SQL DERS NOTLARI

I. SQL DEYİMLERİ

SQL deyimleri veritabanları üzerinde çeşitli işlemleri yerine getirirler. Veritabanından sorgulama yapmak için SELECT, ekleme yapmak için INSERT güncelleme yapmak için UPDATE, silme yapmak için DELETE, yeni tablo oluşturmak için CREATE TABLE gibi komutlara sahiptir. Bu komutlar işlevlerine göre şu şekilde kategorilendirilir:

- DDL (Data Definition Language): Veri tanımlama deyimleri.
- DML (Data Manuplation Language) : Veri düzenleme dili.
- DCL (Data Control Language): Veri kontrol dili.

A. DDL (DATA DEFİNİT

CREATE DEYİMİ İON LANGUAGE) DEYİMLERİ

DDL deyimleri veritabanı üzerinde nesne yaratmak için kullanılırlar. En yaygın kullanılan DDL deyimleri şunlardır: **CREATE ,ALTER,DROP**

CREATE deyimi tablo ve view gibi bir veritabanı nesnesi yaratmayı sağlar.

Tablo Yaratmak:

Veritabanı üzerinde bir tablo yaratmak için **CREATE** deyimi kullanılır.

Yapısı: **CREATE <tablo adı>**

Örnek: CREATE TABLE Musteri

```
(
  mus_id  char(4) NOT NULL
  mus_ad            varchar(40)    NULL,
  ili               varchar(20)    NULL,
  ulke              char(2) NULL,
  adres    varchar(30)    NULL
)
```

NOT: Char, varchar, integer, numeric gibi sözcükler tablo alanlarındaki temsil edilecek verinin türünü belirtir. **SQL'de SMALLINT, VARCHAR, DECIMAL(x,y), FLOAT(x;y), DATE, LOGICAL, TIME, TIMESTAMP, GRAPHIC(n)** gibi alan veri türleri vardır.

Örnek: CREATE TABLE personel

```
(
Sskno Integer,
Adi Varchar(20) not null,
Soyadi Varchar(20) not null,
Departman integer
)
```

ALTER DEYİMİ

Daha önce yaratılmış nesnenin değiştirilmesini sağlar. Örneği bir tablonun tasarımını değiştirmek gibi.

Örnek: ALTER TABLE Musteri
ADD tel varchar(20) NOT NULL

Yukarıdaki deyimde müşteri tablosunun alanlarına tel adlı bir alan daha eklenmiştir.

DROP DEYİMİ

Bir nesnesin silinmesini sağlar.

Örnek: **DROP TABLE MUSTERI**

Müşteri tablosunun verilerini ve tabloyu siler.

B. DML (DATA MANIPULATION LANGUAGE) DEYİMLERİ

Veritabanı içindeki verileri elde etmek ve değiştirmekle ilgili SQL deyimleridir.

1. **SELECT**
2. **INSERT**
3. **UPDATE**
4. **DELETE**

SELECT DEYİMİ

Veritabanındaki verilere erişmenin, diğer bir deyişle onları görmenin ya da onları elde etmenin en sık kullanılan yöntemidir. Genellikle bir ya da daha çok tablonun bütün alanları ya da belli alanları için SELECT deyimi yazılır.

Temel Yapısı: **SELECT [ALL] [DISTINCT] liste [INTO yeni tablo] FROM [tablo]**
[WHERE ifade]
[GROUP BY ifade]

[HAVING ifade]
[ORDER BY ifade]
[COMPUTE ifade]

Seçeneklerin Anlamları:

ALL sözcüğü bütün satırların sonuç listesinde görünmesini sağlar.

DISTINCT sözcüğü sadece tek olan (unique) kayıtların sonuç listesinde yer almasını sağlar.

liste parametresi veriden seçilecek kolonu (sütunu) belirtir.

INTO sözcüğü yeni bir tablo yaratmayı sağlar.

yeni tablo parametresi sorgu sonucu yaratılacak tabloyu belirtir.

FROM sözcüğü belli bir tablonun seçilmesini sağlar.

tablo parametresi ise sorgulanacak olan tablo ya da tabloları, görünümleri belirtir.

WHERE bir koşulu belirterek sadece o kuşula uyan kayıtların seçilmesini sağlar.

GROUP BY Kayıtların gruplanmasını sağlar. HAVING deyimiyle de ara toplamların alınmasını sağlar.

