# BMB202. Veritabanı Yönetimi

## Ders 6.
## SQL

Erdinç Uzun
NKÜ Çorlu Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

# Dersin Planı

- SQL Fonksiyonları
- Çoklu Tablo İşlemleri
  - İç içe Select'ler
  - JOIN
  - Birleştirme
- DML
- DDL

# Örnek Veritabanı "Öğrenci İşleri"

- OGRENCILER Tablosu:
  - OGR_NO (PK)
  - AD
  - SOYAD
  - TEL
  - ADRES
- DERSLER Tablosu:
  - DERS_KODU (PK)
  - DERS_ADI
  - DERS_KREDISI
  - HOCASI (FK-HOCALAR)
- NOTLAR Tablosu:
  - OGRENCI (FK-OGRENCILER)
  - DERS (FK-DERSLER)
  - DERS_YILI
  - VIZE
  - FINAL
- HOCALAR Tablosu:
  - HOCA_NO (PK)
  - AD
  - SOYAD
  - UNVAN

# SQL Fonksiyonları

- Programlama dillerinde olduğu gibi, SQL'de de bazı aritmetik işlemler için yada tip dönüşümü yapmak için hazır olarak sunulan fonksiyonlar mevcuttur.
- Bu fonksiyonların bazıları (SUM, AVG, MIN, MAX, …) birçok kayıt üzerinde işlem yapıp tek bir sonuç üretirken, bazıları ise (Örneğin; tip dönüşümü ile ilgili olanlar) üzerinde işlem yaptığı her kayıt için ayrı sonuç üretir.

# SUM

- Belirli bir sütundaki sayısal verilerin toplanarak sonucun gösterilmesi istenirse SUM, aritmetik ortalamasının gösterilmesi istenirse AVG kullanılır.
- Aşağıdaki SQL cümlelerinden ilki tüm personelin maaşlarının toplamını, ikincisi ise maaşların aritmetik ortalamasını gösterir:
  - SELECT SUM(MAAS) AS TOPLAM_MAAS FROM PERSONEL
  - SELECT AVG(MAAS) AS [MAASLARIN ORTALAMASI] FROM PERSONEL

NOT: _ karakteri yerine boşluk karakteri kullanılması istenirse [] içinde yazılmalıdır.

# MIN ve MAX

- Belirli bir sütundaki en büyük veriyi görüntülemek için MAX, en küçük veriyi görüntülemek için ise MIN fonksiyonları kullanılır.
- Aşağıdaki SQL cümlesi, “2009-2010” öğretim yılında “BM 316” dersinden en yüksek başarı notunu gösterir:
  - SELECT MAX(VIZE*0.3 + FINAL*0.7)

    FROM NOTLAR

    WHERE DERS = “BM 316” AND DERS_YILI = “2009-2010”

# COUNT

- Sorgunun ürettiği satır sayısını döndürür.
- Aşağıdaki SQL cümlesi PERSONEL tablosundaki toplam kayıt sayısını döndürür:

```
SELECT COUNT(*) FROM PERSONEL
```

- Eğer COUNT içinde * yerine belirli bir sütun ismi verilirse o sütundaki NULL olmayan değer sayısını döndürür:

```
SELECT COUNT(ADRES) FROM OGRENCILER
```

# TOP

- Önceki örnekte en yüksek başarı notunu alan öğrencinin numarasını da göstermek istersek aşağıdaki kullanım hata verecektir:
  - SELECT OGRENCI, MAX(VIZE*0.3 + FINAL*0.7)
    FROM NOTLAR
    WHERE DERS = “BM 316” AND DERS_YILI = “2009-2010”
- SELECT sonrasında “TOP n” kullanımı en üstteki n kaydı gösterir. Örneğimizde başarı notuna göre azalan sıralama yapıp en üstteki kaydı almak için “TOP 1” deyimini kullanmak en uygun çözümdür:
  - SELECT TOP 1 OGRENCI, VIZE*0.3 + FINAL*0.7 AS Başarı_Notu
    FROM NOTLAR
    WHERE DERS = “BM 316” AND DERS_YILI = “2009-2010”
    ORDER BY VIZE*0.3 + FINAL*0.7

