standart olarak **MDF** ve **LDF** adlı iki dosya oluşturulmaktadır.
- ✓ **MDF dosyasını**, veri dosyamız olarak adlandırabiliriz. Veri tabanımız üzerinde bulunan tablo, sp ve view gibi tüm sistem bu dosya üzerinden tutulmaktadır.
- ✓ **LDF dosyası**, yapılan işlemlerin tutulduğu dosyadır. Burada kayıt tutulmaz, veri tabanında yapılan sorgularımızın tutulması sağlanır.
- ✓ **NDF dosyası**, ikincil veri tabanı dosyasıdır. Özellikle büyük işletmelerde veriyi bölmek için kullanılır.

| Data Definiton Language | Data Query Language | Data Manipulation Language | Data Control Language |
|-------------------------|---------------------|----------------------------|-----------------------|
| CREATE                  | SELECT              | INSERT                     | GRANT                 |
| ALTER                   |                     | UPDATE                     | REVOKE                |
| DROP                    |                     | DELETE                     |                       |
| TRUNCATE                |                     | MERGE                      |                       |
| RENAME                  |                     | CALL                       |                       |
| COMMENT                 |                     | EXPLAIN PLAN               |                       |
|                         |                     | LOCK TABLE                 |                       |

## VERİ TİPLERİ

Sabit uzunluklu veriler için **CHAR**, **NCHAR**, **VARCHAR** ve **NVARCHAR** değişkenleri kullanılır.

Örneğin telefon numarası, TC kimlik numarası gibi.

isim **CHAR(10)**

Herbir karakteri 1 byte olan ve en fazla 10 karakter içerebilen metinsel bir ifadeyi tutabilir.

isim **NCHAR(10)**

Her bir karakteri 2 byte olan ve en fazla 10 karakter içerebilen metinsel bir ifadeyi tutabilir. Ayrıca uluslar arası kodlar (UNICODE) içeren metinsel ifadeler tutulabilir. CHAR ile NCHAR arasındaki fark budur.

isim **CHAR(10)** -----> Data, 'DENEME ' şeklinde tutulur ve 10 byte yer kaplar.

isim **NCHAR(10)** -----> Data, 'DENEME ' şeklinde tutulur ve 20 byte yer kaplar.

isim **VARCHAR(10)** ---> Data, 'DENEME' şeklinde tutulur ve 6 byte yer kaplar. (DENEME 6 karakterden oluşuyor. VAR yazdığımızdan dolayı geriye kalan 4 karakter CHAR tipindeki gibi boşlukla tamamlanmak yerine, görmezden geliniyor. Bu sayede veri gereğinden fazla yer kaplamamış oluyor.)

isim **NVARCHAR(10)** --> Data 'DENEME' şeklinde tutulur ve 12 byte yer kaplar.

**Date:** Yıl, ay ve gün tutar.

**Datetime:** Yıl, ay, gün, saat, dakika ve saniyeyi tutar.

### 1. DDL (Data Definition Language)

*Verritabanı üzerinde nesne (tablo gibi) oluşturmak, silmek vb. için kullanılırlar.*

**CREATE, ALTER, DROP, TRUNCATE** kullanılır.

### 2. DML (Data Manuplation Language)

*Veritabanı içindeki verilere ulaşmak ve değiştirmekle ilgili SQL deyimleridir.*

**INSERT, UPDATE, DELETE** kullanılır.

### 3. DCL (Data Control Language)

*Veritabanındaki kullanıcı haklarını düzenlemek için kullanılan deyimlerdir.*

**GRANT, DENY, REVOKE** kullanılır

## PRIMARY KEY (BİRİNCİL ANAHTAR)

Bir kolondaki bir değerin sadece bir kez girilmesini sağlayan yapıya **Primary Key (Birincil Anahtar)** denir.

```
Create table tblPers(  
    TCNo varchar(11) primary key,  
    Ad varchar(40),  
    Soyad varchar(50)  
);
```

```
insert into tblPers values ('12345678900', 'Ahmet', 'Efendi'),  
( '12345678901', 'Ahmet', 'Efendi')
```

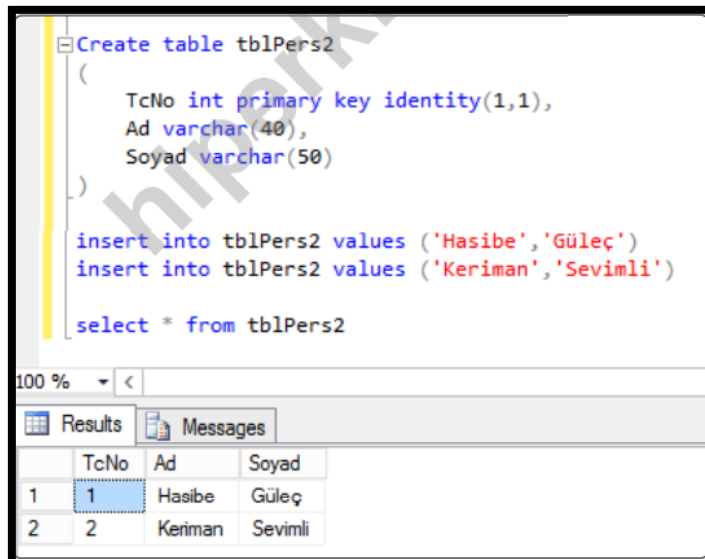
Burada *TCNo* değerlerinin farklı, *Ad* ve *Soyad* kolonlarının aynı olduğunu görebiliriz. Fakat *TCNo* değerleri farklı olduğu için iki ayrı kişi olduğu kavranmaktadır.



*TCNo* kolonu *Primary Key* olduğundan dolayı, **aynı değere sahip ikinci bir kayıt giremeyiz**. Tablolarımızda bu sebepten dolayı *primary key* kullanmamız gerekmektedir. **Primary key** kolonuna otomatik olarak sayı atayabiliriz. Böylece her kayıt için benzersiz bir değer atanacaktır.

Otomatik bir şekilde sayı arttırmak için **IDENTITY(başlangıç\_degeri, artis\_miktari)** ifadesini kullanacağız. Bunun için *TCNo* kolonunun veri tipinin **int** olduğuna dikkat ediniz.

```
Create table tblPers2(  
    TCNo int primary key identity(1,1),  
    Ad varchar(40),  
    Soyad varchar(50)  
);  
insert into tblPers2 values ('Hasibe', 'Güleç')  
insert into tblPers2 values ('Keriman', 'Sevimli')
```



## DDL KOMUTLARI (Data Definition Language)

### 1- CREATE

Kullanıcının seçimine göre veritabanı ya da tablo oluşturur.

```
CREATE DATABASE OKUL          -- OKUL isminde bir veritabanı oluşturur
CREATE TABLE ogrenci (
    siraNumarasi INT(11) AUTO_INCREMENT PRIMARY KEY,
    isim VARCHAR(50) NOT NULL,
    soyisim VARCHAR(50) NOT NULL,
    bolum VARCHAR(50)
)
```

**AÇIKLAMA:** siraNumarasi INT(11) UNSIGNED AUTO\_INCREMENT PRIMARY KEY,  
siraNumarasi: -- Değişken ismi,  
INT(11) -- 11 karakter genişliğinde tam sayı  
AUTO\_INCREMENT: -- Eklenen her veride "siraNumarasi" değeri otomatik olarak birer birer artacak.  
PRIMARY KEY -- "siraNumarasi" nin benzersiz alan olacağını belirtiyor.

**AÇIKLAMA:** isim VARCHAR(50) NOT NULL,  
isim: -- Alan ismi,  
VARCHAR(50): -- 50 karakter genişliğinde metin  
NOT NULL : -- Veri eklemekten ilerlenemeyeceğini, boş bırakılamayacağını ifade ediyor.

### 2- USE

İşlem yapılmak istenen veritabanını seçmek için kullanılır. Kullanım şekli; **USE veritabanı\_ismi**

### 3- ALTER

Daha önceden oluşturulmuş olan veritabanı tabloları üzerinde değişiklik yapmak için kullanılır. (Sütun ekleme, çıkarma gibi)

```
ALTER TABLE ogrenci RENAME ogrenciler;
-- Tablo adı değiştirme

ALTER TABLE tabloadi ADD (fakulte VARCHAR(50), kayit_tarihi DATE);
-- Tabloya yeni alan ekleme

ALTER TABLE ogrenciler ALTER sınıf VARCHAR(3);
-- ogrenciler tablosundaki int tipindeki sınıf alanını 3 karakterlik VARCHAR'a dönüştürme.

ALTER TABLE ogrenciler DROP COLUMN fakulte, DROP COLUMN sınıf;
-- ogrenciler tablosundaki "fakulte" ve "sınıf" alanları silme
```

### 4- DROP

**Veritabanını** ya da **tabloyu** silmek için kullanılan komuttur.

**NOT:** DROP komutu ile veritabanını sildiğimizde içerisinde yer alan tüm verileri kalıcı olarak kaybedeceğimizi unutmamalıyız.

```
DROP DATABASE okul -- okul adlı veritabanını sil
DROP TABLE sınıf -- sınıf adlı tabloyu veritabanından kaldı
```

### 5- TRUNCATE

Tablonun içindeki verileri siler, ancak tablonun kendisini silmez.

## DML KOMUTLARI (Data Manipulation Language)

### 1- INSERT INTO

Veritabanına kayıt eklemek için kullanılır.

```
INSERT INTO personel (ID, isim, bolum, maas)
VALUES (65, 'Metin Yıldız', 'Reklam', 1350)
```

Eğer tablonun tüm alanlarına bilgi girilecekse, bu durumda alan adlarını yazmamıza gerek yoktur. Tabloda NULL olarak bırakılacak alanlar varsa yani her alana bilgi girişi yapmak istemiyorsak, sadece istenilen alanların adı yazılarak veri girişi yapılabilir.

```
INSERT INTO personel(ID, isim, bolum) VALUES (5, 'Ahmet Demirci', 'reyon')
```

### 2- UPDATE

Veritabanında bulunan verileri değiştirmek (güncellemek) amacıyla kullanılan bir deyimdir.

**UPDATE** [tablo\_adi] **SET** [yeni\_bilgiler] **WHERE** [şartlar]

- Bu formata göre **UPDATE** deyiminden sonra hangi veritabanı tablosunda güncelleme yapmak istiyorsak o tablonun adını yazıyoruz.
- **SET** deyiminden sonra değiştirmek istediğimiz bilgileri giriyoruz.
- **WHERE** ifadesinden sonra değiştirme işlemi yapacağımız kayıtlarla ilgili şartı veya şartları yazıyoruz. **WHERE** ifadesinin kullanımı zorunlu değildir, fakat WHERE kullanılmazsa bütün kayıtlar değiştirme işleminden etkilenecektir.

❖ Birden fazla kolonun değerini aynı sorgu ile değiştirebiliriz. Bunun için set ifadesinde kolonlar arasındaki değer atamalarını virgül ile yazmamız yeterlidir.

```
update tblPersonel set Ad='Keriman', Yas=45,Ulke='Hollanda' where ID=7
```

❖ **UPDATE** personel **SET** bolum='reyon'

Yukardaki komut çalıştırıldığında tüm çalışanların bölüm bilgisi reyon olarak değiştirilir.

Eğer sadece bir meslek grubu üzerinde işlem yapacaksak o zaman kullanılacak SQL cümlecigi;

```
UPDATE personel SET bolum='reyon' WHERE bolum='bilgi işlem'
```

### 3- DELETE

Veritabanında bulunan bir tablodaki kayıtları silmek amacıyla kullanılan bir SQL komutudur.

**DELETE FROM** [tablo adı] **WHERE** [şartlar]

- Bu formata göre **DELETE FROM** deyiminden sonra hangi veritabanı tablosunda alan silmek istiyorsak o tablonun adını yazıyoruz.

```
DELETE FROM personel
```

Yukardaki komut çalıştırıldığında bütün personele ait bilgiler silinmektedir.

Eğer sadece bir meslek grubu üzerinde işlem yapacaksak o zaman kullanılacak SQL cümlecigi;

```
DELETE FROM personel WHERE bolum='bilgi işlem'
```

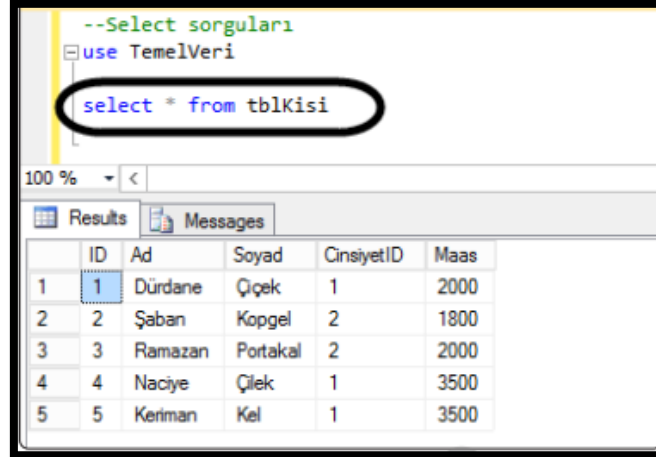
## DQL KOMUTLARI (Data Query Language)

### 1- SELECT

Tablolarımızda bulunan verilerin istediğimiz kriterler doğrultusunda gelmesini sağlar.

```
select * from tblKisi
```

\* tblKisi tablosundaki tüm sütunları (özellikleri) getirmek için kullanılır.



The screenshot shows a SQL query window with the following content:

```
--Select sorguları
use TemelVeri

select * from tblKisi
```

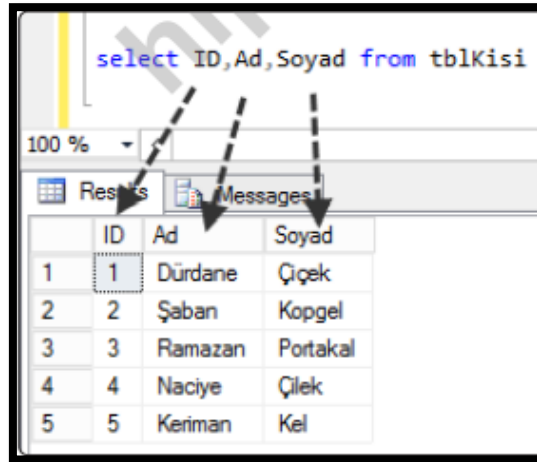
Below the query window, the 'Results' tab is active, displaying the following table:

|   | ID | Ad      | Soyad    | CinsiyetID | Maas |
|---|----|---------|----------|------------|------|
| 1 | 1  | Dürdane | Çiçek    | 1          | 2000 |
| 2 | 2  | Şaban   | Kopgel   | 2          | 1800 |
| 3 | 3  | Ramazan | Portakal | 2          | 2000 |
| 4 | 4  | Naciye  | Çilek    | 1          | 3500 |
| 5 | 5  | Keriman | Kel      | 1          | 3500 |

**Sonuç kümesi** yukarıdaki gibidir. Bir sorgudan dönen kayıtlar her zaman bir tablo şeklinde olurlar ve bu tabloya **sonuç tablosu (result set)** denir.

tblKisi tablosunda yer alan sütunlardan (özelliklerden) istediğimizi getirmek için:

```
select ID, Ad, Soyad from tblKisi
```



The screenshot shows a SQL query window with the following content:

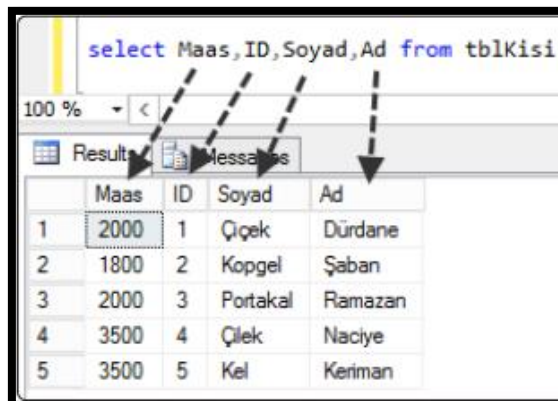
```
select ID,Ad,Soyad from tblKisi
```

Below the query window, the 'Results' tab is active, displaying the following table:

|   | ID | Ad      | Soyad    |
|---|----|---------|----------|
| 1 | 1  | Dürdane | Çiçek    |
| 2 | 2  | Şaban   | Kopgel   |
| 3 | 3  | Ramazan | Portakal |
| 4 | 4  | Naciye  | Çilek    |
| 5 | 5  | Keriman | Kel      |

Tabloda yer alan sütunları (özellikleri) istediğimiz sırada getirmek için:

```
select Maas, ID, Soyad, Ad from tblKisi
```



The screenshot shows a SQL query window with the following content:

```
select Maas,ID,Soyad,Ad from tblKisi
```

Below the query window, the 'Results' tab is active, displaying the following table:

|   | Maas | ID | Soyad    | Ad      |
|---|------|----|----------|---------|
| 1 | 2000 | 1  | Çiçek    | Dürdane |
| 2 | 1800 | 2  | Kopgel   | Şaban   |
| 3 | 2000 | 3  | Portakal | Ramazan |
| 4 | 3500 | 4  | Çilek    | Naciye  |
| 5 | 3500 | 5  | Kel      | Keriman |

## WHERE Koşul Yapısı

`Select * from tblKisi Where Ad= 'Ramazan'`

tblKisi tablosunda Ad sütununda (özellğinde) **Ramazan** olanları getirildi.

```
select * from tblKisi Where Ad='Ramazan'
```

100 % <

Results Messages

|   | ID | Ad      | Soyad    | CinsiyetID | Maas |
|---|----|---------|----------|------------|------|
| 1 | 3  | Ramazan | Portakal | 2          | 2000 |

## AND – OR İşleçleri

Birden fazla koşul istediğimiz durumda **and (ve), or (veya)** kullanılır. And ifadesinde tüm durumların karşılanması (doğru olması) gerekmektedir.

tblKisi tablosunda Ad kolonunda **Dürdane**, Soyad kolonunda **Çiçek** olması durumunda verilerimiz gelecektir.