HAVING sözcüğü de kayıtlarda kısıtlama yapar ancak hesaplamayı etkilemez.

ORDER BY sözcüğü ise belirtilen kolona göre listelenen kayıtları sıralamayı sağlar. Sıralama artan (ASC) ya da azalan (DESC) olabilir

COMPUTE sözcüğü ise hesaplama yapar. Tipik olarak SUM, AVG, MIN, MAX, COUNT gibi fonksiyonları kullanarak hesaplama yapar.

Örnek: SELECT * FROM musteri

Yukarıdaki deyim ile musteri tablosundaki bütün bilgiler elde edilir. SELECT deyiminin ardından kullanılan * (asterisk) işareti bütün kayıtlar anlamına gelir. Bu deyimin aynısı (aynı sonucu vereni) şu şekilde de yapılabilir:

Örnek: SELECT kod, ad, soyad, grup, il, bakiye FROM musteri

Müşteri tablosu:

kodu	Ad	Soyad	grup	il	bakiye
1	Ahmet	Uzun	ithal	İZMİR	300000
2	Ayşe	Yılmaz	ithal	ANKARA	400000
3	Mehmet	Yılmaz	ihraç	ANKARA	100000
4	Hüseyin	Uzun	ihraç	İZMİR	600000
5	Nuri	Gezer	ithal	İZMİR	900000
6	Fatma	Örnek	ihraç	İSTANBUL	300000

Sorgunun sonucu:

1	Ahmet	Uzun	ithal	İZMİR	300000
2	Ayşe	Yılmaz	ithal	ANKARA	400000
3	Mehmet	Yılmaz	ihraç	ANKARA	100000
4	Hüseyin	Uzun	ihraç	İZMİR	600000
5	Nuri	Gezer	ithal	İZMİR	900000
6	Fatma	Örnek	ihraç	İSTANBUL	300000

SELECT deyimi ile sadece belli kolonlar (alanlar) da seçilebilir:

ARAMA : Tablo: Arama Kriterleri:

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**

ÖRNEK: SELECT * from musteri WHERE bakiye BETWEEN 100000 AND 3000000

Aynı anlamda: SELECT * from musteri WHERE bakiye <= 100000 AND>= 3000000

ÖRNEKLER: SELECT * FROM MUSTERI WHERE borcu BETWEEN 100 AND 2000 AND grup='özel'

Yukarıdaki sorgulamada grubu 'özel' olan ve ayrıca da borcu değerinin 100 ile 200 arasında olanı listelenir.

ARİTMETİK İŞLEMLER

Aritmetik işlemleri gerçekleştirmek için belli operatörler kullanılır:

OPERATÖRLER	AÇIKLAMALARI
+	Toplama ,
-	Çıkarma
/	Bölme ,

***** **Çarpma**

SELECT borcu, borcu*2 FROM MUSTERI

VERİLERİ SIRALAMAK

SELECT deyimi ile elde edilen veriler istenirse sıralanabilir. Sıralama belirtilen bir ya da daha fazla kolona göre yapılır. Bunun dışında sıralama ASC (ascending-artan) ya da **DESC** (descending-azalan) olarak belirtilebilir.

Kullanım Biçimi: SELECT kolon_listesi

ORDER BY kolon adı ASC ya da DESC

ORDER BY sözcüğü ise verilerin istenilen alan göre sıralı olarak listelenmesini sağlar.

SELECT * FROM musteri ORDER BY ad

Yukarıdaki deyim ile müşteri tablosundaki bütün kayıtlar ad alanına göre sıralı olarak listelenirler.

GRUPLANDIRMA İŞLEMLERİ

Gruplama seçenekleri tablo satırlarının belli alanlarına göre gruplandırılmasını sağlar.

GROUP BY sözcüğü ise yapılan bir gruplandırma işlemine göre listeleme sağlar.

Örnek: SELECT grup, sum(bakiye) FROM musteri

GROUP BY grup

Yukarıdaki örnekte müsteri tablosundaki bakiye alanı grup kodu bazında toplanır.

GROUP BY ile yapılan gruplandırma (alt toplamlar) işlemi içinde ayrıca HAVING sözcüğü kullanılarak bir koşul da verilebilir.