NOT: Bu ifade yerine takma ismi olan Başarı_Notu kullanılması hata verir

# LCASE, UCASE ve LEN

- LCASE: Tüm karakterleri küçük harfe dönüştürür. (Oracle'da LOWER)
- UCASE: Tüm karakterleri büyük harfe dönüştürür. (Oracle'da UPPER)
- LEN: Sütun yada ifade içindeki karakter sayısını döndürür (Oracle'da LENGTH)
- Aşağıdaki SQL cümlesi SOYAD bilgileri küçük harfli bile girilmiş olsa tüm karakterleri büyük harf olarak görüntüler:
  - SELECT AD, LCASE(SOYAD) FROM PERSONEL

# Tip Dönüşümleri

- Access'te tip dönüşümleri için aşağıdaki fonksiyonlar kullanılır:

  CBool CDate CInt CStr

  CByte CDbl CLng CVar

  CCur CDec Csng

- Diğer VTYS içinde tip dönüşümü mümkündür.
- Fakat, Tip Dönüşümü pek tavsiye edilmez.

# Tarihsel Fonksiyonlar

## Access

- NOW: Sistem tarihini ve saatini döndürür.
- DATEDIFF: İki tarih arasındaki farkı verir.
- DATEADD: Aldığı tarihin üzerine aldığı değeri (gün, ay, yıl) ekleyerek yeni bir tarih değeri üretir.
- DAY: Aldığı tarihin gün kısmını döndürür.
- MONTH: Aldığı tarihin gün kısmını döndürür.
- YEAR: Aldığı tarihin gün kısmını döndürür.

- Programlama Dili tarafında da yönetilebileceği unutulmamalıdır.

# GROUP BY

- SUM, AVG gibi bazı fonksiyonların tablonun tamamı için değil de, belirli bir alana (yada alanlara) göre gruplandırılarak çalıştırılması GROUP BY deyimi ile sağlanabilir.
- Aşağıdaki SQL cümlesi personelin ortalama maaşlarını her bölüm için ayrı ayrı listeler:

```
SELECT AVG(MAAS) FROM PERSONEL
GROUP BY BOLUM
```

# HAVING

- HAVING ifadesinin işlevi WHERE ifadesininkine çok benziyor. Ancak kümeleme fonksiyonları ile WHERE ifadesi birlikte kullanılamadığından HAVING ifadesine ihtiyaç duyulmuştur.
- Maaşı 3000'den büyük maaşlar için
  - SELECT * FROM PERSONEL

    WHERE MAAS > 3000
- Maaş ortalaması 1500'den fazla görevlerin görüntülenmesi
  - SELECT Gorev, AVG(maas) FROM PERSONEL GROUP BY Gorev HAVING AVG(Maas) > 1500

# Çok tablolu sorgulamalar

- Eğer birden fazla tabloda yer alan verilerin tek bir sorgu ile görüntülenmesi istenirse FROM kısmında ilgili tablolar araya virgül konularak yazılmalı, WHERE kısmında ise o tabloları birbirine bağlayan alanların birbirine eşit olması kriteri verilmelidir. WHERE kısmında böyle bir kriter verilmezse, iki tablonun tüm kayıtları birbiri ile eşleştirilecek (kartezyen çarpım) ve ortaya istenilenden daha çok sayıda kayıt çıkacaktır.
- Kartezyen çarpımı için “İlişkisel Çebir” konusuna bak.

# 2 tablolu sorgulama örneği

- PERSONEL tablosundaki BOLUM (FK) alanı ile BOLUMLER tablosundaki BOLUM_NO (PK) alanı birbirine bağlıdır. Eğer personel bilgileri görüntülenirken personelin çalıştığı bölümün numarasını değil de adını göstermek istersek bu adı BOLUMLER tablosundan elde etmeliyiz. Bu nedenle iki tablonun ismini de FROM kısmında kullanmalıyız:
  - SELECT AD, SOYAD, GOREV, BOLUM_ADI
    FROM PERSONEL, BOLUMLER
    WHERE BOLUM = BOLUM_NO

NOT: Bu satır yazılmadığında görüntülenen kartezyen çarpıma dikkat!