`select * from tblKisi Where Ad='Dürdane' and Soyad='Çiçek'`

```
select * from tblKisi Where Ad='Dürdane' and Soyad='Çiçek'
```

100 % ▾ <

Results Messages

|   | ID | Ad      | Soyad | CinsiyetID | Maas |
|---|----|---------|-------|------------|------|
| 1 | 1  | Dürdane | Çiçek | 1          | 2000 |

Or ifadesinde ise durumların herhangi birini karşılaması yeterlidir.

`select * from tblKisi Where Ad='Dürdane' or Soyad='Portakal'`

Sorgumuzda Ad kolonunda **Dürdane** olanlar ya da Soyad kolonunda **Portakal** olanları getirecektir.

Where sorgularımızda aynı anda **And** ve **Or** ifadeleri kullanılabilir.

`select * from tblKisi Where Ad='Dürdane' and Soyad='Çiçek' or Maas=1800`

```
select * from tblKisi Where Ad='Dürdane' and Soyad='Çiçek' or Maas=1800
```

100 % < |||

Results Messages

|   | ID | Ad      | Soyad | CinsiyetID | Maas |
|---|----|---------|-------|------------|------|
| 1 | 1  | Dürdane | Çiçek | 1          | 2000 |
| 2 | 2  | Şaban   | Kögel | 2          | 1800 |

Sorgumuzda Ad kolonunda **Dürdane**, Soyad kolonunda **Çiçek** olacak, bu iki durumun ikisi de sağlanması gerekirken Maas kolonunda **1800** yazan veriler gelecektir.

**NOT I:** where isim <'ALP' -----> Adı alfabetik olarak ALP kelimesinden önce gelenler getirilir.

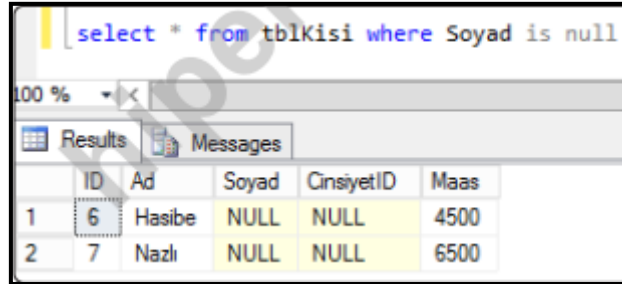
**NOT II:** Yaşı 30'dan büyük veya Ankara'da oturan ve maaşı 250'den büyük veya kadın olanlar:

`select*from personel where (yas>30 and kent='Ankara') and (maas>250 or cinsiyet='K')`

## IS NULL – IS NOT NULL

**Soyad** değeri **boş (null)** olanları getirmek için **is null** parametresi kullanılır.

```
select * from tblKisi where Soyad is null
```



The screenshot shows a SQL query window with the query `select * from tblKisi where Soyad is null`. Below the query, the 'Results' tab is active, displaying a table with two rows. The first row has ID 6, Ad 'Hasibe', Soyad 'NULL', CinsiyetID 'NULL', and Maas 4500. The second row has ID 7, Ad 'Nazlı', Soyad 'NULL', CinsiyetID 'NULL', and Maas 6500.

|   | ID | Ad     | Soyad | CinsiyetID | Maas |
|---|----|--------|-------|------------|------|
| 1 | 6  | Hasibe | NULL  | NULL       | 4500 |
| 2 | 7  | Nazlı  | NULL  | NULL       | 6500 |

Boş olmayanları getirmek için **is not null** parametresi kullanılmalıdır.

## OPERATÖRLER

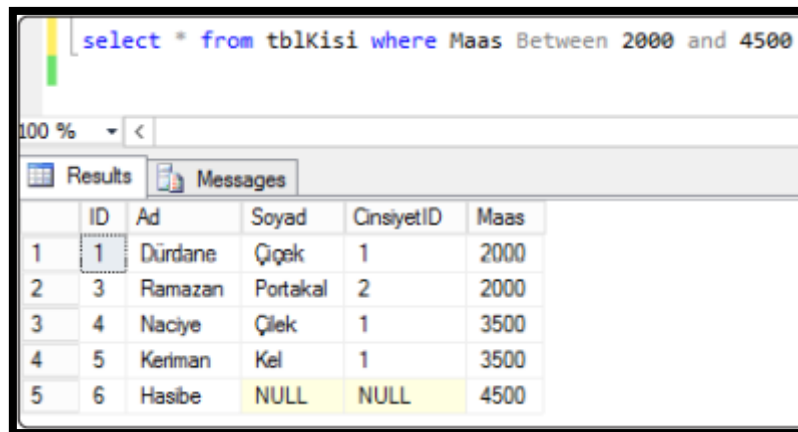
|  |   |
|--|---|
| <code>select * from tblKisi where Ad = 'Şaban'</code>  | <b>Ad</b> sütununda <b>Şaban</b> olanları getir.            |
| <code>select * from tblKisi where Ad != 'Şaban'</code> | <b>Ad</b> sütununda <b>Şaban</b> olmayanları getir.         |
| <code>select * from tblKisi where Maas != 3000</code>  | <b>Maas</b> değeri <b>3000'den</b> küçük olmayanları getir. |
| <code>select * from tblKisi where Maas != 3000</code>  | <b>Maas</b> değeri <b>3000'den</b> büyük olmayanları getir. |

## BETWEEN

Belirtilen iki değer arasındaki değerlere sahip verileri getirir.

**Maas** sütununa göre **aralık sorgusu** çekelim. 2000 ile 4500 (2000 ve 4500 dâhil) getirelim.

```
select * from tblKisi where Maas Between 2000 and 4500
```



The screenshot shows a SQL query window with the query `select * from tblKisi where Maas Between 2000 and 4500`. Below the query, the 'Results' tab is active, displaying a table with five rows. The first row has ID 1, Ad 'Dürdane', Soyad 'Çiçek', CinsiyetID 1, and Maas 2000. The second row has ID 3, Ad 'Ramazan', Soyad 'Portakal', CinsiyetID 2, and Maas 2000. The third row has ID 4, Ad 'Naciye', Soyad 'Çiçek', CinsiyetID 1, and Maas 3500. The fourth row has ID 5, Ad 'Keriman', Soyad 'Kel', CinsiyetID 1, and Maas 3500. The fifth row has ID 6, Ad 'Hasibe', Soyad 'NULL', CinsiyetID 'NULL', and Maas 4500.

|   | ID | Ad      | Soyad    | CinsiyetID | Maas |
|---|----|---------|----------|------------|------|
| 1 | 1  | Dürdane | Çiçek    | 1          | 2000 |
| 2 | 3  | Ramazan | Portakal | 2          | 2000 |
| 3 | 4  | Naciye  | Çiçek    | 1          | 3500 |
| 4 | 5  | Keriman | Kel      | 1          | 3500 |
| 5 | 6  | Hasibe  | NULL     | NULL       | 4500 |

**Between** sorgusunu hatırlamadığımızda aynı işlemi **operatör** kullanarak da gerçekleştirebiliriz.

```
select * from tblKisi where Maas>=2000 and Maas<=4500
```



## LIKE İşleci

| Like         | Açıklama  |
|--------------|---|
| 'A%'         | İlk karakteri <b>A</b> olanlar gelecektir.                      |
| '%A'         | Son karakteri <b>A</b> olanlar gelecektir.                      |
| %AR%         | Başında, sonunda ya da içerisinde <b>AR</b> olanlar gelecektir. |
| '_a_'        | Üç harfli ortadaki harfi 'a' olanlar.                           |
| 'm_s_n'      | mısın, musun, muson gibi kelimeler.                             |
| 'c[ai]n'     | can ve cin gibi kelimeler.                                      |
| 'er[hckm]an' | Sadece erhan, erkan, ercan, erman isimlerini içerir.            |
| _ (alt tire) | Bir harf veya Rakam karakterini ifade eder.                     |
| '_a'         | Joker karakter  |
| '[AZ]'       | A'dan Z'ye kadarın işleci                                       |
| '[A-K]abc'   | Abc ile biten, baş harfi A'dan K'ye olan tüm kayıtlar           |
| 'A[^B]%'     | A ile başlayan ikinci harfi B olmayan tüm kayıtlar              |

```
--ilk harf ne olursa olsun sonu "el" ile biten soyadlar
select * from tblKisi where Soyad like '_el'

--Ad kolonunda N veya H ile başlayan tüm kayıtlar
select * from tblKisi where Ad like '[NH]%'

--Ad kolonunda A ile biten ilk harfi A'dan Y'ye kadar olan kayıtlar
select * from tblKisi where Ad like '[A-Y]a%'

-- İsmi içerisinde 'er' ifadesi geçmeyen üyeleri bulur.
select * from üyeler where isim not like '%er%'

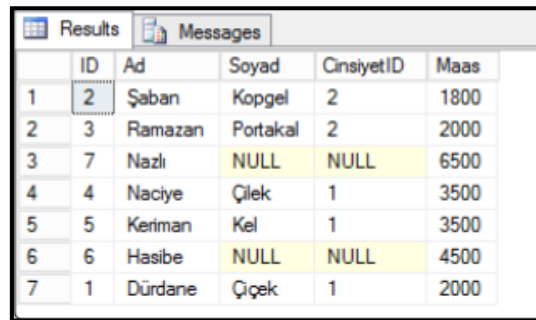
select * from tblKisi Where Ad like 'K%'      --ilk harfi K olanları getir
select * from tblKisi Where Ad like 'E%'      --Ad kolonunda son harfi E olanları getir
select * from tblKisi Where Ad like '%E%'     --Ad kolonunda içerisinde E olanları getir
```

## ORDER-BY

- SELECT sorgusu ile çektiğimiz kayıtları **sıralamak için** kullanırız. **ORDER BY** kelimesi her zaman **sorgunun sonuna gelir**.
- Sıralamanın artan şekilde (**sayılar için küçükten büyüğe, metinsel ifadeler için A'dan Z'ye**) olması için **ASC** yazarız bu aynı zamanda default (varsayılan değer) değerdir. Sıralamanın azalan şekilde olması için ise **DESC** yazarız.

Ad kolonuna göre **Z'den A'ya doğru** sıralaması yapmak için aşağıdaki kodu yazarız.

```
select * from tblKisi order by AD desc
```



|   | ID | Ad      | Soyad    | Cinsiyet | ID | Maas |
|---|----|---------|----------|----------|----|------|
| 1 | 2  | Şaban   | Koppel   | 2        |    | 1800 |
| 2 | 3  | Ramazan | Portakal | 2        |    | 2000 |
| 3 | 7  | Nazlı   | NULL     | NULL     |    | 6500 |
| 4 | 4  | Naciye  | Çilek    | 1        |    | 3500 |
| 5 | 5  | Keriman | Kel      | 1        |    | 3500 |
| 6 | 6  | Hasibe  | NULL     | NULL     |    | 4500 |
| 7 | 1  | Dürdane | Çiçek    | 1        |    | 2000 |

```
select * from tblKisi order by AD asc
select * from tblKisi order by AD      --ASC yazmak zorunda değiliz.
-- Ad kolonunda aynı değerler var ise soyad'a göre ASC sırala
select * from tblKisi order by Ad asc, Soyad asc
```

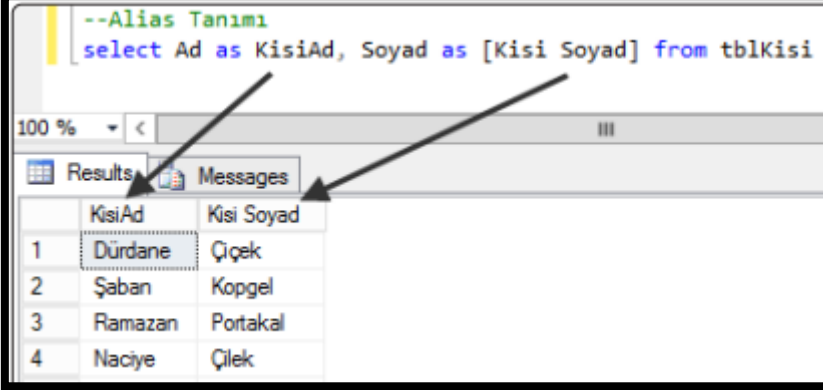
## AS (Takma İsim) İşleci

Tablolara, sütunlara takma isimler vererek daha anlamlı görüntüler elde edebiliriz. **AS** ifadesinden sonra istenilen takma adı verebiliriz.

```
select Ad as KisiAd, Soyad as [Kisi Soyad] from tblKisi
```

**Ad** sütununu **KisiAd** olarak, **Soyad** sütununu ise **Kisi Soyad** olarak değiştirdik. Bu değişimin sadece görüntü olduğunu, tablodaki kolonun değerinin değişmediğini gösterir.

Köşeli parantez [ ] kullanmamızın sebebi boşluklu isim tanımladığımız için.



```
--Alias Tanımı
select Ad as KisiAd, Soyad as [Kisi Soyad] from tblKisi
```

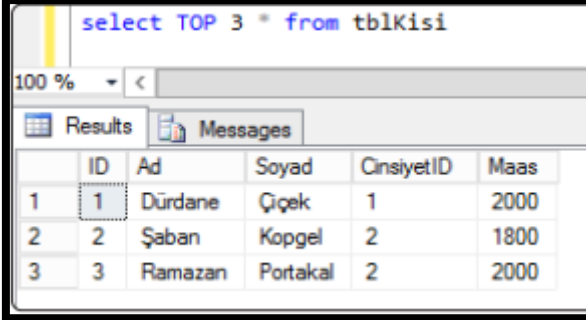
|   | KisiAd  | Kisi Soyad |
|---|---------|------------|
| 1 | Dürdane | Çiçek      |
| 2 | Şaban   | Koppel     |
| 3 | Ramazan | Portakal   |
| 4 | Naciye  | Çilek      |

## TOP

SELECET sorgusu ile dönen kayıtların belli bir sayıda olmasını istememiz halinde TOP anahtar sözcüğünü kullanırız.

Tablomuzda yer alan kayıtlardan ilk 3 kaydı görüntülemek için TOP 3 yazarız

```
select TOP 3 * from tblKisi
```



```
select TOP 3 * from tblKisi
```

|   | ID | Ad      | Soyad    | CinsiyetID | Maas |
|---|----|---------|----------|------------|------|
| 1 | 1  | Dürdane | Çiçek    | 1          | 2000 |
| 2 | 2  | Şaban   | Koppel   | 2          | 1800 |
| 3 | 3  | Ramazan | Portakal | 2          | 2000 |

Kayıtların %25'ini görüntülemek için **TOP 25 Percent** yazmamız gerekmektedir.

Top sorgumuzu kullanarak son kayıt(lar)a ulaşabiliriz. Bunun için TOP ifadesi ile birlikte **Order By DESC** ifadesini eklememiz gerekecektir.

## IN (in) İşleci

```
SELECT * FROM personel WHERE unvani IN ( "Genel Müdür", "Müdür", "Müdür Yardımcısı" )
```

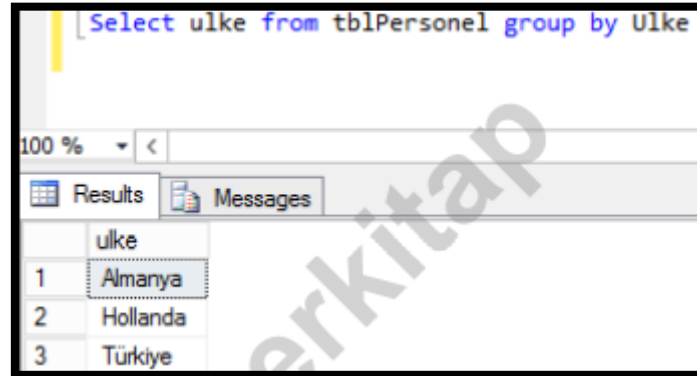
Bu örnekte personel tablosunda **unvani alanı** Genel Müdür, Müdür veya Müdür Yardımcısı olan kayıtlar listelenecektir. Bu ifadeler yerine ne yazarsak onu arar SQL.