SELECT grup, sum(bakiye) FROM musteri

GROUP BY grup

HAVING il = 'İZMİR'

Yukarıdaki deyim ile müşteri tablosundaki bütün kayıtların bakiye bilgiler gruplarına göre ara toplam alınır, bir de il bilgisine göre filtrelenir.

NOT: Gruplama yapılmayan he sütun **GROUP BY** deyiminde kullanılması gerekir**:**

Örnek: SELECT PRODUCTID, CATEGORYID, SUM (UNITPRICE) FROM PRODUCTS

GROUP BY PRODUCTID, CATEGORYID

COMPUTE SÖZCÜĞÜ

COMPUTE sözcüğü toplama fonksiyonunun kullanarak query sonucunda bir toplam satırı üretir**. COMPUTE BY** sözcüğü ise ek toplam satırları yaratır.

Kullanım Biçimi: COMPUTE fonksiyon (kolon_adı)

Örnek: Sipariş tablosu:

tarih	Mkodu	ürün	adet	fiyatı
1/1/1999	1	X-TV	10	100000
2/1/1999	1	X-TR	10	100000
2/1/1999	2	X-TV	15	100000
2/1/1999	3	X-KT	10	200000
3/1/1999	3	X-KT	20	150000
2/2/1999	1	X-TV	22	400000

```
SELECT ürün, adet
 FROM siparis
 ORDER BY ürün
 COMPUTE SUM(adet)
```

Yukarıdaki örnekte adet kolonunun toplamı alınır.

Sonucu:

```
ürün        adet
X-KT        10
X-KT        20
X-TR        10
X-TV        10
X-TV        15
X-TV        22
              Sum
 ==================
              87
```

INSERT DEYİMİ

Tabloya veri girmek için kullanılır.
INSERT INTO <tablo adı>
(sütunlar listesi) **VALUES** (değerler listesi)
ÖRNEK: INSERT INTO CARIANA
(kodu, adi, grubu, adresi)
VALUES ('600', 'FARUK', 'A', '76 sokak no 5')
Örnek: INSER INTO deyimi ile bir tabloyu diğer bir tablodan doldurmak:
use ornek
INSERT INTO cariyedekf
(kodu, adi, grubu, adresi)
(SELECT kodu, adi,
case grubu

when 'A' THEN 'ITHAL'
when 'B' THEN 'YERLİ'
ELSE 'DİĞER'
end,
adresi from cariana)
UPDATE DEYİMİ:
Tablodaki verileri güncellemek için kullanılır. Genellikle güncelleştirilecek satırı belirtmek **için WHERE** sözcüğüyle kullanılır.
Mevcut bir tablodaki satırları değiştirmek için **UPDATE** deyimi kullanılır. **UPDATE** deyimi sadece bir tablo üzerinde kullanılmalıdır. **UPDATE** deyimi ile **SET** ve **WHERE** sözcüğü kullanılır.
SET sözcüğü değiştirilecek kolonları ve değerleri belirtir. **WHERE** sözcüğü ise değiştirilecek satırı belirtir.
Kullanım biçimi:
UPDATE tablo
SET kolon = ifade
WHERE arama_koşulu
Örnek: Aşağıdaki örnekte fiyat değerini %10 artırır.
UPDATE siparis
SET fiyatı= fiyatı * 1.1
Örneğin bir kaydı düzeltmek istersek ;
UPDATE Musteri
SET Ad = 'Nuri Yılmaz'
WHERE kod='1';
Örneğin tüm musterilerin bakiyesini %10 artırmak istediğimizde;
UPDATE Musteri
SET bakiye=bakiye*1.1;
DELETE DEYİMİ
Bir tablodaki verileri silmek için **DELETE** komutu kullanılır. Örneğin Öğrenci tablosundaki tüm verileri silmek için;
DELETE * from musteri;
Tabloda, bakiyesi 1000'den küçük olan müşterilerin satırlarını silmek için:
DELETE * FROM musteri WHERE bakiye <=1000
Kullanım biçimi:
DELETE tablo
WHERE arama_koşulu
Örnek: Tablodan satır silmek
Aşağıdaki örnekte müşteri tablosundan 'B' grubuna sahip olan müşteriler silinir.
DELETE musteri
WHERE grubu = 'B'