# Alan isimlerinin aynı olması durumu

- Eğer bir çok tablolu sorgulamada ilişkinin her iki tarafındaki alanlar aynı isimde ise; alan isimlerinden önce, o alanın ait olduğu tablo ismi de yer almalıdır.
- Örneğin PERSONEL tablosundaki alan ismi de BOLUM_NO olsaydı sorgu şu şekilde yazılmalıydı:

```
SELECT AD, SOYAD, GOREV, BOLUM_ADI
FROM PERSONEL, BOLUMLER
WHERE PERSONEL.BOLUM_NO =
BOLUMLER.BOLUM_NO
```

- “Takma İsim” Kullanarak

```
SELECT AD, SOYAD, GOREV, BOLUM_ADI
FROM PERSONEL P, BOLUMLER B
WHERE P.BOLUM_NO = B.BOLUM_NO
```

# 3 tablolu sorgulama örneği

- Öğrencilerin adı, soyadı, aldıkları derslerin adı ve bu derslerden başarı notları görüntülenmek istenirse:
  - SELECT AD, SOYAD, DERS_ADI, VIZE*0.3 + FINAL*0.7 AS BN
    FROM OGRENCILER, NOTLAR, DERSLER
    WHERE OGR_NO = OGRENCI AND DERS = DERS_KODU

# İç içe SELECT ifadesi

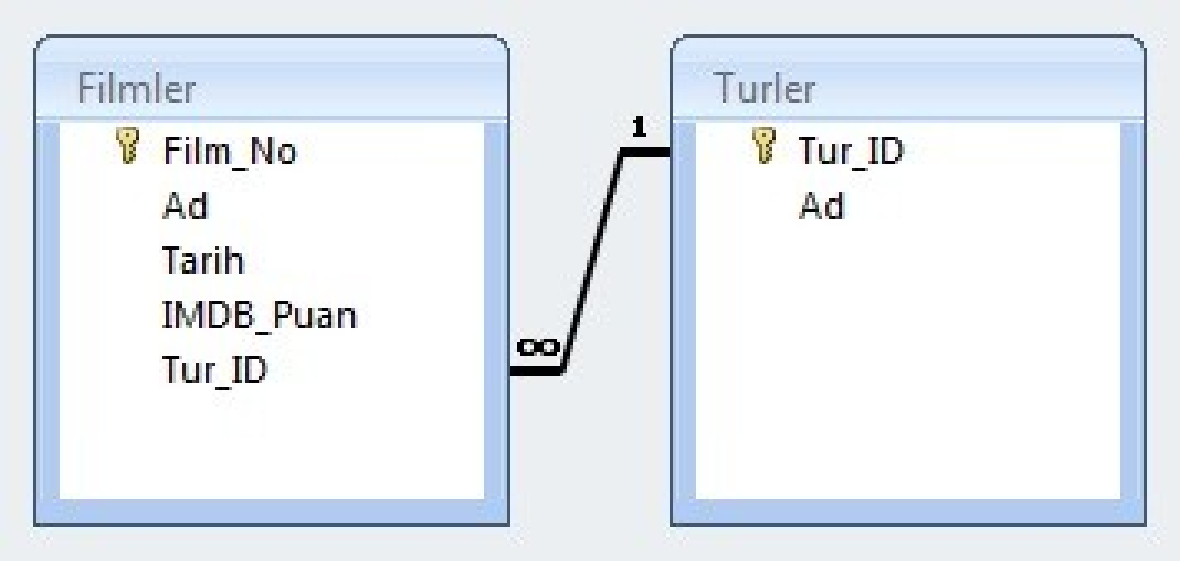
- Eğer SELECT sonrasında yazılan alanların hepsi aynı tabloda, fakat kriter olarak kullanılan alanlar onlardan farklı bir tabloda ise iç içe SELECT ifadeleri kullanılabilir.
- Aşağıdaki SQL cümlesi ‘Fizik’ dersini alan öğrencilerin bilgilerini gösterir:
  - SELECT OGR_NO, AD, SOYAD FROM OGRENCILER
    WHERE OGR_NO IN (
    SELECT OGRENCI FROM NOTLAR
    WHERE DERS IN (
    SELECT DERS_KODU FROM DERSLER
    WHERE DERS_ADI = ‘Fizik’))