## GROUP BY

Tablolarımızdaki verileri bir veya birden fazla sütuna göre gruplandırma yapmak için kullanılır.

Öncelikle tablomuzdaki ülkelere göre bir gruplandırma yapalım. Aynı ülkeden birden fazla olmasına rağmen aynı ülkelerden **sadece bir tanesi** görülecektir.

`Select ulke from tblPersonel group by Ulke`



The screenshot shows a SQL query editor with the query `Select ulke from tblPersonel group by Ulke`. Below the query, the 'Results' tab is active, displaying a table with two columns: 'ulke' and an index. The results are as follows:

|   | ulke     |
|---|----------|
| 1 | Almanya  |
| 2 | Hollanda |
| 3 | Türkiye  |

Bir ülkeye ait şehir sayısını öğrenebiliriz:

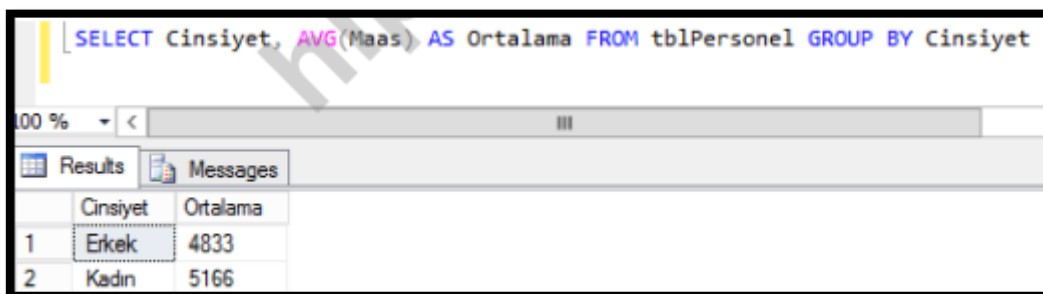


The screenshot shows a SQL query editor with the query `SELECT ulke, COUNT(*) AS ToplamSehir FROM tblPersonel GROUP BY ulke`. Below the query, the 'Results' tab is active, displaying a table with two columns: 'ulke' and 'ToplamSehir'. The results are as follows:

|   | ulke     | ToplamSehir |
|---|----------|-------------|
| 1 | Almanya  | 2           |
| 2 | Hollanda | 3           |
| 3 | Türkiye  | 4           |

Cinsiyet kolonuna göre erkek ve kadınların maaş ortalamasını ayrı ayrı alabiliriz.

`SELECT Cinsiyet, AVG(Maas) AS Ortalama FROM tblPersonel GROUP BY Cinsiyet`

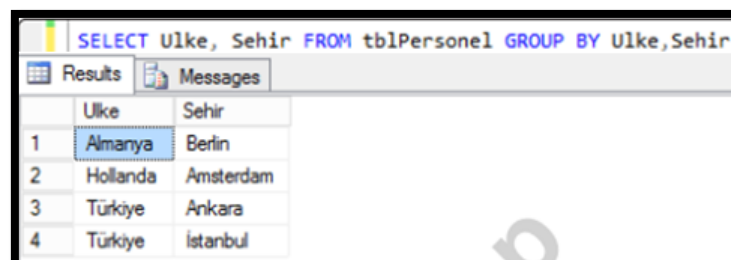


The screenshot shows a SQL query editor with the query `SELECT Cinsiyet, AVG(Maas) AS Ortalama FROM tblPersonel GROUP BY Cinsiyet`. Below the query, the 'Results' tab is active, displaying a table with two columns: 'Cinsiyet' and 'Ortalama'. The results are as follows:

|   | Cinsiyet | Ortalama |
|---|----------|----------|
| 1 | Erkek    | 4833     |
| 2 | Kadın    | 5166     |

Buradaki sorgumuzda, Ulke ve Sehir sütununa göre gruplandırma yapılacaktır.

`SELECT Ulke, Sehir FROM tblPersonel GROUP BY Ulke,Sehir`

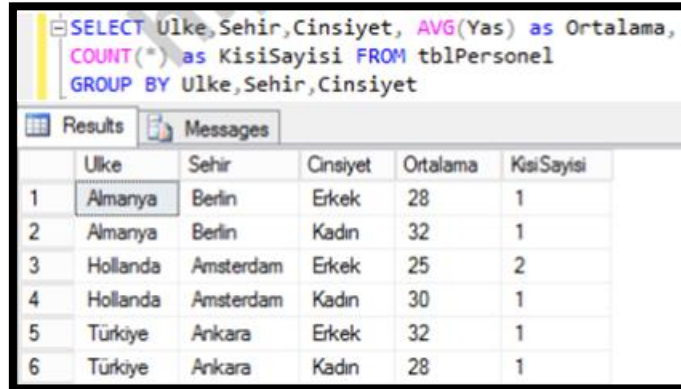


The screenshot shows a SQL query editor with the query `SELECT Ulke, Sehir FROM tblPersonel GROUP BY Ulke,Sehir`. Below the query, the 'Results' tab is active, displaying a table with two columns: 'Ulke' and 'Sehir'. The results are as follows:

|   | Ulke     | Sehir     |
|---|----------|-----------|
| 1 | Almanya  | Berlin    |
| 2 | Hollanda | Amsterdam |
| 3 | Türkiye  | Ankara    |
| 4 | Türkiye  | İstanbul  |

Her ülke/şehir bazında cinsiyete göre yaş ortalamasını ve hangi şehirde kaç kişi olduğunu öğrenebiliriz.

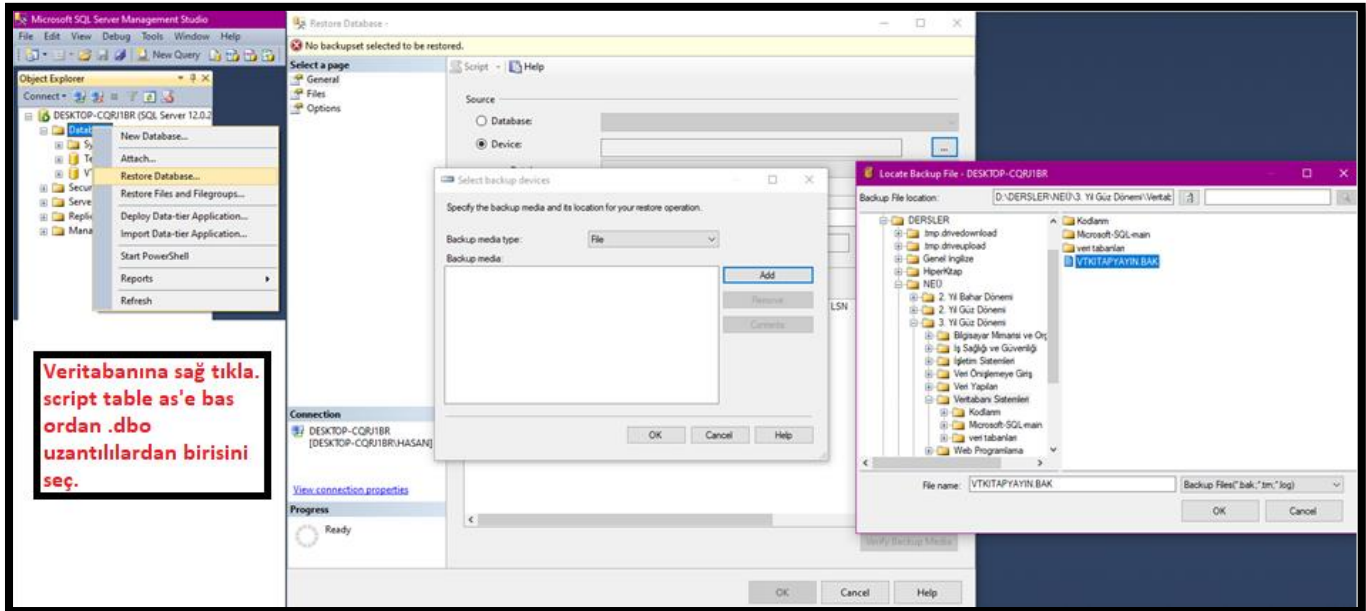
```
SELECT Ulke, Sehir, Cinsiyet, AVG(Yas) as Ortalama, COUNT(*) as KisiSayisi  
FROM tblPersonel GROUP BY Ulke, Sehir, Cinsiyet
```



The screenshot shows a SQL query in the 'Query Editor' window: `SELECT Ulke, Sehir, Cinsiyet, AVG(Yas) as Ortalama, COUNT(*) as KisiSayisi FROM tblPersonel GROUP BY Ulke, Sehir, Cinsiyet`. Below the query, the 'Results' pane displays a table with 6 rows and 6 columns: Ulke, Sehir, Cinsiyet, Ortalama, and KisiSayisi. The data is as follows:

|   | Ulke     | Sehir     | Cinsiyet | Ortalama | KisiSayisi |
|---|----------|-----------|----------|----------|------------|
| 1 | Almanya  | Berlin    | Erkek    | 28       | 1          |
| 2 | Almanya  | Berlin    | Kadin    | 32       | 1          |
| 3 | Hollanda | Amsterdam | Erkek    | 25       | 2          |
| 4 | Hollanda | Amsterdam | Kadin    | 30       | 1          |
| 5 | Türkiye  | Ankara    | Erkek    | 32       | 1          |
| 6 | Türkiye  | Ankara    | Kadin    | 28       | 1          |

### .bak Dosyası Nasıl Açılır?

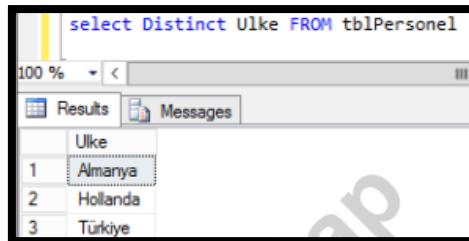


### DISTINCT

Tekrarlanan verileri eleyerek her farklı veriden yalnız bir adet bulunmasını istiyorsak kullanırız.

tblPersonel tablomuzda kaç farklı ülke olduğunu öğrenebiliriz.

```
select Distinct Ulke FROM tblPersonel
```



The screenshot shows a SQL query in the 'Query Editor' window: `select Distinct Ulke FROM tblPersonel`. Below the query, the 'Results' pane displays a table with 3 rows and 1 column: Ulke. The data is as follows:

|   | Ulke     |
|---|----------|
| 1 | Almanya  |
| 2 | Hollanda |
| 3 | Türkiye  |

## HAVING

Having aslında where ifadesine benzemektedir. Aralarındaki fark;

1. **where, group by** öncesinde yapılır ve bu koşula göre group by ile sıralanır
2. **having**, oluşturulmuş group by üzerinden koşul yapısını çalıştırır.

Şehir kolonuna göre toplam maaş kolonu değeri 10000'den büyük olanları listemek için having yapısını kullanabiliriz.

```
SELECT Sehir, Sum(Maas) as Toplam from tblPersonel group by Sehir Having Sum(Maas)>10000
```

|   | Sehir     | Toplam |
|---|-----------|--------|
| 1 | Amsterdam | 14000  |
| 2 | Ankara    | 11500  |
| 3 | İstanbul  | 11000  |

Maaş ortalaması 3000'den fazla olan ülkelerdeki erkek çalışanların maaş ortalamasının sorgusu:

```
SELECT ülke avg(maas) FROM personel WHERE cinsiyet='E' GROUP BY ülke HAVING avg(maas)>3000
```

## Kümeleme Fonksiyonları

**1- count**, bir alandaki NULL olmayan değerlerin kaç adet olduğunu yani sayısını hesaplar.

Amerikada çalışanların sayısını öğrenmek için aşağıdaki sorguyu çalıştırırız.

```
select count(*) as Sayi from personel where ülke='ABD'
```

Kaç farklı ülkeden çalışan kişi olduğunu hesaplamak istersek:

```
select count(distinct ULKE) from personel
```

## 2- max() – min() - sum() Fonksiyonları

Bir yerdeki en yaşlı çalışanı ismi ile birlikte sorgulamak istersek:

```
SELECT isim, yas AS enYasli FROM personel WHERE yas=(SELECT max(yas) from personel)
```

Sayısal değerlerin toplamını bulmak için sum() fonksiyonu kullanılır.

```
SELECT sum(maas) AS toplamMaas FROM personel
```

## 3- round() – ceiling() – floor() – avg() Fonksiyonları

**round(sayı, basamak)** Yakın olan tam sayıya yuvarlar. İkinci parametre olarak verilen değer, virgülden sonra kaçınıcı değerden sonra dikkate alınacağını belirler.

**ceiling(sayı)** Üste yuvarlanır. **floor(sayı)** Alta yuvarlanır. **avg()**: Belirtilen kolonun ortalamasını alır.

```
SELECT round(5.576, 1) as ROUND_1,  
       round(5.576, 2) as ROUND_2,  
       ceiling(5.576) as CEILING_,  
       floor(5.576) as FLOOR_,  
       round(5.576, 2) as ROUND_2
```

|   | ROUND_1 | ROUND_2 | CEILING_ | FLOOR_ | ROUND_2 |
|---|---------|---------|----------|--------|---------|
| 1 | 5.600   | 5.580   | 6        | 5      | 5.580   |

## Join (Inner Join)

Veri tabanında oluşturduğumuz tabloları birleştirmek için Join ifadesini kullanırız. Sadece Join yazdığımızda sistem bunu Inner Join olarak algılamaktadır. **Kısaca, JOIN'in sağında ve soluna kalan tablolar birleştirilir.**

```
Create table tblCins(  
    ID int identity(1,1) primary key,  
    Cins varchar(10)  
);  
insert into tblCins values ('Kadın')  
insert into tblCins values ('Erkek')  
  
Create table tblMedeni(  
    ID int identity(1,1) primary key,  
    Medeni varchar(10)  
);  
insert into tblMedeni values ('Evli')  
insert into tblMedeni values ('Bekar')  
  
Create table tblPersDetay(  
    ID int identity(1,1) primary key,  
    Ad varchar(50),  
    CinsiyetID int,  
    MedeniID int  
);  
insert into tblPersDetay values ('Dürdane', 1, 2)  
insert into tblPersDetay values ('Hacer', 1, 1)  
insert into tblPersDetay values ('Şaban', 2, 1)  
insert into tblPersDetay values ('Ramazan', 2, 2)  
insert into tblPersDetay values ('Bayram', 2, 1)
```

Tablolarımızı incelediğimizde **tblPersDetay** tablosunda yer alan **CinsiyetID** ve **MedeniID** sütunlarında yer alan değerler, **tblCins** ve **tblMedeni** tablosunda yer alan **ID** sütununa göre yazılmıştır. **ID** değerine göre karşılığını alabilmemiz gerekmektedir.

select \* from tblPersDetay  
select \* from tblCins  
select \* from tblMedeni

100 %

Results Messages

|   | ID | Ad      | CinsiyetID | MedeniID |
|---|----|---------|------------|----------|
| 1 | 1  | Dürdane | 1          | 2        |
| 2 | 2  | Hacer   | 1          | 1        |
| 3 | 3  | Şaban   | 2          | 1        |
| 4 | 4  | Ramazan | 2          | 2        |
| 5 | 5  | Bayram  | 2          | 1        |

Burada yer alan sayıları anlamak zor.  
Hangi değerin Kadın/Erkek veya Evli/Bekar olduğu belli değil.

| ID | Cins  |
|----|-------|
| 1  | Kadın |
| 2  | Erkek |

| ID | Medeni |
|----|--------|
| 1  | Evli   |
| 2  | Bekar  |

Tablolarda yer alan ID kolonlarının karşılıkları gösterdiğimizde daha anlaşılır olacaktır.

