create database stok; // veritabanı oluşturma

CREATE TABLE Musteriler // tablo oluşturma

(

MusteriNo integer NOT NULL,

Adi varchar(20 not NULL,

Soyadi varchar(20) NULL,

DogumTarih date NULL,

sehir varchar(20) NULL,

Telefon varchar(11) Null,

)

INSERT INTO Müşteriler

( MusteriNo,Adi,Soyadi, DogumTarih,Şehir, Telefon,Adres)

VALUES (22,’AHMET’, ‘KARA’,’01.11.2000’,’ANKARA’,1234567,’Yukarı mahalle’)

ALTER TABLE Musteriler // tablo güncelleme

ADD adres varchar(30) NOT NULL

UPDATE Musteriler set sehir=’konya’ WHERE sehir=’Ankara’

// şehiri ankara olanları konya olarak düzenliyor

DROP TABLE Musteri // tablo silme

DELETE from Musteriler where adres='yukarý mahelle'

// tablodan kayıt silme

select MusteriNo as 'Musteri Numarası' from Musteriler

// sütun ismini değiştirmek için kullanılır.

SELECT adi,soyadi FROM Musteriler;

// seçim için kullanılır, Select \* ile tüm alanlar listelenir.

SELECT distinct adi,Soyadi FROM Musteriler;

// distinct aynı olan kayıtlardan birini gösterir. Adı ve soyadı aynı olan kişelerin tek kayıtlarını gösterir.

select \* from Musteriler order by MusteriNo DESC ,

telefon desc

// DESC azalan sıralama – ASC artan sıralama yapılır. Araya , konarak birden fazla alan için sıralama belirtilebilir.

**select \* from Musteriler**

where MusteriNo between 1 and 3 // 1 ile 3 arası

where MusteriNo <3 and MusteriNo >1 // 1 ile 3 arası

where MusteriNo = 3 // 3 e eşit

where MusteriNo < 3 // 3 ten kücük

where MusteriNo <> 22 //22 eşit olmayanlar

where MusteriNo = 22 and Soyadi= 'kara'

where MusteriNo = 22 or Soyadi= 'kara'

where MusteriNo in(2,22,1) // musteri no 2,22,1 olan ları listeler.

LİKE KOMUTU KULLANIMI

where adres like 'karatay sanayi' // adres= karatay sanayi demek

where adres like '% sanayi' // sonu sanayi ile biten adresleri listeler

where adres like '%san%' // icinde san kelimesi olan adresleri listeler.

KOMUT KULLANIM SIRALAMASI

SELECT [ALL | DISTINCT] *column1*[,*column2*]

FROM *table1*[,*table2*]

[WHERE "conditions"]

[GROUP BY "column-list"]

[HAVING "conditions]

[ORDER BY "column-list" [ASC | DESC] ]

**ARİTMETİK KOMUTLAR**

select Adi,Soyadi,borc, borc\*1.1 from Musteriler

// borc alanını yüzde on fazlasını gösterir.

select SUM (borc) from Musteriler where Adi ='ali'

// adı ali olan kişinin toplam borcu

select AVG (borc) from Musteriler

where Adi ='ali'

// Alinin borc ortalaması.

select \* from Musteriler

where borc=(select MAX (borc)

where borc=(select MIN (borc)

from musteriler where Adi <> 'ali')

// adı ali olmayan kişiler arasındaki borcu en yüksek olan kişiyi buluyor.

select COUNT (MusteriNo) from Musteriler where Adi='ali'

// adı ali olan kayıt sayısını verir.

select COUNT ( distinct Adi) from Musteriler

// adı farklı olan toplam kayıt sayısını verir.

select Adi from Musteriler group by sehir

// kişileri şehirlerine göre grup olarak gösteriyor.

**HAVING KULLANIMI**

HAVING clause, GROUP BY’dan sonra yer almalıdır.

Müsteriler tablosundaki kişileri adına göre gruplandırma yapıp, ortalama borcu 10 dan fazla olan kayıtları göseriyor.