Fizik dersinin bir ders kodu olacağı için bu satırdaki IN yerine = kullanılabilirdi.

# İç İçe SELECT ifadesi

- Önceki örneği çok tablolu sorgulama türünde de yapabilirdik:
  - SELECT OGR_NO, AD, SOYAD
    FROM OGRENCILER, NOTLAR, DERSLER
    WHERE OGR_NO = OGRENCI AND
    DERS_NO = DERS AND DERS_ADI = ‘Fizik’
- Fakat bu sorgu tabloların kartezyen çarpımına neden olacağı için muhtemelen daha yavaş çalışacaktır.

# Örnek Veritabanı



# JOIN vs WHERE

- Tablolar üzerinden bilgi çekmeyi 2 farklı yöntem ile yapabilirsiniz:
  - WHERE ile sorgu koşulu belirleyerek Tablo Birleştirme
  - JOIN komutlarını kullanarak Tablo Birleştirme

# WHERE üzerinden

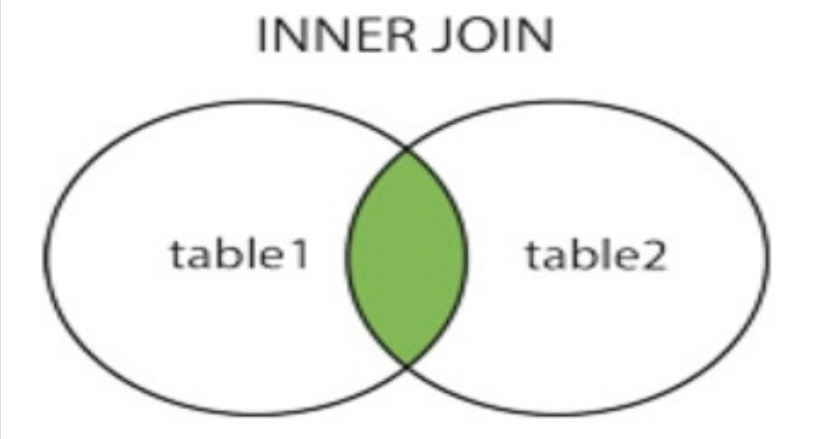
- Filmlerin türlerini gösterme
  - SELECT F.Ad AS Film_Adı, T.Ad AS Tur_Adı
    FROM Filmler F, Turler T
    WHERE F.Tur_ID = T.Tur_ID
- Eğer birleştirme koşulu belirtilmeseydi, VTYS iki farklı tablodaki her değer için kartezyen çarpım yapacaktı ve 70 farklı sonuç elde edecektik. Dolayısıyla istediğimiz sonuca ulaşamayacaktık.
- Tablo birleştirme 2 den fazla sayıda tablo için de uygulanabilir. Böyle bir durumda sorgu koşulu giderek büyüyecektir.
- Fakat bu sorgu tabloların kartezyen çarpımına neden olacağı için muhtemelen daha yavaş çalışacaktır.

# JOIN

- Birbirleri ile ilişkisi olan bu tablolar joinlerle birleştirmeler yapılarak gerekli alanlar alınabilir.
  - Inner join
  - Left join
  - Right join
  - Full join
- VTYS'ye göre ufak farklılıklar gösterebilir.