Şimdi, **JOIN** kullanarak **tblPersDetay** tablosunda yer alan anlamsız kolonlarımızın karşılıklarını getirelim. **JOIN**'i anlamak için hangi tablodan hangi kolon değerini çekeceğimizi belirlememiz yeterli olacaktır.

| tblPersDetay |    |         |            |          |
|--------------|----|---------|------------|----------|
|              | ID | Ad      | CinsiyetID | MedeniID |
| 1            | 1  | Dürdane | 1          | 2        |
| 2            | 2  | Hacce   | 1          | 1        |
| 3            | 3  | Şaban   | 2          | 1        |
| 4            | 4  | Ramazan | 2          | 2        |
| 5            | 5  | Bayram  | 2          | 1        |

| tblCins |       |
|---------|-------|
|         | Cins  |
| 1       | Kadın |
| 2       | Erkek |

| tblMedeni |        |
|-----------|--------|
|           | Medeni |
| 1         | Evlü   |
| 2         | Bekar  |

Sorgumuzu yazdığımızda **tabloAdı.sütun** değerine göre yazmamız gerekmektedir.

```
select tblPersDetay.Ad, tblCins.Cins, tblMedeni.Medeni from tblPersDetay
JOIN tblCins on tblPersDetay.CinsiyetID = tblCins.ID
JOIN tblMedeni on tblPersDetay.MedeniID = tblMedeni.ID
```

```
select tblPersDetay.Ad, tblCins.Cins, tblMedeni.Medeni
from tblPersDetay
JOIN tblCins on tblPersDetay.CinsiyetID=tblCins.ID
JOIN tblMedeni on tblPersDetay.MedeniID=tblMedeni.ID
```

|   | Ad      | Cins  | Medeni |
|---|---------|-------|--------|
| 1 | Dürdane | Kadın | Bekar  |
| 2 | Hacce   | Kadın | Evlü   |
| 3 | Şaban   | Erkek | Evlü   |
| 4 | Ramazan | Erkek | Bekar  |
| 5 | Bayram  | Erkek | Evlü   |

## GO İşleci

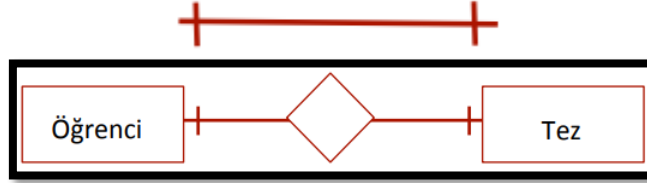
```
INSERT INTO TblDersler VALUES ('Fen Bilgisi')
-- Burada id'miz otomatik olarak artacak ve şuan id'miz 1.
GO
INSERT INTO TblDersler VALUES ('Matematik') -- id'miz 2 oldu.
GO 5
-- Yukarıdaki GO'lar anlamı GO'lar arasında kalan kodu 5 kere çalıştır demek lakin
id'lerimiz otomatik olarak artacak ve birbirlerinden farklıdır.
```

## İlişkisel Tablolar

Tutulan her türlü veriye **varlık (sütun-kolon)** denir. Varlık kümeleri arasındaki bağıntıya **ilişki** adı verilir. İlişkisel veri tabanı, birbirileriyle ilişkili verilerin aralarında bağlantı kurularak tasarlanması ile oluşturulmuş bir veri tabanı türüdür. Bu ilişkisel yapı aynı anda birçok tablodan veri çekilebilmesini sağlar. İlişkisel tablo kullanılarak; **tekrar edilen veriler ve iş yükü azaltılır ayrıca veritabanının kontrolü sağlanır.**

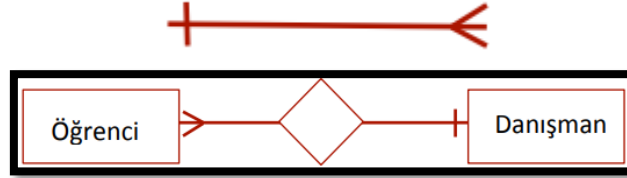
### Bire Bir İlişki (1 --> 1)

- Bir kişiye yalnızca bir e-posta adresi atanabilir, bir e-posta adresi tek bir kişiye atanabilir.



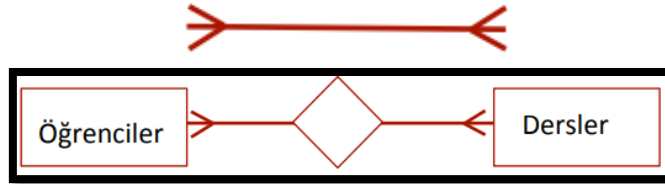
### Bire Çok İlişki (1 --> n)

- Bir öğrenci bir tek sınıfta bulunabilir, ancak bir sınıfta birden çok öğrenci olabilir.



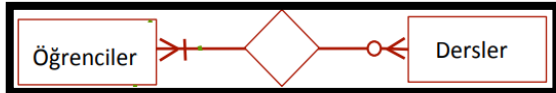
### Çoka Çok İlişki (m --> n)

- Şarkılar ve sanatçılar tablosu düşünün, bir sanatçı birden fazla şarkıyı seslendirebilir, aynı zamanda bir şarkı birden fazla sanatçı tarafından seslendirilebilir. Böyle durumlarda her iki tablonun da başvurduğu bir ara tablo yapılır. Bu tabloda her iki tablonun ID değerleri saklanır. Tablolar, bu ara tablo aracılığı ile bağlanırlar.
- Bir öğrenci birden çok ders alabilir, bir derste birden çok öğrenci bulunabilir.
- Bir öğrenci birden çok kulübe üye olabilir, bir kulüp birden çok öğrenci içerebilir.



### Zorunlu Katılım

- Her derste en az bir öğrenci olmak zorundadır.
- Öğretim üyesi bulunmayan bir bölüm açılmaz.



### Opsiyonel Katılım

- Öğrenciler bir kulübün üyesi olmak zorunda değildir.



### Hiç ya da Bir



### Hiç ya da Çok



### Yalnızca Bir



### Bir ya da Çok

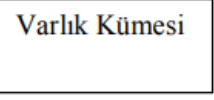


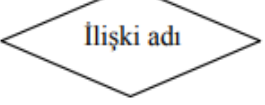
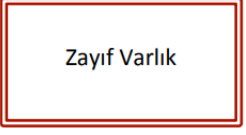

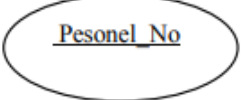




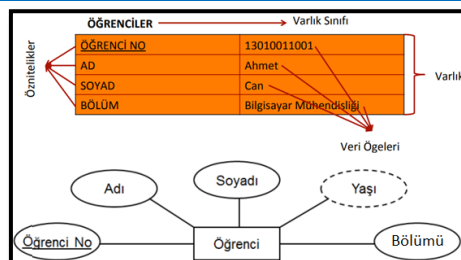


## Varlık – İlişki Diyagramı (ER Diagram)

**Foreign Key**, başka bir tablonun birincil anahtarının diğer bir tablo içerisinde yer almasıdır. **Özetle** bir tablonun Primary Key alanı, diğer bir tabloda Foreign Key olarak tanımlanır.

**Foreign Key**, **primary key** ile bağlıysa silinmez.

|  |  |
|--|--|
|     | Veritabanında bilgisi tutulmak istenen her şeye <b>varlık</b> denir. Varlığın ismi diktörtgenin içine yazılır.<br>Benzer varlıklardan oluşan gruba <b>varlık sınıfı</b> ya da <b>veri kümesi</b> denir.  |
|     | Bir varlığın veritabanındaki her bir özelliğine <b>nitelik</b> denir.<br><b>Çekirdek Nitelik:</b> Daha alt parçalara bölünemeyen niteliklerdir.  |
|     | <b>Birleşik Nitelikler:</b> Çekirdek niteliklerin birleştirilmesiyle oluşturulur.  |
|     | <b>İlişki</b> , iki veya daha fazla varlık kümesini birbirine bağlamada kullanılır.<br><b>Diğer bir deyişle</b> , veritabanında bulunan varlıklar arasındaki işlemlerdir.  |
|    | Sistemde tek başına ayakta duramayan, varlığı bir başka varlığın bulunmasına bağlı olan varlıklara <b>zayıf varlık</b> denir.<br><b>Diğer bir deyişle</b> ; bir varlık kümesi anahtar niteliğe (Primary Key) sahip değilse <b>zayıf varlık kümesi</b> olarak adlandırılır.<br>Bu varlıklar arasındaki bağıntılar <b>varolma bağıntısı</b> olarak adlandırılır. |
|  |  |
|   | Bir varlık kümesi içindeki varlıkları birbirinden ayırt etmek için kullanılan nitelik veya nitelik grubuna <b>birincil anahtar</b> ya da <b>anahtar nitelik</b> denir. (PRIMARY KEY) <b>Altı çizili şekilde gösterilir.</b>  |
|   | Veri tabanında başka bir nitelik kullanılarak elde edilen özelliğe <b>türetilmiş nitelik</b> denir.<br>Örneğin doğum tarihinden yararlanarak yaş niteliğinin elde edilmesi buna en iyi örnektir. Güncel tarihten doğum tarihi çıkarılarak güncel yaş hesaplanır ve gösterilir  |
|   | Birden fazla niteliğin bir araya gelerek başka bir niteliğin oluşturulmasına <b>çok değerli nitelik</b> denir.<br>Buna en iyi örnek adres alanı olabilir. Tek tek kaydedilen adres bilgileri birleştirilerek tek bir adres niteliğinde gösterilebilir.   |



| KitapTable |                |             |            |       |
|------------|----------------|-------------|------------|-------|
| KitapID    | KitapAd        | Yazar       | YayinEviID | Fiyat |
| 1          | Suç ve Ceza    | Dostoyewski | 1          | 18    |
| 2          | Suç ve Ceza    | Dostoyewski | 3          | 25    |
| 3          | Sefiller       | Victor Hugo | 2          | 20    |
| 4          | Anne Karenina  | Tolstoy     | 3          | 30    |
| 5          | Diriliş        | Tolstoy     | 3          | 21    |
| 6          | Savaş ve Barış | Tolstoy     | 1          | 29    |

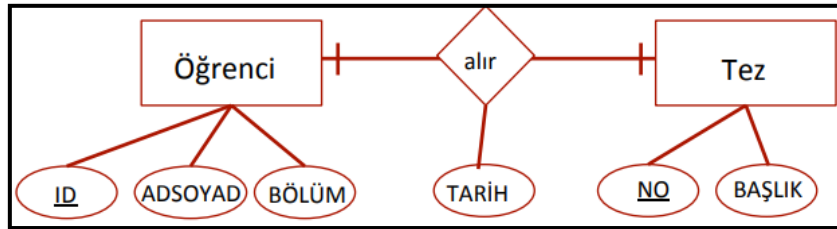
Primary Key: KitapID  
Foreign Key: YayinEviID

| YayinEviTable |                |                  |                      |              |
|---------------|----------------|------------------|----------------------|--------------|
| YayinEviID    | Yayinevi       | YayinEviIletisim | YayineviAdres        | YayinEviIBAN |
| 1             | İş Bankası     | 2126665544       | Gül Sk. Menekşe Apt. | 6251545454   |
| 2             | Sis Yayıncılık | 2165451252       | Güneş mah. Çiçek sk. | 2514253625   |
| 3             | Yapı Kredi     | 312654821        | hagsd ahgsfdhgasvd   | 6,55487E+12  |

### ER Diyagramlarının Tablolara Dönüştürülmesi

#### Bire Bir İlişki (1 --> 1)

- Varlıkların nitelikleri tabloların kolonları/sütunlarıdır.
- Uygun **yabancı anahtar (foreign key)** ile iki tablo birleştirilir.
- Aradaki ilişkiye ait nitelikler varsa, **yabancı anahtar (foreign key)** eklenen tabloya sütun olarak eklenir.

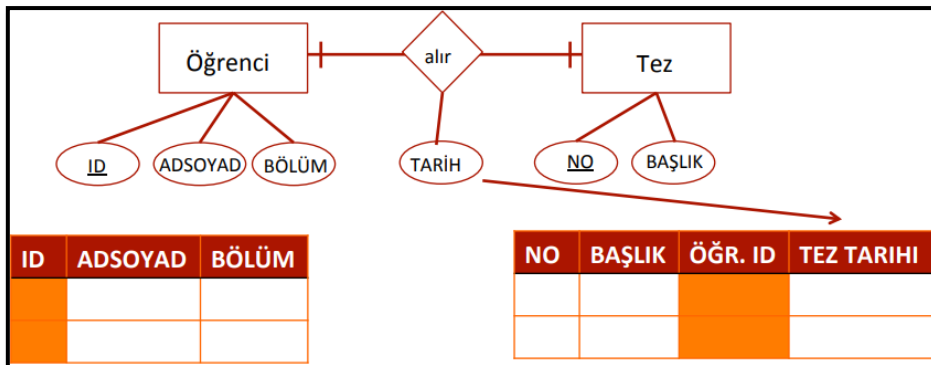


| ID | AD SOYAD | BÖLÜM |
|----|----------|-------|
|    |          |       |
|    |          |       |

| NO | BAŞLIK |
|----|--------|
|    |        |
|    |        |

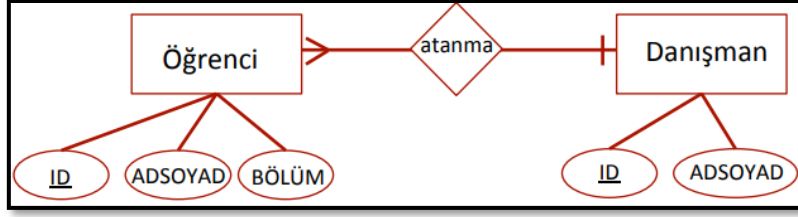
| ID | AD SOYAD | BÖLÜM |
|----|----------|-------|
|    |          |       |
|    |          |       |

| NO | BAŞLIK | ÖĞR. ID |
|----|--------|---------|
|    |        |         |
|    |        |         |



### Bire Çok İlişki (1 --> n)

- Yabancı anahtar, ilişkinin n tarafındaki tabloya eklenir.



| ID | AD SOYAD | BÖLÜM |
|----|----------|-------|
|    |          |       |
|    |          |       |

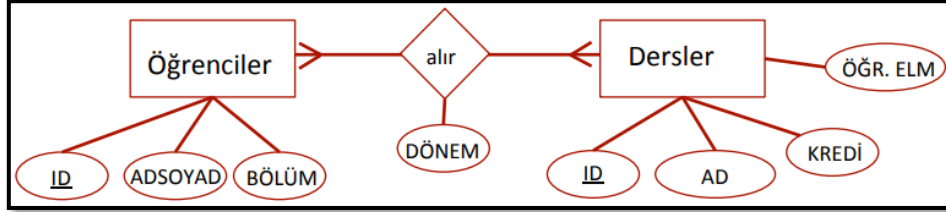
| ID | AD SOYAD |
|----|----------|
|    |          |
|    |          |

| ID | AD SOYAD | BÖLÜM | DANŞ. ID |
|----|----------|-------|----------|
|    |          |       |          |
|    |          |       |          |

| ID | AD SOYAD |
|----|----------|
|    |          |
|    |          |

### Çoka Çok İlişki (m --> n)

- Varlıklar arasındaki ilişki için yeni bir tablo oluşturulur.
- Varlıkların anahtarları ilişki tablosuna yabancı anahtar (foreign key) olarak eklenir.
- Gerekirse oluşan yeni tabloya birincil anahtar (primary key) eklenir.



| ID | AD SOYAD | BÖLÜM |
|----|----------|-------|
|    |          |       |
|    |          |       |

| ID | AD | KREDİ | ÖĞR. ELM. |
|----|----|-------|-----------|
|    |    |       |           |
|    |    |       |           |

| ID | AD SOYAD | BÖLÜM |
|----|----------|-------|
|    |          |       |
|    |          |       |

| ÖĞR. ID | DERS. ID | DÖNEM |
|---------|----------|-------|
|         |          |       |
|         |          |       |

| ID | AD | KREDİ | ÖĞR. ELM. |
|----|----|-------|-----------|
|    |    |       |           |
|    |    |       |           |

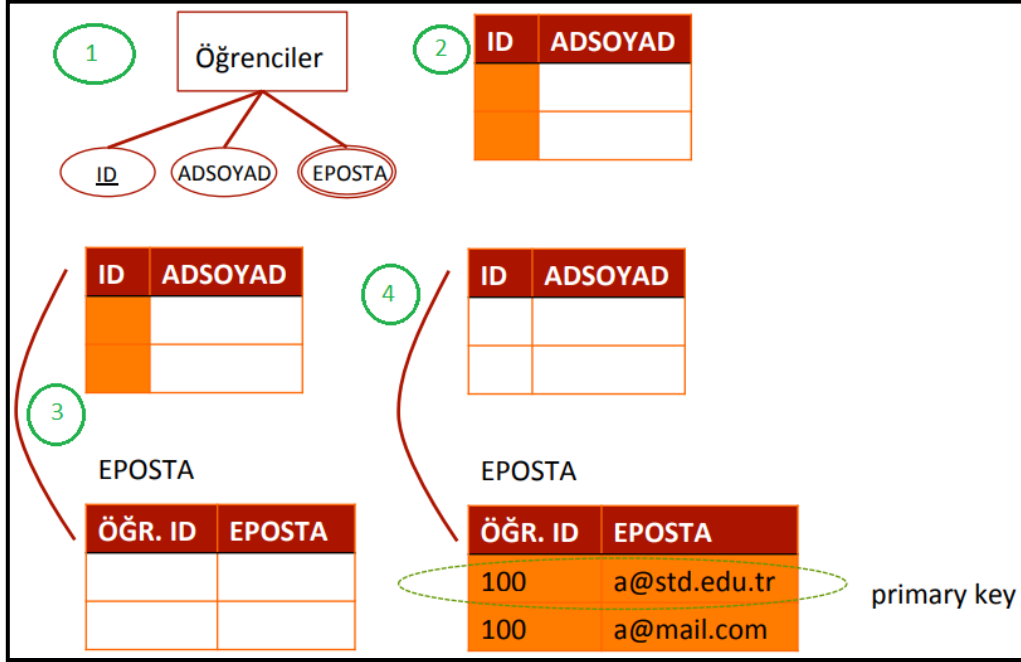
| ID | AD SOYAD | BÖLÜM |
|----|----------|-------|
|    |          |       |
|    |          |       |

| DERS ALMA ID | ÖĞR. ID | DERS. ID | DÖNEM |
|--------------|---------|----------|-------|
|              |         |          |       |
|              |         |          |       |

| ID | AD | KREDİ | ÖĞR. ELM. |
|----|----|-------|-----------|
|    |    |       |           |
|    |    |       |           |