C. DCL (DATA CONTROL LANGUAGE) DEYİMLERİ

Veritabanındaki kullanıcı haklarını düzenlemek için kullanılan deyimlerdir. Örneğin **GRANT, DENY, REVOKE** gibi.
Örnek: USE Northwind
GRANT SELECT ON Stok TO PUBLIC

GRANT DEYİMİ

Aşağıdaki örnek Ayşe adlı kullanıcı veritabanı ve tablo oluşturma izni verilir**:**

GRANT CREATE DATABASE, CREATE TABLE TO AYŞE

GÖZDEN GEÇİRME

1. SQL deyimleri hangi gruplara ayrılır.
2. SELECT deyiminin kullanım şekillerini açıklayınız?
3. INSERT deyiminin amacı nedir?
4. UPDATE deyiminin amaçları nelerdir?

I. SQL FONKSİYONLARI

SQL Server'da Fonksiyonlar, hesaplamalarda ve özellikle sistem hakkında bilgi almada yaygın olarak kullanılan araçlardır. Transact-SQL (T-SQL) programlama dilinde değişik kategorilerle adlandırılan fonksiyonlar vardır. Bunlardan en yaygını veriler üzerinde işlem yapan toplama ya da gruplama olarak adlandırabileceğimiz aggregate fonksiyonlarıdır**.**

A. GRUPLAMA FONKSİYONLARI

Gruplama (aggregate) fonksiyonları bir dizi değer üzerinde hesaplama yaparlar ve bir sonuç değer döndürürler. Toplama ya da gruplama fonksiyonları olarak adlandırabileceğimiz bu fonksiyonlar genellikle **GROUP BY** deyimi ile kullanılırlar.

Gruplama fonksiyonları aşağıdaki ifadeler içinde kullanılabilirler.

- **SELECT** deyiminin listesinden (bir subquery olarak)
- Bir **COMPUTE** ya da **COMPUTE BY** sözcüğü ile.
- Bir **HAVING** sözcüğü ile.

Transact-SQL programlama dilinde şu aggregate fonksiyonları kullanılır:

- **AVG**
- **COUNT**
- **GROUPING**
- **MAX**
- **MIN**
- **SUM**
- **STDEV**
- **STDEVP**
- **VAR**
- **VARP**

AVG (T-SQL) Fonksiyonu

Bir grup içindeki değerlerin ortalamasını döndürür. Null değerler dikkate alınmaz.

Kullanım Biçimi: **AVG([ALL | DISTINCT] ifade)**

Argümanları: **ALL**

Ortalama fonksiyonunu bütün değerlere uygular.

DISTINCT

İşlemin her tek değer için uygulanacağını belirtir. Diğer bir deyişle tekrar eden değerlerin yerine birisi kullanılır.

İfade : Sayısal bir değeri olan ifade ya da kolon adı.

Örnek: Aşağıdaki örnekte satış miktarları toplanmakta ve ortalaması alınarak iki ayrı sonuç değeri verilmektedir:

```
USE Northwind
GO
SELECT AVG(quantity), SUM(quantity)
FROM [Order Details]
GO
```

COUNT (T-SQL) Fonksiyonu

Bir grup içindeki eleman sayısını verir.

Kullanım Biçimi:

COUNT({[ALL | DISTINCT] ifade] | *})

Argümanları:

ALL

Fonksiyonunu bütün değerlere uygular.

DISTINCT

İşlemin her tek değer için uygulanacağını belirtir. Diğer bir deyişle tekrar eden değerlerin yerine birisi kullanılır.

İfade :Bir ifade ya da kolon adı.

*****

Bir tablodaki bütün satırların sayısını döndürmek için kullanılır. COUNT(*) herhangi bir paratmetre ile kullanılmaz ve DISTINCT ile kullanılmaz.

COUNT(*) ile null ve tekrar eden değerler dahil bütün elemanlar sayılır.

Örnek: Aşağıdaki örnekte ürünler tablosundaki ürünler sayılır. DISTINCT ile tekrara izin verilmez.

```
USE Northwind
GO
SELECT COUNT(DISTINCT productname)
FROM products
GO
```

MAX (T-SQL) Fonksiyonu:

İfade içindeki maksimum değeri döndürür.

Kullanım Biçimi:

MAX([ALL | DISTINCT] ifade)

Argümanları: **ALL**

Fonskiyonunu bütün değerlere uygular.

MIN (T-SQL) FOnksiyonu

İfade içindeki minimum değeri döndürür.