# INNER JOIN

- 2 tablo için de boş verileri, yani bir tabloda Tur_ID'si olan fakat diğerinde olmayan verileri temizler ve göstermez.
  - SELECT column_name(s)
    FROM table1
    INNER JOIN table2
    ON table1.column_name=table2.column_name;

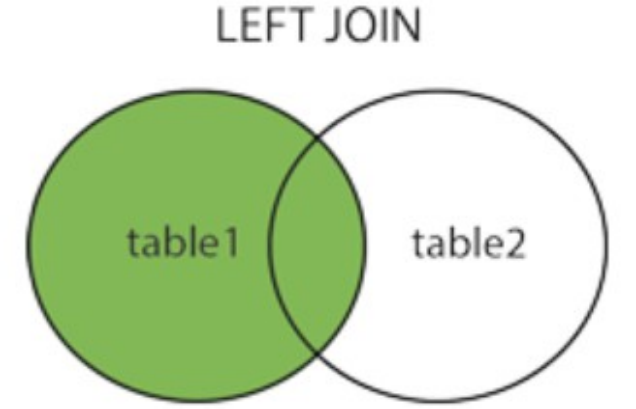


# INNER JOIN

- SELECT F.Ad AS Film_Adı, T.Ad AS Tur_Adı
  FROM Filmler F
  INNER JOIN Turler T
  ON F.Tur_ID = T.Tur_ID
  ORDER BY T.Ad

# LEFT JOIN

- İlk (Filmler) tablo için boş olan verileri de göstermenizi sağlar. Yani Filmler tablosunda Tur_ID değeri boş olarak gösterilecektir.
  - SELECT column_name(s)
    FROM table1
    LEFT JOIN table2
    ON table1.column_name=table2.column_name;



# LEFT JOIN

- Aşağıdaki sorgu, türü olmayan filmler dahil tüm filmlerin adlarını, türlerini ve IMDB Puanlarını, IMDB Puanlarına göre sıralayarak listeleyecektir.
  - SELECT F.Ad, T.Ad, IMDB_Puan

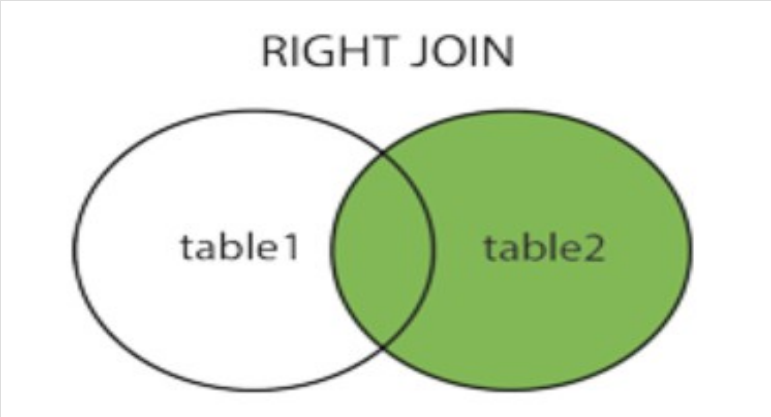
    FROM Filmler F

    LEFT JOIN Turler T ON F.Tur_ID = T.Tur_ID

    ORDER BY IMDB_Puan

# RIGHT JOIN

- İkinci (Türler) tablo için boş olan verileri de göstermenizi sağlar.
  - SELECT column_name(s)
    FROM table1
    RIGHT JOIN table2
    ON table1.column_name=table2.column_name;