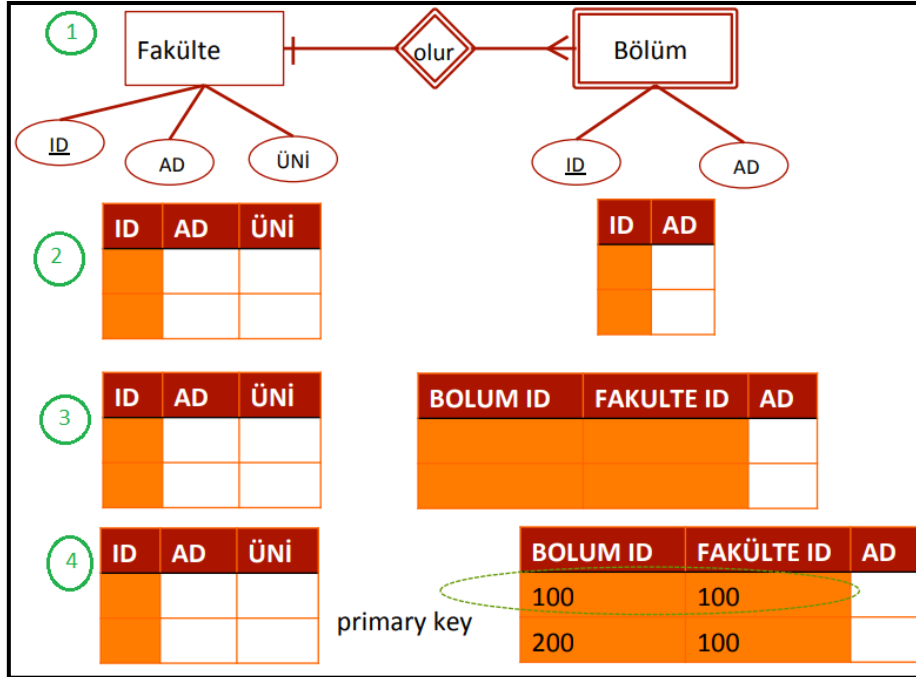
### Multi-Valued (Çok Değer Alabilen)

- Çok değerli nitelikler için yeni bir tablo oluşturulur.
- Mevcut tablonun **primary key**'i, yeni tabloya **yabancı anahtar (foreign key)** olarak eklenir.
- Yeni tablonun **primary key**'i, çok değerli alan ile **foreign key**'in **birleşimidir**.



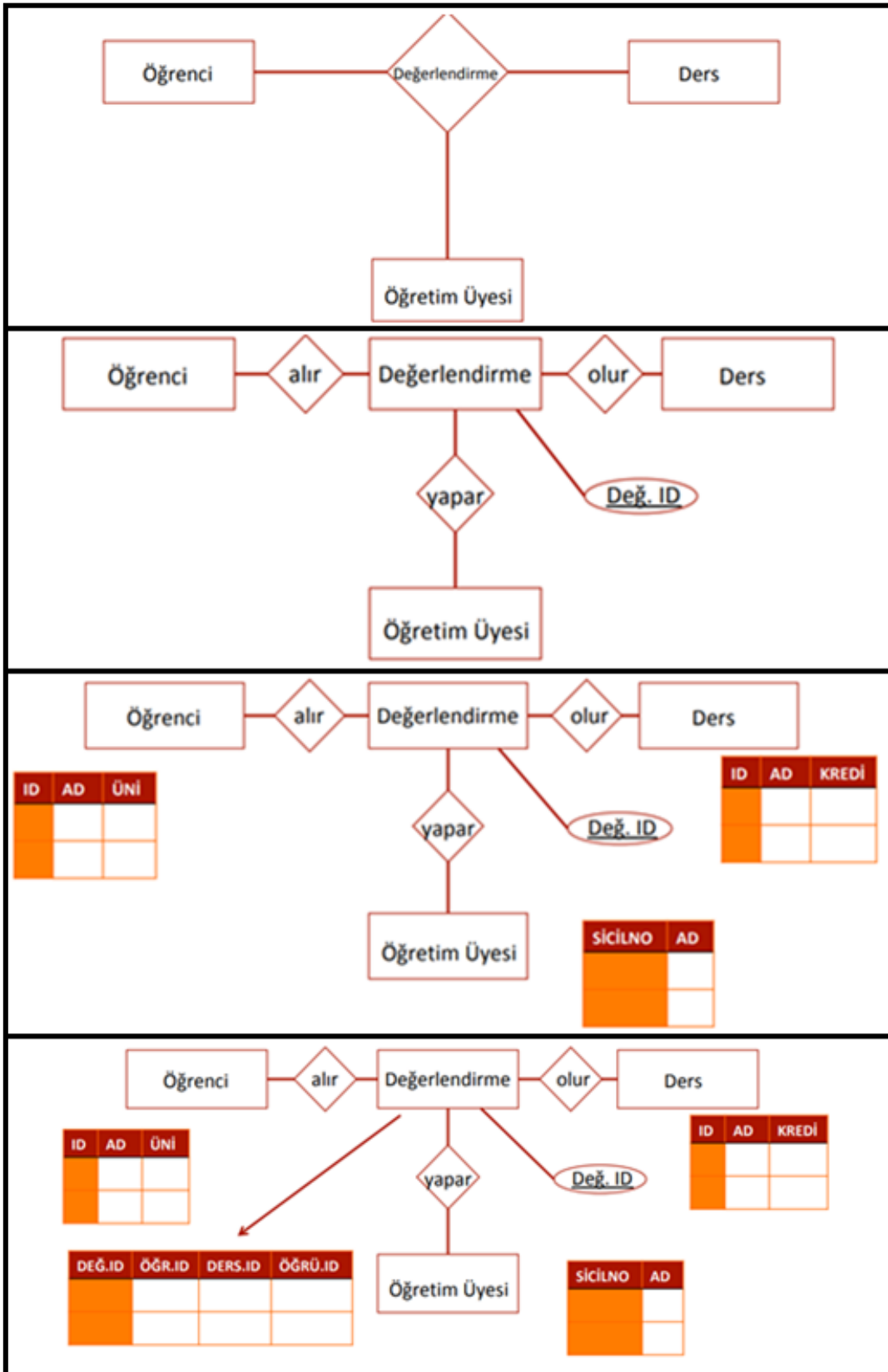
### Zayıf Varlıklar

- Zayıf varlıklar tabloya dönüştürülür.
- Tablonun **primary key**'i, kendi anahtarı ile bağımlı olduğu varlığa ait **foreign key**'in **birleşimidir**.

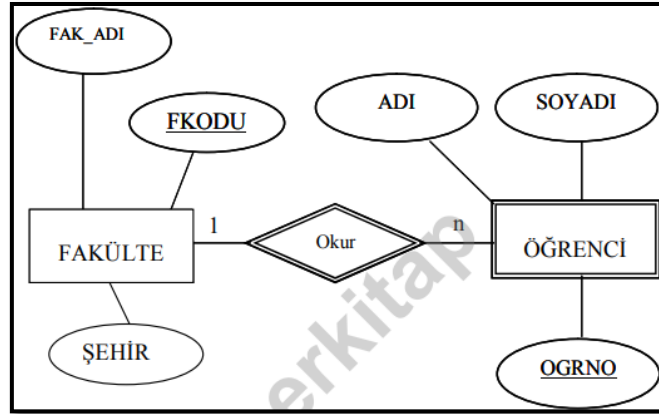


### Çok Varlıklı İlişkiler

- Varlıklar arasında ikili ilişkiler oluşacak şekilde, ilişki adına yeni bir varlık oluşturulur.
- Oluşturulan yeni varlığa anahtar atanır.
- Diğer varlıkların anahtarları yeni varlığa **yabancı anahtar (foreign key)** olarak eklenir.



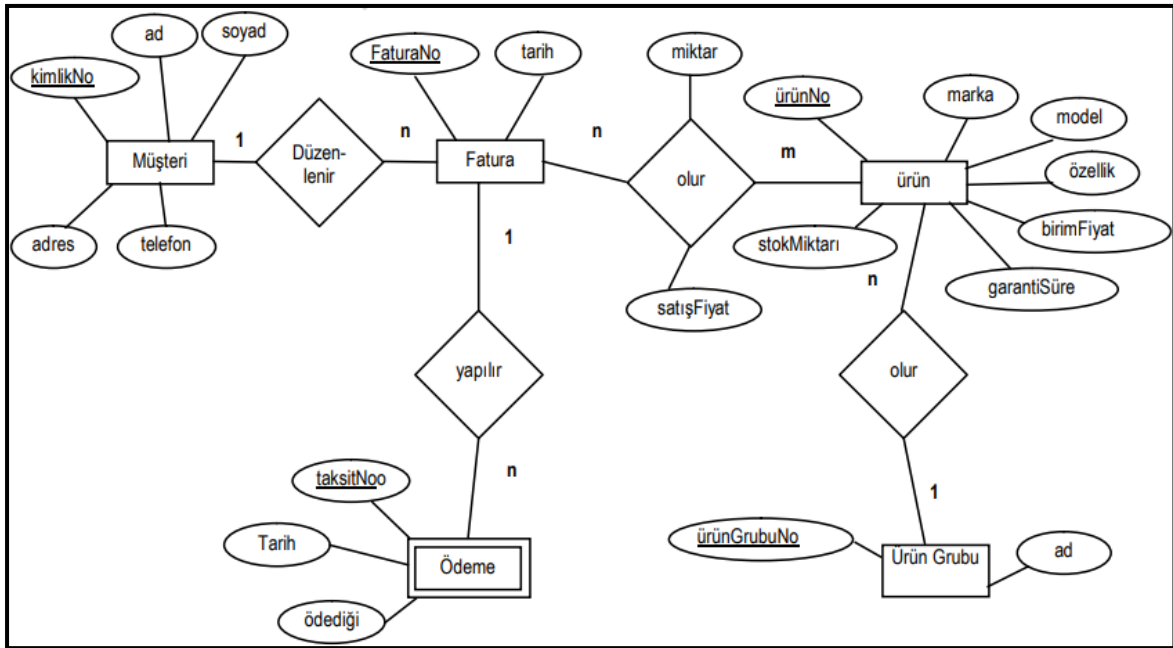
FAKÜLTE-ÖĞRENCİ ilişkisinde farklı fakültelerde okuyan öğrencilerin numaralarının aynı olabileceğinden ÖĞRENCİ varlık kümesi **zayıf varlık kümesi** olarak gösterilir.



Zayıf varlık kümesini güçlendirmek için ilişkili olduğu **güçlü varlık kümesinin** anahtarlı niteliği ile birlikte tek bir anahtar oluşturarak güçlendirilmiş olur. Dolayısıyla öğrenci numarası aynı olsa bile FKODU farklı olacağından ikisi birlikte öğrenci için güçlü bir anahtar oluşturmuş olur.

Beyaz eşya satan bir firma için aldığı ve sattığı ürün bilgilerini tutmak için veri tabanı hazırlanacaktır. Buna göre;

1. Firma satılan her ürünün ürünNo, markası, modeli, özellikleri, birim fiyatı, garanti süresi, ve stok miktarı bilgileri bulunmaktadır.
2. Satılan ürünler kategorilere ayrılmıştır, bir ürün sadece bir kategoride bulunurken bir kategoride birçok ürün bulunabilir.
3. Firmadan ürünleri satın alan müşteri bilgileri de tutulmaktadır. Müşterilerin kimlik No, ad, soyad, adres, telefon bilgileri saklanmaktadır. Müşteri ürün satın aldığı anda müşteriye fatura düzenlenir. Faturada faturano ve tarih bilgileri bulunmaktadır. Bir müşteri için birden çok fatura düzenlenir.
4. Bir faturada birden çok ürün bulunabilir. Bir ürün de birden çok faturada bulunabilir. Her ürünün satış fiyatı ve adeti bilgileri de ilişki içerisinde saklanmaktadır.
5. Faturadaki ürünlerin ödemesi peşin de yapılabilir taksit taksit de yapılabilir. Bu durumda bir faturanın bir ya da birden çok ödemesi olur. Yapılan ödemeni taksiti numarası, miktarı ve tarih bilgileri bulunur.

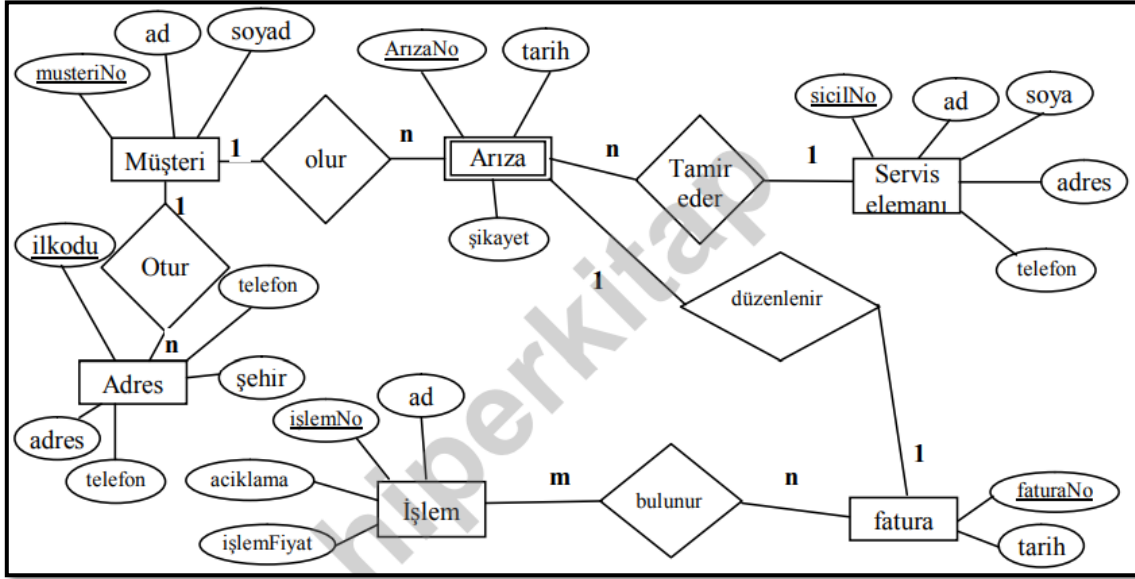


### Varlık İlişki Şeması

- Müşteri(kimlikNo, ad, soyada, adres, telefon)
- Fatura(faturaNo, tarih, kimlikNo)
- Ürün(ürünNo, marka,model, özellik, birimFiyat, garantiSüre, stokMiktarı, ürünGrubuNo)
- Firma(Firma\_no, firma\_adi)
- ÜrünGrubu(ürünGrubuNo, ad)
- Ödeme(taksitNo, tarih, ödediği, faturaNo)
- Olur(olurNo, faturaNo, ürünNo, adet, satışFiyatı)
- Alınır(Firma\_no, Urun\_no, miktar)

Bir beyaz eşya servisi, ürünlerinin bakım ve onarımını yaptığı müşterilerinin bilgilerini bir veri tabanında tutmak istiyor. Buna göre;

1. Serviste çalıştığı personelin sicil\_no, ad, soyad, telefon, adres bilgileri bulunmaktadır.
2. Müşterilerin müşteriNo, adı, soyadı,
3. Müşterinin adres bilgileri ayrı bir tabloda tutulmaktadır bir adres tablosunda ilkodu, şehir, adresi, telefon bilgilerini bulunmaktadır. Bir şehirde birden fazla müşteri bulunabilir, bir müşteride sadece bir adrese kaydedilmektedir.
4. Müşterinin ürünü ile ilgili arıza şikayeti olmaktadır. Arıza bilgileri içerisinde arıza talebinin tarihi, şikayet bilgileri bulunmaktadır. Bir müşterinin kombisinin birden çok arızası olmaktadır.
5. Bir arızayı gidermek için servis elemanı müşteriye gönderilir. Bir servis elemanı birden çok arızaya bakabilir.
6. Bakım yapıldıktan sonra bakım ile ilgili fatura düzenlenir. Faturano ve tarih bilgileri faturada bulunmaktadır. Bir arızaya bir fatura yazılır.
7. Faturada arızayı gidermek için yapılan her işlemin bir numarası, adı, açıklaması ve işlem fiyatı bulunmaktadır. Bir faturada birden çok işlem bulunabilir. Bir işlem birden çok faturada bulunabilir.