SELECT Adi, avg(borc)

FROM Musteriler

GROUP BY adi

having AVG(borc)>10

|  |  |
| --- | --- |
| Karşılaştırma operatörleri | (=, >, <, >=, <>, !=, !<, !> |
| Aralık belirtme | BETWEEN ve NOT BETWEEN |
| Liste | IN ve NOT IN |
| String karşılaştırma | LIKE ve NOT LIKE |
| Bilinmeyen değerler | IS NULL ve IS NOT NULL |
| Koşulların birleştirilmesi | AND, OR |
| Olumsuzlaştırma | NOT |

**COMPUTE KULLANIMI**

SELECT adi,telefon FROM Musteriler ORDER BY Telefon

COMPUTE max (adet)// son satırda maximum adeti gösterir

COMPUTE min (adet)// son satırda minimum adeti gösterir

COMPUTE avg (adet)// son satırda ortalama adeti gösterir

COMPUTE sum (adet)// son satırda tüm adetlerin toplamını gösterir

**TABLOLARDA İLİŞKİLENDİRME**

**Örenk 1: üç tane tablodan yararlanarak tüm doktorların istenilen kayıtlarını gösterir.**

select Unvanlar.unvan,

doktorlar.Dadi, uzmanliklar.uzmanlik

from Unvanlar,doktorlar,uzmanlýklar, doktorUzmanlýklarý

where doktorlar.unvanno=Unvanlar.unvanno and

uzmanliklar.uzmanlikno=doktorUzmanliklarý.uzmanlýkno and doktorlar.doktorno=doktorUzmanlýklarý.doktorno

**unvan Dadı uzmanlık**

**Docent Mehmet Dahiliye**

**Y.docent Yasin Hariciye**

**Y.docent Oğuzhan Dahiliye**

**Y.docent Selman Hariciye**

**Docent Mehmet Çocuk**

**Örnek 2: uzmanlık kodu 1 olan doktorları listeleme.**

select doktorlar.Dadý, uzmanlýklar.uzmanlýk

from doktorlar,uzmanlýklar, doktorUzmanlýklarý

where uzmanlýklar.uzmanlýkno ='1' and

doktorUzmanlýklarý.uzmanlýkno=doktorlar.doktorno

**Dadı uzmanlık**

Mehmet Dahiliye

Yasin Dahiliye

Mehmet Dahiliye

Yasin Dahiliye

Oğuzhan Dahiliye

* Tablo isimlerini kısaltarak kullanabiliriz.

select d.Dadý, u.uzmanlýk

from doktorlar d,uzmanlýklar u, doktorUzmanlýklarý du

where u.uzmanlýkno ='1' and

du.uzmanlýkno=d.doktorno

**örnek 3: Adanalı doktarları listeler**

select d.Dadý, i.il

from doktorlar d, iller i

where d.ilno=i.plaka and i.il='Adana'

**örnek 4: Bölümü dahiliye olan doktarları listele**

select d.Dadý, u.uzmanlýk

from doktorlar d

inner join doktorUzmanlýklarý du on d.doktorno =du.doktorno

inner join uzmanlýklar u on u.uzmanlýkno=du.uzmanlýkno

where u.uzmanlýk='Dahiliye'

**ALT SORGULAR (İç İçe Sorgular)**

Alt sorguların en basit kullanımı yukarıdaki örnekteki gibidir. Bu örneğimizi biraz daha geliştirmek için biraz daha konuyu açalım.Basit bir alt sorguda bulunan "iç sorgu" yanlızca bir kez değerlendirilmektedir.Elbette değerlemeyi dış sorgudan aldığı bir değerle karşılaştırmakta münkündür. "İlişkili Alt Sorgular" olarak adlandırdığımız bu sorguları ilerleyen Ms Sql yazılarımda bulabilrsiniz. Basit Alt sorgulara dönersek

**Basit sorgular aşağıdaki operatörlerle kullanılabilmektedir.**

• Karşılaştırma Operatörleri

• IN Operatörü

• ANY ve ALL Operatörleri

• EXISTIS Fonksiyonu

ÖRNEK 1: ÜNVANI DOCENT OLAN DOKTARLARIN İSİMLERİNİ LİSTELEME.

select doktorlar.Dadi, doktorlar.Dsoyadý

from doktorlar

where doktorlar.unvanno =(select unvanno from Unvanlar

where unvan='Docent')

ÖRNEK 2 IN ve Karşılaştırma Operatörü):geriye birden fazla değer döndürülmek istenildiğinde kullanılır.Aşağıdaki örnekte hem docent hem doktar olanlar listeleniyor.

select doktorlar.Dadý, doktorlar.Dsoyadý

from doktorlar

where doktorlar.unvanno in (select unvanno from Unvanlar

where unvan='Docent' or unvan='doktor')

örnek 3 any kullanılım : Maaşı ortalamanın üzerinde olan doktarları listeleme.

select doktorlar.Dadý, doktorlar.Dsoyadý

from doktorlar

where doktorlar.doktorno = any

(

select doktorid from doktormaas where dmaas >

(select AVG(doktormaas.dmaas) from doktormaas)

)

***Dml KOMUTLARI***

* **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.’)

* **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

* **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’)

* **Sorgu sonucunu tablo yapma**
* 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\_TARIHI > #1/1/2010#