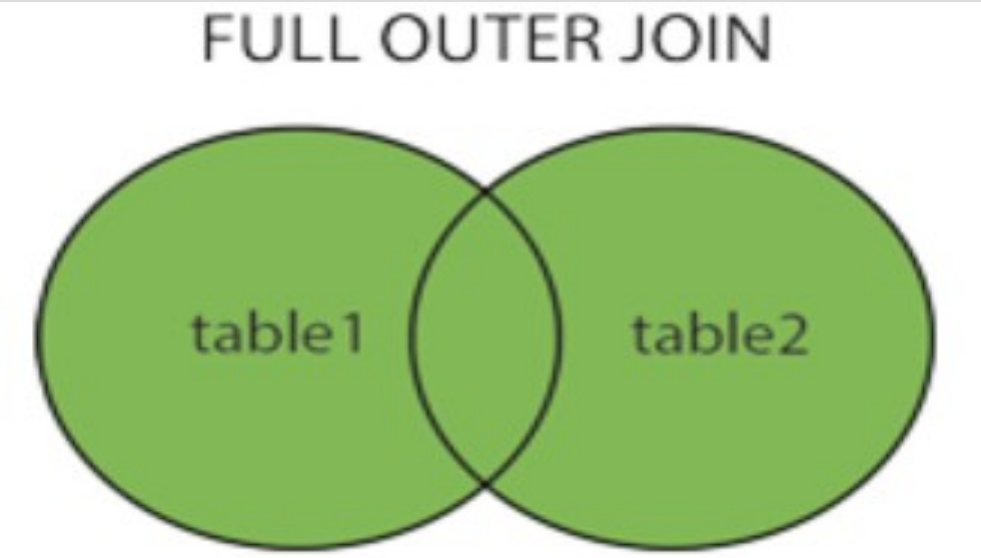


# RIGHT JOIN

- SELECT F.Ad, T.Ad, IMDB_Puan
  FROM Filmler F
  RIGHT JOIN Turler T ON F.Tur_ID = T.Tur_ID
  ORDER BY IMDB_Puan

# FULL JOIN

- İki tablo için de boş olan değerler gösterilir.
  - SELECT column_name(s)
    FROM table1
    FULL OUTER JOIN table2
    ON table1.column_name=table2.column_name;



# FULL JOIN

- SELECT F.Ad, T.Ad, IMDB_Puan
  FROM Filmler F
  FULL JOIN Turler T ON F.Tur_ID = T.Tur_ID

# UNION ve UNION ALL işleçleri

- Union, verilen koşul ifadesine uygun olarak, çift kayıtların göz ardı edilerek her iki kümedeki tüm kayıtları seçer.
- Union All, çift kayıtları da dahil eden birleştirme işlemidir.
- Örnek: PERSONEL tablosunda, adı “A” ile başlayanları veya görevi “M” ile başlayanları topluca listelemek için:

```
SELECT Adı,Görevi FROM PERSONEL
WHERE Adı LIKE “A*”
UNION
SELECT Adı, Görevi FROM PERSONEL
WHERE Görevi LIKE “M*”;
```

# DML

- DML (Data Manipulation Language): Tablolara veri girme, var olan veriyi güncelleme ve veri silme işlemleri için kullanılan komutları içerir.
  - INSERT
  - UPDATE
  - DELETE

# INSERT

- Bir tabloya kayıt eklerken kullanılan DML komutudur.
- Aşağıdaki DML ifadesi HOCALAR tablosuna kayıt ekler:

  INSERT INTO HOCALAR VALUES (15, ‘Ahmet’, ‘Çalışkan’, ‘Prof. Dr.’)
- Eğer eklenecek kaydın sadece belirli alanlarına veri girilecekse (Ör. HOCA_NO alanı ‘otomatik sayı’ veri türünde ise o alana veri giremeyiz), tablo adından sonra bu alanlar belirtilmelidir:

  INSERT INTO HOCALAR (AD, SOYAD, UNVAN) VALUES (‘Ahmet’, ‘Çalışkan’, ‘Prof. Dr.’)

# UPDATE

- Bir tablodaki kayıtların güncellenmesi amacıyla kullanılan DML komutudur. Hangi kayıt yada kayıtların güncelleneceği WHERE sözcüğü ile verilen kriter ile, kayıtlardaki güncellenecek alanlar ise SET sözcüğü sonrasında yeni değerlerinin atanması ile belirtilir.
- Aşağıdaki DML ifadesi PERSONEL tablosundaki Pazarlama bölümünde çalışanların maaşlarını %10 oranında arttırır.

```
UPDATE PERSONEL SET MAAS = MAAS * 1.1
WHERE BOLUM = (SELECT BOLUM_NO FROM
BOLUMLER WHERE BOLUM_ADI = 'Pazarlama')
```