### Varlık İlişki Şeması

- Musteri (musteriNo, ad, soyad)
- Adres(ilkodu, şehir, adres, telefon)
- Arıza (ArızaNo, tarih, şikayet, müşteriNo, sicilNo, faturaNo)
- Servis\_elemanı (sicilNo, ad, soyad, adres, telefon)
- Fatura (faturaNo, tarih)
- İşlem (işlemNo, ad, açıklama, işlemFiyat)
- Bulunur (bulunurNo, işlemNo, faturaNo)

## NORMALİZASYON

- **Normalizasyon**, veritabanlarında çok fazla sütun ve satırdan oluşan bir tabloyu tekrarlardan arındırmak için daha az satır ve sütun içeren alt kümelerine ayrıştırma işlemidir.
- Normalizasyonun amacı, **veri tekrarını ortadan kaldırmak, veritabanını uygulamadan bağımsız hale getirmek (uygulama değişse bile veritabanı tutarlı olarak çalışmalı), performansı artırmaktır (arama hızlı olur).**
- Normalizasyon yaparken uyulması gereken kuralların her birine **normal form** denir. **Normal form derecesi arttıkça veri tutarlılığı artar.**
- Birinci – İkinci – Üçüncü Normal Form, ihlal edilirse ya da bu formların yapılmadığı tablolarda: **INSERT, DELETE** ve **UPDATE** işlemlerinde zorluk çekilebilir ya da hatalar meydana gelir.
- Üçüncü Normal Formda olan tablolar İkinci ve Birinci Normal Form tablolarına uygundur. Aynı şekilde İkinci Normal Forma uygun tablolar Birinci Normal Forma uygundur.

### 1. Birinci Normal Form

- Bir sütun, birden fazla bilgi içermemelidir.
- Tekrarlanan sütun yapıları olmamalıdır.

| MÜŞTERİ NO | ŞEHİR KODU | ŞEHİR    | GÖNDERİ NO | MİKTAR              |
|------------|------------|----------|------------|---------------------|
| 1          | 34         | İstanbul | 1,2,3,4,6  | 300,200,400,200,100 |
| 2          | 6          | Ankara   | 1,2        | 300,400             |
| 3          | 6          | Ankara   | 2          | 200                 |
| 4          | 34         | İstanbul | 2,4,5      | 200,300,400         |

| MÜŞTERİ NO | ŞEHİR KODU | ŞEHİR    | GÖNDERİ NO | MİKTAR | Birinci Normal Form Sorunları:  |
|------------|------------|----------|------------|--------|---|
| 1          | 34         | İstanbul | 1          | 300    | <b>insert</b> GÖNDERİ NO ve MİKTAR bilgileri olmadan yeni müşteri kaydı yapılamıyor.                      |
| 1          | 34         | İstanbul | 2          | 200    | <b>delete</b> Tek gönderi kaydı bulunan müşteriye ait kayıt silinirse, müşteriye ait veriler de kaybolur. |
| 1          | 34         | İstanbul | 3          | 400    | <b>update</b> Herhangi bir müşterinin şehir bilgisi değiştiğinde, tüm kayıtların güncellenmesi gerekir.   |
| 1          | 34         | İstanbul | 4          | 200    |   |
| 1          | 34         | İstanbul | 6          | 100    |   |
| 2          | 6          | Ankara   | 1          | 300    |   |
| 2          | 6          | Ankara   | 2          | 400    |   |
| 3          | 6          | Ankara   | 2          | 200    |   |
| 4          | 34         | İstanbul | 2          | 200    |   |
| 4          | 34         | İstanbul | 4          | 300    |   |
| 4          | 34         | İstanbul | 5          | 400    |   |



## 2. İkinci Normal Form (Tam İşlevsel Bağımlılık)

- Tabloda bir primary key olmalı ve anahtar olmayan sütunlar primary key'e bağımlı olmalı.
- Primary key birden fazla sütundan oluşuyorsa tablodaki veriler her iki sütuna da bağımlı olmalıdır.
- Kısmi bağımlı niteliklerin ayrılmasıyla yeni tablolar oluşabilir.

| MÜŞTERİ NO | ŞEHİR KODU | ŞEHİR    | GÖNDERİ NO | MİKTAR |
|------------|------------|----------|------------|--------|
| 1          | 34         | İstanbul | 1          | 300    |
| 1          | 34         | İstanbul | 2          | 200    |
| 1          | 34         | İstanbul | 3          | 400    |
| 1          | 34         | İstanbul | 4          | 200    |
| 1          | 34         | İstanbul | 6          | 100    |
| 2          | 6          | Ankara   | 1          | 300    |
| 2          | 6          | Ankara   | 2          | 400    |
| 3          | 6          | Ankara   | 2          | 200    |
| 4          | 34         | İstanbul | 2          | 200    |
| 4          | 34         | İstanbul | 4          | 300    |
| 4          | 34         | İstanbul | 5          | 400    |

**İkinci Normal Form (2NF) Sorunları**  
**insert** Yeni bir şehir kaydı için, yeni bir müşteri kaydı gerekiyor.  
**delete** Bir şehirdeki tek müşteri silinirse, şehire ait veriler kaybolur.

| MÜŞTERİ NO | ŞEHİR KODU | ŞEHİR    |
|------------|------------|----------|
| 1          | 34         | İstanbul |
| 2          | 6          | Ankara   |
| 3          | 6          | Ankara   |
| 4          | 34         | İstanbul |

| MÜŞTERİ NO | GÖNDERİ NO | MİKTAR |
|------------|------------|--------|
| 1          | 1          | 300    |
| 1          | 2          | 200    |
| 1          | 3          | 400    |
| 1          | 4          | 200    |
| 1          | 6          | 100    |
| 2          | 1          | 300    |
| 2          | 2          | 400    |
| 3          | 2          | 200    |
| 4          | 2          | 200    |
| 4          | 4          | 300    |
| 4          | 5          | 400    |

## 3. Üçüncü Normal Form (Geçişsiz Bağımlılık)

- Anahtar olmayan sütunlar anahtar sütuna tam bağımlı olmalı. Anahtar olmayan sütuna bağımlı olmamalı.

| M.NO | Ş.KOD | ŞEHİR    |
|------|-------|----------|
| 1    | 34    | İstanbul |
| 2    | 6     | Ankara   |
| 3    | 6     | Ankara   |
| 4    | 34    | İstanbul |

| M.NO | G.NO | MİKTAR |
|------|------|--------|
| 1    | 1    | 300    |
| 1    | 2    | 200    |
| 1    | 3    | 400    |
| 1    | 4    | 200    |
| 1    | 6    | 100    |
| 2    | 1    | 300    |
| 2    | 2    | 400    |
| 3    | 2    | 200    |
| 4    | 2    | 200    |
| 4    | 4    | 300    |
| 4    | 5    | 400    |

| MÜŞTERİ NO | GÖNDERİ NO | MİKTAR |
|------------|------------|--------|
| 1          | 1          | 300    |
| 1          | 2          | 200    |
| 1          | 3          | 400    |
| 1          | 4          | 200    |
| 1          | 6          | 100    |
| 2          | 1          | 300    |
| 2          | 2          | 400    |
| 3          | 2          | 200    |
| 4          | 2          | 200    |
| 4          | 4          | 300    |
| 4          | 5          | 400    |

| MÜŞTERİ NO | ŞEHİR KODU |
|------------|------------|
| 1          | 34         |
| 2          | 6          |
| 3          | 6          |
| 4          | 34         |

| ŞEHİR KODU | ŞEHİR    |
|------------|----------|
| 34         | İstanbul |
| 6          | Ankara   |

## 4. Dördüncü Normal Form

Bir tabloda anahtar nitelik ile anahtar olmayan nitelikler arasında birden fazla (1-n) ilişki olmamalıdır.

## 5. Beşinci Normal Form

Veri tekrarının minimum olması için tablolar ayrılacakları en küçük tablolara ayrılmalıdır.

## ÖRNEK-1

| Ö.NO    | Ö.AD   | Ö.SOYAD | DERS_NO | DERS_ADİ                             | VIZE | FINAL | H.NO | H.AD    | H.SOYAD |
|---------|--------|---------|---------|--------------------------------------|------|-------|------|---------|---------|
| 2001001 | Ahmet  | Solmaz  | 202     | Matematik 2                          | 70   | 60    | 11   | Özlem   | UÇAR    |
| 2001001 | Ahmet  | Solmaz  | 203     | Fizik 2                              | 80   | 40    | 11   | Özlem   | UÇAR    |
| 2001001 | Ahmet  | Solmaz  | 204     | Bilgisayar Mühendisliğine Giriş 2    | 60   | 45    | 3    | Aydın   | CARUS   |
| 2001001 | Ahmet  | Solmaz  | 205     | Atatürk İlkeleri ve İnkılap Tarihi 2 | 90   | 95    | 9    | Zeki    | DURMUŞ  |
| 2001001 | Ahmet  | Solmaz  | 206     | Türk Dili 2                          | 70   | 75    | 12   | Nebahat | YILDIZ  |
| 2001005 | Seyhan | Gülmez  | 202     | Matematik 2                          | 80   | 95    | 11   | Özlem   | UÇAR    |
| 2001005 | Seyhan | Gülmez  | 203     | Fizik 2                              | 80   | 70    | 11   | Özlem   | UÇAR    |
| 2001005 | Seyhan | Gülmez  | 204     | Bilgisayar Mühendisliğine Giriş 2    | 60   | 70    | 3    | Aydın   | CARUS   |
| 2001002 | Selim  | Solmaz  | 702     | Veri Tabanı Yönetimi                 | 60   | 50    | 6    | Altan   | MESUT   |
| 2001003 | Ahmet  | Vardar  | 702     | Veri Tabanı Yönetimi                 | 60   | 60    | 6    | Altan   | MESUT   |
| 2001004 | Sezai  | Kantar  | 702     | Veri Tabanı Yönetimi                 | 65   | 55    | 6    | Altan   | MESUT   |

### 1NF (Tekrarlı sütun yok, çoklu veri yok)

| Ö.NO    | Ö.AD   | Ö.SOYAD | DERS_NO | DERS_ADİ                             | VIZE | FINAL | H.NO | H.AD    | H.SOYAD |
|---------|--------|---------|---------|--------------------------------------|------|-------|------|---------|---------|
| 2001001 | Ahmet  | Solmaz  | 202     | Matematik 2                          | 70   | 60    | 11   | Özlem   | UÇAR    |
| 2001001 | Ahmet  | Solmaz  | 203     | Fizik 2                              | 80   | 40    | 11   | Özlem   | UÇAR    |
| 2001001 | Ahmet  | Solmaz  | 204     | Bilgisayar Mühendisliğine Giriş 2    | 60   | 45    | 3    | Aydın   | CARUS   |
| 2001001 | Ahmet  | Solmaz  | 205     | Atatürk İlkeleri ve İnkılap Tarihi 2 | 90   | 95    | 9    | Zeki    | DURMUŞ  |
| 2001001 | Ahmet  | Solmaz  | 206     | Türk Dili 2                          | 70   | 75    | 12   | Nebahat | YILDIZ  |
| 2001005 | Seyhan | Gülmez  | 202     | Matematik 2                          | 80   | 95    | 11   | Özlem   | UÇAR    |
| 2001005 | Seyhan | Gülmez  | 203     | Fizik 2                              | 80   | 70    | 11   | Özlem   | UÇAR    |
| 2001005 | Seyhan | Gülmez  | 204     | Bilgisayar Mühendisliğine Giriş 2    | 60   | 70    | 3    | Aydın   | CARUS   |
| 2001002 | Selim  | Solmaz  | 702     | Veri Tabanı Yönetimi                 | 60   | 50    | 6    | Altan   | MESUT   |
| 2001003 | Ahmet  | Vardar  | 702     | Veri Tabanı Yönetimi                 | 60   | 60    | 6    | Altan   | MESUT   |
| 2001004 | Sezai  | Kantar  | 702     | Veri Tabanı Yönetimi                 | 65   | 55    | 6    | Altan   | MESUT   |

### 2NF (Kısmi bağımlılıklar kaldırıldı)

| Ö.NO    | DERS_NO | VIZE | FINAL | Ö.NO    | Ö.AD   | Ö.SOYAD | DERS_NO | DERS_ADİ           | H.NO | H.AD    | H.SOYAD |
|---------|---------|------|-------|---------|--------|---------|---------|--------------------|------|---------|---------|
| 2001001 | 202     | 70   | 60    | 2001001 | Ahmet  | Solmaz  | 202     | Matematik 2        | 11   | Özlem   | UÇAR    |
| 2001001 | 203     | 80   | 40    | 2001005 | Seyhan | Gülmez  | 203     | Fizik 2            | 11   | Özlem   | UÇAR    |
| 2001001 | 204     | 60   | 45    | 2001002 | Selim  | Solmaz  | 204     | Bilg. Müh. Giriş 2 | 3    | Aydın   | CARUS   |
| 2001001 | 205     | 90   | 95    | 2001003 | Ahmet  | Vardar  | 205     | A.İ.İ.T 2          | 9    | Zeki    | DURMUŞ  |
| 2001001 | 206     | 70   | 75    | 2001004 | Sezai  | Kantar  | 206     | Türk Dili 2        | 12   | Nebahat | YILDIZ  |
| 2001005 | 202     | 80   | 95    |         |        |         | 702     | Veri Tabanı Yön.   | 6    | Altan   | MESUT   |
| 2001005 | 203     | 80   | 70    |         |        |         |         |                    |      |         |         |
| 2001005 | 204     | 60   | 70    |         |        |         |         |                    |      |         |         |
| 2001002 | 702     | 60   | 50    |         |        |         |         |                    |      |         |         |
| 2001003 | 702     | 60   | 60    |         |        |         |         |                    |      |         |         |
| 2001004 | 702     | 65   | 55    |         |        |         |         |                    |      |         |         |

### 3NF (Geçişli bağımlılıklar kaldırıldı)

| Ö.NO    | DERS_NO | VIZE | FINAL | Ö.NO    | Ö.AD   | Ö.SOYAD | H.NO | H.AD    | H.SOYAD |
|---------|---------|------|-------|---------|--------|---------|------|---------|---------|
| 2001001 | 202     | 70   | 60    | 2001001 | Ahmet  | Solmaz  | 11   | Özlem   | UÇAR    |
| 2001001 | 203     | 80   | 40    | 2001005 | Seyhan | Gülmez  | 3    | Aydın   | CARUS   |
| 2001001 | 204     | 60   | 45    | 2001002 | Selim  | Solmaz  | 9    | Zeki    | DURMUŞ  |
| 2001001 | 205     | 90   | 95    | 2001003 | Ahmet  | Vardar  | 12   | Nebahat | YILDIZ  |
| 2001001 | 206     | 70   | 75    | 2001004 | Sezai  | Kantar  | 6    | Altan   | MESUT   |
| 2001005 | 202     | 80   | 95    |         |        |         |      |         |         |
| 2001005 | 203     | 80   | 70    |         |        |         |      |         |         |
| 2001005 | 204     | 60   | 70    |         |        |         |      |         |         |
| 2001002 | 702     | 60   | 50    |         |        |         |      |         |         |
| 2001003 | 702     | 60   | 60    |         |        |         |      |         |         |
| 2001004 | 702     | 65   | 55    |         |        |         |      |         |         |

| DERS_NO | DERS_ADİ           | H.NO |
|---------|--------------------|------|
| 202     | Matematik 2        | 11   |
| 203     | Fizik 2            | 11   |
| 204     | Bilg. Müh. Giriş 2 | 3    |
| 205     | A.İ.İ.T 2          | 9    |
| 206     | Türk Dili 2        | 12   |
| 702     | Veri Tabanı Yön.   | 6    |

## Örnek-2

| SİPARİŞ NO | TARİH      | ÜRÜN                  | ÜRÜN NO | ADET | MÜŞ. NO | MÜŞ. AD | MÜŞ. SOYAD |
|------------|------------|-----------------------|---------|------|---------|---------|------------|
| 1          | 23.11.2007 | Nokia 6300            | 57463   | 1    | 875     | Ali     | Korkmaz    |
| 1          | 23.11.2007 | Kingston 2 GB USB     | 73624   | 2    | 875     | Ali     | Korkmaz    |
| 2          | 23.11.2007 | Samsung D600          | 72352   | 1    | 932     | Selin   | Atasoy     |
| 3          | 24.11.2007 | Nokia 5070            | 71224   | 1    | 123     | Kamil   | Sönmez     |
| 4          | 24.11.2007 | Philips DVP 5160/12   | 90876   | 1    | 452     | Metin   | Kaplan     |
| 5          | 25.11.2007 | Samsung Digimax S850  | 98123   | 1    | 786     | Kemal   | Durukan    |
| 6          | 25.11.2007 | Sinbo SBS-4414 Baskül | 35465   | 2    | 932     | Selin   | Atasoy     |
| 7          | 25.11.2007 | Canon Powershot A560  | 95293   | 1    | 875     | Ali     | Korkmaz    |
| 7          | 25.11.2007 | Kingston 2 GB SD      | 37285   | 1    | 875     | Ali     | Korkmaz    |
| 8          | 26.11.2007 | Nokia 6300            | 57463   | 1    | 321     | Ece     | Çağlayan   |

### 1 NF

| SİPARİŞ NO | TARİH      | ÜRÜN                  | ÜRÜN NO | ADET | MÜŞ. NO | MÜŞ. AD | MÜŞ. SOYAD |
|------------|------------|-----------------------|---------|------|---------|---------|------------|
| 1          | 23.11.2007 | Nokia 6300            | 57463   | 1    | 875     | Ali     | Korkmaz    |
| 1          | 23.11.2007 | Kingston 2 GB USB     | 73624   | 2    | 875     | Ali     | Korkmaz    |
| 2          | 23.11.2007 | Samsung D600          | 72352   | 1    | 932     | Selin   | Atasoy     |
| 3          | 24.11.2007 | Nokia 5070            | 71224   | 1    | 123     | Kamil   | Sönmez     |
| 4          | 24.11.2007 | Philips DVP 5160/12   | 90876   | 1    | 452     | Metin   | Kaplan     |
| 5          | 25.11.2007 | Samsung Digimax S850  | 98123   | 1    | 786     | Kemal   | Durukan    |
| 6          | 25.11.2007 | Sinbo SBS-4414 Baskül | 35465   | 2    | 932     | Selin   | Atasoy     |
| 7          | 25.11.2007 | Canon Powershot A560  | 95293   | 1    | 875     | Ali     | Korkmaz    |
| 7          | 25.11.2007 | Kingston 2 GB SD      | 37285   | 1    | 875     | Ali     | Korkmaz    |
| 8          | 26.11.2007 | Nokia 6300            | 57463   | 1    | 321     | Ece     | Çağlayan   |

### 2 NF (Kısmi bağımlılıklar giderildi)

| SİPARİŞ NO | ÜRÜN NO | ADET | ÜRÜN NO | ÜRÜN              | SİPARİŞ NO | TARİH      | MÜŞ. NO | MÜŞ. AD | MÜŞ. SOYAD |
|------------|---------|------|---------|-------------------|------------|------------|---------|---------|------------|
| 1          | 57463   | 1    | 57463   | Nokia 6300        | 1          | 23.11.2007 | 875     | Ali     | Korkmaz    |
| 1          | 73624   | 2    | 73624   | Kingston 2 GB USB | 2          | 23.11.2007 | 932     | Selin   | Atasoy     |
| 2          | 72352   | 1    | 72352   | Samsung D600      | 3          | 24.11.2007 | 123     | Kamil   | Sönmez     |
| 3          | 71224   | 1    | 71224   | Nokia 5070        | 4          | 24.11.2007 | 452     | Metin   | Kaplan     |
| 4          | 90876   | 1    | 90876   | Philips DVP ...   | 5          | 25.11.2007 | 786     | Kemal   | Durukan    |
| 5          | 98123   | 1    | 98123   | Samsung Digim...  | 6          | 25.11.2007 | 932     | Selin   | Atasoy     |
| 6          | 35465   | 2    | 35465   | Sinbo SBS-...     | 7          | 25.11.2007 | 875     | Ali     | Korkmaz    |
| 7          | 95293   | 1    | 95293   | Canon Powers...   | 8          | 26.11.2007 | 321     | Ece     | Çağlayan   |
| 7          | 37285   | 1    | 37285   | Kingston 2 GB SD  |            |            |         |         |            |
| 8          | 57463   | 1    |         |                   |            |            |         |         |            |

### 3 NF (Geçişli bağımlılıklar giderildi)

|            |         |      |         |                   |  |         |         |            |
|------------|---------|------|---------|-------------------|--|---------|---------|------------|
|            |         |      |         |                   |  | MÜŞ. NO | MÜŞ. AD | MÜŞ. SOYAD |
|            |         |      |         |                   |  | 875     | Ali     | Korkmaz    |
|            |         |      |         |                   |  | 932     | Selin   | Atasoy     |
|            |         |      |         |                   |  | 123     | Kamil   | Sönmez     |
|            |         |      |         |                   |  | 452     | Metin   | Kaplan     |
|            |         |      |         |                   |  | 786     | Kemal   | Durukan    |
|            |         |      |         |                   |  | 321     | Ece     | Çağlayan   |
| SİPARİŞ NO | ÜRÜN NO | ADET | ÜRÜN NO | ÜRÜN              |  |         |         |            |
| 1          | 57463   | 1    | 57463   | Nokia 6300        |  |         |         |            |
| 1          | 73624   | 2    | 73624   | Kingston 2 GB USB |  |         |         |            |
| 2          | 72352   | 1    | 72352   | Samsung D600      |  |         |         |            |
| 3          | 71224   | 1    | 71224   | Nokia 5070        |  |         |         |            |
| 4          | 90876   | 1    | 90876   | Philips DVP ...   |  |         |         |            |
| 5          | 98123   | 1    | 98123   | Samsung Digim...  |  |         |         |            |
| 6          | 35465   | 2    | 35465   | Sinbo SBS-...     |  |         |         |            |
| 7          | 95293   | 1    | 95293   | Canon Powers...   |  |         |         |            |
| 7          | 37285   | 1    | 37285   | Kingston 2 GB SD  |  |         |         |            |
| 8          | 57463   | 1    |         |                   |  |         |         |            |

|            |            |         |
|------------|------------|---------|
| SİPARİŞ NO | TARİH      | MÜŞ. NO |
| 1          | 23.11.2007 | 875     |
| 2          | 23.11.2007 | 932     |
| 3          | 24.11.2007 | 123     |
| 4          | 24.11.2007 | 452     |
| 5          | 25.11.2007 | 786     |
| 6          | 25.11.2007 | 932     |
| 7          | 25.11.2007 | 875     |
| 8          | 26.11.2007 | 321     |

## DELETE UPDATE RULES

### Delete Cascade

Bire çok ilişkiye sahip, foreign key ile bağlanmış iki tablodan bir'e sahip tablonun bir satırı silindiğinde çok'a sahip tablodaki, **foreign key'i** sütunun değeri olarak alan satırlar komple silinecek.

Örneğin id'si 2 olan cinsiyet silinirse, cinsiyeti 2 olan müşteriyi de siler.

### No Action

Bire çok ilişkiye sahip, foreign key ile bağlanmış iki tablodan bir'e sahip tablonun bir satırı silindiğinde çok'a sahip tablodaki, foreign keyi sütunun değeri olarak alan satırlar silinemeyecek. SQL hata döndürecektir. "Bu tablo, silmek istediğin sütunu foreign key olarak kullanıyor", silemezsin, demektir. Çünkü eğer silersen kesinlikle veri kaybı olacak.

Örneğin id'si 2 olan cinsiyet silinirse, hata verecektir.

### Set Default

Bire çok ilişkiye sahip, foreign key ile bağlanmış iki tablodan bir'e sahip tablonun bir satırı silindiğinde çok'a sahip tablodaki, foreign keyi sütunun değeri olarak alan satırlardaki ilgili sütun, o sütunun default değeri ile doldurulacak.

Örneğin id'si 2 olan cinsiyet silinirse, cinsiyeti 2 olan müşterinin cinsiyet sütununu, cinsiyet sütunun default değeriyle yani 3 ile doldurur.

### Set NULL

Bire çok ilişkiye sahip, foreign key ile bağlanmış iki tablodan bir'e sahip tablonun bir satırı silindiğinde çok'a sahip tablodaki, foreign key'i sütunun değeri olarak alan satırlardaki ilgili sütunun değeri NULL olarak ayarlanacak.

Örneğin id'si 2 olan cinsiyet silinirse, cinsiyeti 2 olan müşterinin cinsiyet sütununu, NULL olarak ayarlar.

## CONSTRAINED (KISIT)

### Kısıt Ekleme

```
ALTER TABLE TblUrunler ADD CONSTRAINT
CK_TblUrunler_FiyatKontrol CHECK(fiyat>0)
-- fiyat sutununun 0'dan büyük olması gerektiğini söyledik.
-- 'Alter' komutunu kullanıyoruz çünkü tablo üzerinde değişiklik yapacağız.
-- 'CK' ifadesi 'constraint key' anlamına gelmektedir.
```

### Kısıt Silme

```
ALTER TABLE TblUrunler DROP CONSTRAINT CK_TblUrunler_FiyatKontrol
-- Kısıtı bu şekilde kaldırırız.
```

### Default Kısıt Ekleme

```
ALTER TABLE TblMusteriler ADD CONSTRAINT
DF_TblMusteriler_cinsId DEFAULT 3 FOR cinsId
-- 'DF' ifadesi bu kısıtın, DEFAULT bir kısıt olduğunu belirtiyor.
-- 'default 3 for cinsId' ifadesi; cinsId sütununa default olarak 3 değerini atar.
-- Eğer cinsId belirtilmesse cinsId, default olarak 3 olur.
```

```
insert into TblMusteriler (isim, soyisim, mail)
values('Veli', 'ATAK', 'va@com.tr')
-- Veli ATAK için cinsiyet girilmediği için cinsId alanı default olarak 3 olacak.
```

## VIEW TABLE (SANAL TABLO)

- Bir ya da birden fazla tablonun istenilen bilgilerini birleştirerek sanal bir tablo oluşturup bu tablodan sorgu yapılabilmesini mümkün kılar.
- Normal tablolar gibi, **Tables klasörü** altında bulunmaz. Kendisine özel **Views klasöründe** bulunur.
- **AS** ifadesinden sonra yapmak istenilen sorgu direkt olarak yazılır.
- Ana tabloda yapılan değişiklikler sanal tabloyu, sanal tabloda yapılan değişiklikler ana tabloyu etkiler.

### Sanal Tablo Oluşturma

```
CREATE VIEW vwKategoriUrunSayisi AS
SELECT TblUrunKategori.kategori, COUNT(*) AS URUN_SAYISI
FROM TblUrunKategori INNER JOIN TblUrunler
ON TblUrunKategori.id=TblUrunler.urun_kategori_id
GROUP BY TblUrunKategori.kategori
-- COUNT ile NULL degerler getirilmedi.
-- Urun kategorileri ve urun sayisi getirildi.
-- Artık sadece vwKategoriUrunSayisi ismiyle calistirabiliriz.
```

```
select * from vwKategoriUrunSayisi
```

|   | kategori          | URUN_SAYISI |
|---|-------------------|-------------|
| 1 | BİSKÜVİ           | 2           |
| 2 | GIDA              | 3           |
| 3 | SARKUTERİ         | 2           |
| 4 | TEMİZLİK ÜRÜNLERİ | 1           |

### Sanal Tablo Silme

```
DROP VIEW vwKategoriUrunSayisi
-- Sanal tabloyu şifreli olsa dahi siler.
```

### sp\_helptext Komutu

```
sp_helptext vwKategoriUrunSayisi
-- Sanal tablo'nun hangi kodlarla olusturulduğunu gösterir.
-- SYSTEM STORED PROCEDURE = sp'nin acilimi
```

### View Güncelleme

```
alter view vwPersonelGetir
as
select tblPersonelDurum.ID,tblPersonelDurum.Ad,
tblMedeniDurum.Medeni,tblSehir.Sehir
from tblPersonelDurum
JOIN tblSehir on tblPersonelDurum.SehirID=tblSehir
```

```
UPDATE TblUrunKategori SET kategori = 'MANAV' WHERE kategori = 'SARKUTERİ'
-- İndeksli işlem olduğu için tablo güncellendi.
-- İndeksiz olsaydı güncellemeler sanal tabloya etki etmezdi.
-- SARKUTERİ kategorisini silip MANAV kateogiri eklendi.
```

|   | kategori          | URUN_SAYISI |
|---|-------------------|-------------|
| 1 | BİSKÜVİ           | 2           |
| 2 | GIDA              | 3           |
| 3 | MANAV             | 2           |
| 4 | TEMİZLİK ÜRÜNLERİ | 1           |

```
insert into TblUrunKategori values ('ZUCCACIYE') -- VIEW'e de ekleniyor.
insert into TblUrunler values ('Çay Bardağı',11.25,106,4) -- VIEW'e de ekleniyor.
```

```
insert into vwKategoriUrunSayisi values ('OYUNCAK',5)
-- Bağlı olduğu için dolayısıyla ilişkisel bir yapı içinde olduğu için izin vermez.
UPDATE vwKategoriUrunSayisi SET kategori = 'TEMEL GIDA' WHERE id=2
-- Bağlı olduğu için dolayısıyla ilişkisel bir yapı içinde olduğu için izin vermez.
```

## View Şifreleme

```
118 | Create view vwSifreliView With ENCRYPTION
119 | as
120 | select tblPersonelDurum.ID,tblPersonelDurum.Ad,
121 | tblMedeniDurum.Medeni,tblSehir.Sehir
122 | from tblPersonelDurum
123 | JOIN tblSehir on tblPersonelDurum.SehirID=tblSehir.ID
124 | JOIN tblMedeniDurum on tblPersonelDurum.MedeniID=tblMedeniDurum.ID
```

View oluşturulduktan sonra baktığımızda, **view ikonunun yanında kilit işareti** görülecektir.

**NOT:** BEGIN – END; if’de, while’da, trigger’da kullanılır. Başlangıç ve bitiş noktalarını gösterir.

## STORED PROCEDURE (FONKSİYON)

Sık kullandığımız işlemleri fonksiyon şeklinde veritabanında tanımlamaya **stored procedure** denir. Parametre tanımlarken değişkenin başına "@" işareti koymamız yeterlidir.

### Stored Procedure (Fonksiyon) Ekleme

Aşağıdaki sorgumuzda kullanıcının @persAd girdisine yazdığı değer sonucuna eşit olan kayıt getirilir.

```
CREATE PROCEDURE upPersonelGetir _CinseYasaGore
@persAd varchar(50)
AS
BEGIN
select * from tblPersonel where ad=@persAd
END
```

```
EXEC upPersonelGetir 'Ramazan'
```

|   | ID | Ad      | Soyad | Numara |
|---|----|---------|-------|--------|
| 1 | 4  | Ramazan | Şeker | 5004   |

### Stored Procedure (Fonksiyon) Güncelleme

ALTER, procedur’de değişiklik yapmamızı sağlar.

```
ALTER PROCEDURE upMusteriGetir_CinseYasaGore
@cins int,      --Girdi Parametresi
@yas int        --Girdi Parametresi
AS
BEGIN
select * from TblMusteriler where cinsiyetId=@Cins AND yas <@Yas
END
-- USER PROCEDURE = up'nin acilimi
```

Buradaki @cins ve @yas girdi parametresidir yani kullanıcıdan alınan verilerdir.