# DELETE

- Bir tablodaki bir yada daha çok kaydı silmek amacıyla kullanılan DML komutudur. Hangi kayıt yada kayıtların silineceği WHERE sözcüğünden sonra verilen kriter ile belirlenir.
- Aşağıdaki DML ifadesi NOTLAR tablosundan ‘2007-2008’ öğretim yılına ait tüm kayıtları siler:

  DELETE FROM NOTLAR WHERE DERS_YILI = ‘2007-2008’
- Eğer WHERE sözcüğü hiç kullanılmaz ise tablodaki tüm kayıtlar silinir:

  DELETE FROM NOTLAR
- Silme İşleminin “Primary Key” üzerinden yapılması tavsiye edilir.

# Sorgu Sonucundan Yeni Tablo Yaratma

- Bir sorgu sonucunun yeni bir tablo olarak saklanması isteniyorsa FROM öncesinde INTO TABLO_ADI kullanılır.
- Aşağıdaki SQL cümlesi PERSONEL tablosundaki tüm verileri yeni yaratacağı PERS_YEDEK tablosuna kopyalar:

```
SELECT * INTO PERS_YEDEK FROM PERSONEL
```

- Eğer PERS_YEDEK tablosu önceden varsa, yukarıdaki komut önce tabloyu siler, sonra tekrar yaratarak verileri kopyalar.

# Sorgu sonucunu INSERT ile kullanma

- Bir tabloya veri eklerken INSERT ifadesinde VALUES yazılmayıp bir sorgu da yazılabilir.
- Aşağıdaki ifade PERSONEL tablosundaki tüm verileri önceden yaratılmış olan PERS_YEDEK tablosuna ekler:

```
INSERT INTO PERS_YEDEK SELECT * FROM PERSONEL
```

- Bu komut çalıştırılmadan önce PERS_YEDEK tablosu boş değilse anahtar alan olan PERSONEL_NO alanında veri tekrarına neden olabilir (dolayısı ile hata verebilir).
- Aşağıdaki ifade sadece 2010 yılından sonra işe başlayanları ekler:

```
INSERT INTO PERS_YEDEK SELECT * FROM PERSONEL
WHERE GIRIS_TARİHI > #1/1/2010#
```

# DDL

- DDL (Data Definition Language): Tablo, Kullanıcı gibi nesneleri yaratmak için kullanılan komutları içerir.
  - Create
  - Alter
  - Drop
- VTYS göre ufak farklılıklar gösterebilir. Bu durumda google üzerinden anahtar kelimeye uygun örnekler araştırılmalıdır.

# Create

- Veritabanı üzerinde bir tablo yaratmak için CREATE deyimi kullanılır.

```
CREATE TABLE Persons
(
PersonID int,
LastName varchar(255),
FirstName varchar(255),
Address varchar(255),
City varchar(255)
);
```

# Alter

- Daha önce yaratılmış nesnenin değiştirilmesini sağlar. Örneği bir tablonun tasarımını değiştirmek gibi.
  - Yeni özellik ekleme

    ALTER TABLE table_name

    ADD column_name datatype
  - Bir özelliği düşürme

    ALTER TABLE table_name

    DROP COLUMN column_name
  - Bir özelliğin veri tipini veya iç özelliklerini değiştirme

    ALTER TABLE table_name

    ALTER COLUMN column_name datatype

# Drop

- Bir tablonun veya özelliğin silinmesini sağlar.
  - Tablo Silme

    DROP table_name
  - Tablo içindeki bir özelliğin silinmesi

    DROP INDEX index_name ON table_name
- Eğer bir tablo içindeki veriler tamamen silinmek isteniyorsa
  - TRUNCATE TABLE table_name
  - Truncate kavramı ile bir şarta bakmaksızın tüm tablomuzu boşaltırız. Delete komutunda Where şartı ile belli şartlara bağlayabiliriz.