### Stored Procedure (Fonksiyon) Çalıştırma

```
EXEC upMusteriGetir_CinseYasaGore 2,50
-- Sorguda erkek ve 50 yasından küçük olan kisiler gelir.
```

### Stored Procedure (Fonksiyon) Silme

```
DROP PROC up_MusteriGetir -- Procedur'u (fonksiyonu) sileriz.
```

### sp\_helptext Komutu

```
sp_helptext upMusteriGetir_CinseYasaGore
-- Fonksiyonun hangi kodlarla oluşturulduğunu gösterir.
-- SYSTEM STORED PROCEDURE = sp'nin acilimi
```

### Stored Procedure (Fonksiyon) Şifreleme

```
CREATE PROCEDURE upMusteriGetir_CinseYasaGore_Sifre
@cins int,      --Girdi Parametresi
@yas int       --Girdi Parametresi
WITH ENCRYPTION -- Bu sayede komut icerigi sifreli oluyor.
AS
BEGIN
select * from TblMusteriler where cinsiyetId=@Cins AND yas <@Yas
END
```

```
sp_helptext upMusteriGetir_CinseYasaGore_Sifre
-- Sifreledik bu yuzden kodlari goremeyiz.
```

### Return Stored Procedure (Geriye Değer Döndüren Fonksiyon)

```
CREATE PROCEDURE up_CinsiyeteGoreCalisanSayisi
@cins int ,
@calisansayi int OUTPUT -- Geriye döndürecegi deger (RETURN)
AS
BEGIN
select @calisansayi = count(*) from TblMusteriler where cinsiyetId = @cins
END

EXEC up_CinsiyeteGoreCalisanSayisi -- Bu sekilde dogrudan calistirilmaz
DECLARE @toplamMusteriSayisi int exec up_CinsiyeteGoreCalisanSayisi 2, @toplamMusteriSayisi OUTPUT
print @toplamMusteriSayisi -- OUTPUT, RETURN degerimiz, print ekrana yazdirmek için.
-- Geriye dönecek degeri calistirabilmek için bu uc satirida secmemiz gerekiyor.
-- Cinsiyet degeri 2 olan sahislar gelir bu sorgu sonucunda.
-- DECLARE ile ilgili bir sey tanımladigimizda calistiracagimiz zaman DECLARE'yi de secmeliyiz.
```

### Store Procedure Kullanılma Amaçları ve Avantajları

- ❖ Sıklıkla kullanılan sorguları tekrar tekrar yazmamak. Kodlamada hız sağlar.
- ❖ Kod, tekrar kullanılabilir olur.
- ❖ Sorguları procedure şeklinde veritabanında depolamak güvenliği sağlar. Kimse kodları göremeyecektir. Sadece (parametre göndererek) sorgu çalıştırır ve sonuç alır.
- ❖ SQL Server sorgusu 3 aşamadan geçer:
  - Sorğunun syntax'ında bir hata var mı?
  - Sorgu compile edilir.
  - Execution plan
- ❖ Stored Procedure gerçek tablolar üzerinde işlem yapar.
- ❖ Stored Procedure içinde herhangi bir CRUD işlemi yapılabilir.

### Stored Procedure (Fonksiyon) Şifreleme

```
CREATE PROCEDURE upMusteriGetir_CinseYasaGore_Sifre
@cins int,      --Girdi Parametresi
@yas int       --Girdi Parametresi
WITH ENCRYPTION -- bu sayede komut icerigi sifreli oluyor
AS
-- sp_helptext ile kodlari gormek istersek goremeyiz çünkü sifreli
BEGIN
select * from TblMusteriler where cinsiyetId=@Cins AND yas <@Yas
END

sp_helptext upMusteriGetir_CinseYasaGore_Sifre -- Sifreledik, bu yuzden kodlar gözükmmez.
```

```

DECLARE @start int
SET @start = 65
while(@start<=70)
BEGIN
    print char(@start)
    SET @start = @start + 1
END
-- @start diye bir degisken olusturulur ve ilk deger olarak 65 atanir.
-- While dongusu acilarak degisken, 90'dan kucuk esittir oldukca devam edecek.
-- BEGIN ve END arasindaki bloga islemleri yazdik.
-- Print ile ekrana @start degiskeninin karakter karsiligini yazdik. Degiskeni
-- bir arttirdik. Boylece a'dan z'ye kadar butun karakterleri yazmis oluruz.
-- DECLARE ILE OLUSTURULAN DEGISKEN ANLIK OLUSTURULUR. BIR YERDE CALISTIRMAK ICIN --
-- VEYAHUT KULLANMAK ICIN DECLARE'YI SECMEMIZ LAZIM.
-- Cikti olarak A, B, C, D, E, F degerlerini elde ederiz.

select ltrim('    Merhaba')
-- Ifadenin sadece solundaki bosluklari siler.

select rtrim('Merhaba    ')
-- Ifadenin sagindaki bosluklari siler.

select lower ('MeRhAbA')
-- Aldigi ifadenin tum harflerini kucuk harfe ceviris.

select upper('MeRhAbA')
-- Aldigi ifadenin tum harfleri buyuk harfe ceviris.

select len('Merhaba')
-- Icine aldigi stringin uzunlugunu verir. SONUC: 7

select len('Merhaba    ')
-- Eger ifadenin saginda bosluklar varsa bosluklari saymayacaktır. SONUC: 7

select len('    Merhaba')
-- Eger ifadenin solunda bosluklar varsa bosluklari da karakterden sayar. SONUC: 10

select REVERSE('Merhaba')
-- Ifadeyi tersine ceviris. SONUC: abahreM

select left('Necmettin Erbakan Üniversitesi', 5)
-- Stringin ilk 5 karakterini getirecektir. SONUC: "Necme"

select right('Necmettin Erbakan Üniversitesi', 3)
-- Stringin son 3 karakterini getirecektir. SONUC: "esi"

select charindex('@', 'saidtaylan@gmail.com')
-- Ikinci parametre olarak aldigi string icinde, birinci parametre olarak aldigi
-- karakteri arar. ilk eslesen karakterin index degerini getirir.
-- SQL'de indexler 1'den baslar. SONUC: 11

select substring('saidtaylan@gmail.com', 5, 9)
-- Ikinci ve ucuncu parametre olarak aldigi sayilardan
-- birincisi baslangic degeri, ikincisi ise kac karakter sececegidir.
-- Bu ornekte 5. karakter 't' oldugu icin 'taylan@gm' ifadesini getirecek.
-- Baslangic ve bitis index'leri getirilecek ifadeye dahildir.
select len(ltrim(rtrim(isim))),ltrim(rtrim(soyisim)) from KISILER
-- Isim ve soyisimlerin sagindaki ve solundaki bosluklar giderildi.

select upper(ltrim(isim))+ ' '+lower(ltrim(soyisim)) as ISIM_SOYISIM from KISILER
-- Isimler buyuk, soyisimler kucuk harfle yazilip birlestirildi.

```



## TRIGGER (TETİKLEYİCİLER)

TRIGGER yapısı bir tabloda belirli olaylar meydana geldiğinde veya gelmeden önce otomatik olarak çalışan özel bir Stored Procedure türüdür.

Bir tabloda ekleme, güncelleme ve silme işlemlerinden biri gerçekleştiğinde veya gerçekleşmeden önce, aynı tabloda veya başka bir tabloda belirli işlemlerin yapılmasını istediğimizde, TRIGGER yapısını kullanırız.

## COMMIT TRANSACTION

Transaction, süreci hatasız tamamlandığında, SQL Server'ın işlemi sonlandırmasını, değişiklikleri kalıcı hale getirmesini sağlamaktadır.

## ROLLBACK TRANSACTION

Başarısız bir transaction'ı sonlandırmak için ROLLBACK komutu kullanılır. ROLLBACK ile transaction'ın yaptığı değişiklikler geri alınarak veriler transaction başladığındaki ilk haline geri döndürülür.

Rollback işlemi, oluşturulan nesneleri, eklenen, silinen ve güncellenen satırlar gibi yapılan değişiklikler geri alır.

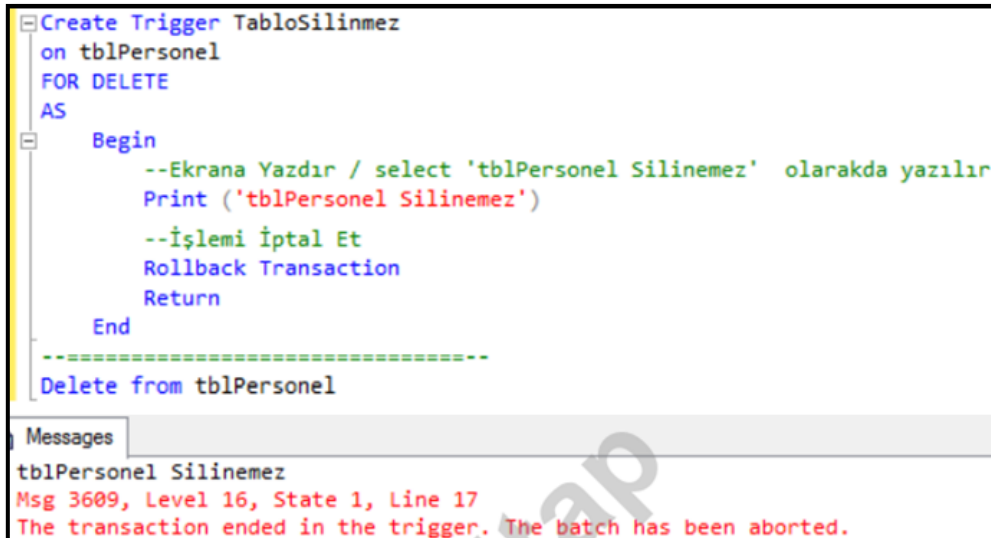
SQL'de 3 tip TRIGGER vardır:

1. **DML Trigger (insert, update, delete)**
  - After trigger (for trigger)
  - Instead of trigger
2. **DDL Triggers: CREATE, ALTER, DROP ifadeleri kullanıldığında devreye giren trigger'dır.**
3. **Logon Triggers (Veritabanına girişte güvenliği denetleyen trigger)**

```
CREATE TRIGGER trg_TblUrunler_forInsert
ON tblUrunler          -- on ile hangi tabloda işlem yapacağımızı belirtiyoruz.
FOR INSERT              --insert işlemi gerçekleşince BEGIN ile END arasındaki kodlar
AS                      -- tetiklenecek yani çalışacak.
BEGIN
    select * from inserted
END
```

```
insert into TblUrunler VALUES('makarna', 10.95, 106, 1)
-- Bize sonuc olarak eklenen veriler (sadece makarna) getirilecek.
```

Bu örneğimizde tanımladığımız tablomuzu silmek isteyeceğiz. Delete komutunu çalıştırdığımızda, tablomuzu silme işlemini iptal edeceğiz.



```
Create Trigger TabloSilinmez
on tblPersonel
FOR DELETE
AS
Begin
    --Ekran Yazdır / select 'tblPersonel Silinmez' olarakda yazılır
    Print ('tblPersonel Silinmez')
    --İşlemi İptal Et
    Rollback Transaction
    Return
End
Delete from tblPersonel
```

Messages

tblPersonel Silinmez  
Msg 3609, Level 16, State 1, Line 17  
The transaction ended in the trigger. The batch has been aborted.

Trigger oluşturulurken trigger işleminin, yapılan tetikleyicinin **öncesinde (Before)** veya **sonrasında (After)** yapılmasını ayarlayabiliriz.

```
Create trigger SilineniAktar
ON tblUrunListe AFTER DELETE --Veri silindikten sonra
As
Begin
    Declare @UrunKod varchar(30),@UrunAd varchar(30),@Fiyat int
    Select @UrunKod=UrunKod,@UrunAd=UrunAd,@Fiyat=Fiyat
    from deleted
    insert into tblSilinenUrun values (@UrunKod,@UrunAd,@Fiyat)
End

Delete from tblUrunListe Where ID=2

select * from tblUrunListe
select * from tblSilinenUrun
```

|   | ID | UrunKod    | UrunAd    | Fiyat |
|---|----|------------|-----------|-------|
| 1 | 1  | 8969885788 | Buzdolabı | 850   |
| 2 | 11 | 5568750245 | Tablet    | 750   |

|   | ID | UrunKod    | UrunAd | Fiyat |
|---|----|------------|--------|-------|
| 1 | 1  | 5568750245 | Tablet | 750   |
| 2 | 2  | 5568750245 | Tablet | 750   |
| 3 | 3  | 5568777458 | Laptop | 1250  |
| 4 | 4  | 6985477745 | Gömlek | 75    |

Tablolarımızda veriyi güncelleme durumunda, yapılmasını istediğimiz işlemlerde kullanılır. tblUrunListe tablomuzdaki verimizi güncelleyelim. Güncellenen verinin eski hâli tblGuncellenenUrun tablosuna eklenecektir.

```
Create trigger GuncellenenVeri
ON tblUrunListe AFTER UPDATE
As
Begin
    Declare @UrunKod varchar(30),@UrunAd varchar(30),@Fiyat int
    Select @UrunKod=UrunKod,@UrunAd=UrunAd,@Fiyat=Fiyat
    from deleted
    insert into tblGuncellenenUrun values (@UrunKod,@UrunAd,@Fiyat)
End

update tblUrunListe Set UrunAd='Yeni Tablet' Where ID=11

select * from tblUrunListe
select * from tblGuncellenenUrun
```

|   | ID | UrunKod    | UrunAd      | Fiyat |
|---|----|------------|-------------|-------|
| 1 | 1  | 8969885788 | Buzdolabı   | 850   |
| 2 | 11 | 5568750245 | Yeni Tablet | 750   |

|   | ID | UrunKod    | UrunAd | Fiyat |
|---|----|------------|--------|-------|
| 1 | 1  | 5568750245 | Tablet | 750   |

Aşağıdaki TRIGGER olayını gerçekleştirdiğimizde, tabloyu silmek istesek de silme işlemi gerçekleşmeyecektir.

```
Create Trigger TabloSilinemez
On DATABASE --Veritabanı Düzeyinde Trigger
FOR Drop_Table
As
Rollback --İşlemi Durdur

Drop Table tblPersonel
```

Messages

Msg 3609, Level 16, State 2, Line 88  
The transaction ended in the trigger. The batch has been aborted.

### T-SQL

```
DECLARE @sayi INT
SELECT @sayi = COUNT(*) from TblUrunler

while @sayi>=1
BEGIN
    print @sayi
    set @sayi= @sayi -1
END
-- TblUrunler tablosundan urunlerin sayisini @sayi degiskenine aktardik. SONUC: 8
-- While islemi, 8'den baslayarak 0'a kadar sayilari ekrana yazdirir.
-----

DECLARE @sayi INT
SELECT @sayi = COUNT(*) from TblUrunler
-- Degisken olusturduk. tblurunlerdeki urun sayisini @sayi degiskenine atadik.
if @sayi>1
BEGIN
    print '1 den fazla kayıt var. Kayıt sayısı: ' + CAST(@sayi AS varchar(5))
    -- CAST yani donusturma islemi, degiskeni burada stringe cevirdik.
END
else
BEGIN
    print 'Kayıt yok!'
END
-----

DECLARE @ay INT
-- Switch Case yapısıyla degiskendeki sayiyi degistirerek hangi aydayız
gorebiliyoruz, degerlere aylarin ismini atadik
set @ay =5
select case @ay
when 1 then 'OCAK'
when 2 then 'SUBAT'
when 3 then 'MART'
when 4 then 'NISAN'
when 5 then 'MAYIS'
when 6 then 'HAZIRAN'
when 7 then 'TEMMUZ'
when 8 then 'AGUSTOS'
when 9 then 'EYLUL'
when 10 then 'EKIM'
when 11 then 'KASIM'
when 12 then 'ARALIK'
else 'Cikmaz ay'
end
```

```

SELECT TblUrunKategori.kategori, COUNT(*) AS URUN_SAYISI
FROM TblUrunKategori INNER JOIN TblUrunler
ON TblUrunKategori.id=TblUrunler.urun_kategori_id
GROUP BY TblUrunKategori.kategori
-- COUNT ile NULL degerler getirilmedi.
-- Urun kategorileri ve urun sayisi getirildi.
-- Artık sadece vwKategoriUrunSayisi ismiyle calistirabiliriz.

select * from vwKategoriUrunSayisi
-----

CREATE PROCEDURE upMusteriGetir_CinseYasaGore_Sifre
@cins int,      --Girdi Parametresi
@yas int        --Girdi Parametresi
@calisansayi int OUTPUT -- Geriye döndürecegi deger (RETURN)
WITH ENCRYPTION -- Bu sayede komut icerigi sifreli oluyor.
AS
BEGIN
select * from TblMusteriler where cinsiyetId=@Cins AND yas <@Yas
END

sp_helptext upMusteriGetir_CinseYasaGore_Sifre
-- Sifreledik bu yuzden kodlari goremeyiz.
-----

DECLARE @sayi INT
SELECT @sayi = COUNT(*) from TblUrunler
-- Degisken olusturduk. tblurunlerdeki urun sayisini @sayi degiskenine atadik.
if @sayi>1
    BEGIN
        print '1 den fazla kayit var. Kayit sayisi: ' + CAST(@sayi AS varchar(5))
        -- CAST yani donusturme islemi, degiskeni burada stringe cevirdik.
    END
else
    BEGIN
        print 'Kayit yok!'
    END
-----

CREATE TRIGGER trg_TblUrunler_forInsert
ON tblUrunler      -- on ile hangi tabloda islem yapacagimizi belirtiyoruz.
FOR INSERT          --insert islemi gerceklesince BEGIN ile END arasindaki kodlar
AFTER DELETE        -- tetiklenecek yani calisacak.
AS
BEGIN
    select * from inserted
END

insert into TblUrunler VALUES('makarna', 10.95, 106, 1)
-- Bize sonuc olarak eklenen veriler (sadece makarna) getirilecek.
-----

DECLARE @ay INT
-- Switch Case yapisiyla degiskendeki sayiyi degistirerek hangi aydayiz
gorebiliyoruz, degerlere aylarin ismini atadik
set @ay =3
select case @ay
when 1 then 'OCAK'
when 2 then 'SUBAT'
when 3 then 'MART'
when 4 then 'NISAN'
when 5 then 'MAYIS'
else 'Cikmaz ay'
end

```