Kullanım Biçimi:

MIN([ALL | DISTINCT] ifade)

Argümanları:

ALL

Fonskiyonunu bütün değerlere uygular.

Örnek:

Aşağıdaki örnekte en küçük satış adedi elde edilmektedir:

```
USE Northwind
GO
SELECT MIN(quantity)
FROM [Order Details]
GO
```

SUM Fonksiyonu

Değerlerin toplamlarını verir. SUM fonksiyonu sadece sayısal alanlarda kullanılır.

Kullanım Biçimi:

SUM ([ALL | DISTINCT] ifade)

Argümanları:

ALL

Toplama (aggregate) fonksiyonunu bütün değerlere uygular. ALL seçeneği varsayım olarak kullanılır.

DISTINCT

SUM fonksiyonunu tek değerlerin (unique) toplamını vermesini sağlar.

İfade

Bir sabit, bir kolon, fonksiyon ya da bir aritmetik işlem.

Örnek:

```
USE pubs
GO
-- Satır toplamları
SELECT type, price, advance
FROM titles
ORDER BY type
COMPUTE SUM(price), SUM(advance) BY type
```

B. TARİH VE ZAMAN FONKSİYONLARI (T-SQL)

Bu fonksiyonlar tarih (date) ve zaman (time) üzerinde işlemler yapmayı sağlar. Tarih ve zaman fonksiyonları şunlardır:

- **DATEADD**
- **DATEDIFF**
- **DATENAME**
- **DATEPART**
- **DAY**
-
- **GETDATE**
- **MONTH**
- **YEAR**
- **CONVERT**

DATEADD

- Belli bir tarihin üzerine değer eklenerek yeni bir tarih değeri üretir.

Kullanım biçimi: **DATEADD (tarihkısmı, sayı, tarih)**

Argümanları: **Tarih kısmı**

Tarih Parçası	Kısaltması
Year	yy, yyyy
Quarter	qq, q
Month	mm, m
Dayofyear	dy, y
Day	dd, d
Week	wk, ww
Hour	hh
Minute	mi, n
Second	ss, s
Millisecond	ms

Sayı: Tarih kısmını artırmak için kullanılan değer.

Tarih: Tarih değerini döndüren tarih bilgidi.

Örnekler: Yayın tarihinin 30 gün sonrası:

```
USE pubs
GO
SELECT DATEADD(day, 30, pubdate)
FROM titles
GO
```

- **Örnek:** Son on gün içinde yapılan siparişler:

```
SELECT ord_num, ord_date
FROM sales
WHERE
(ord_date >=
  DATEADD(day, -10, GETDATE()))
```

DATEDIFF

İki tarih arasındaki gün sayısını verir.

Kullanımı: DATEDIFF (Tarih parçası, başlangıç tarihi, bitiş tarihi)

Argümanları: Tarih parçası

Tarih Parçası	Kısaltması
Year	yy, yyyy
Quarter	qq, q
Month	mm, m
Dayofyear	dy, y
Day	dd, d
Week	wk, ww
Hour	hh
Minute	mi, n
Second	ss, s
millisecond	Ms

Örnek: Şu anki tarih ile yayın tarihi arasındaki fark:

```
USE pubs
GO
SELECT DATEDIFF(day, pubdate, getdate())
FROM titles
GO
```

DATEPART (T-SQL)

_Belirtilen tarihin istenen parçasına karşılık olarak bir tamsayı döndürür.

Kullanım Biçimi: DATEPART(tarihparçası, tarih)

Argümanları: Tarih parçası: Tarih bilgisinin bir kısmını ifade eden bilgi.

Tarih bölümü	Kısaltma
year	yy, yyyy
quarter	qq, q
month	mm, m
dayofyear	dy, y
day	dd, d
week	wk, ww
weekday	Dw
hour	Hh
minute	mi, n
second	ss, s
millisecond	Ms

Örnek: Aşağıdaki örnekte şu anki tarihin karşılık geldiği ay adı ve ay numarası bulunmaktadır:

SELECT GETDATE() Tarihi verir:

SELECT DATEPART(month, GETDATE()) Ayı verir:

DAY (T-SQL)

Bir tarih bilgisinin gün kısmını verir.

Kullanım Biçimi:

DAY(tarih)

Argümanları:

Tarih datetime ya da smalldatetime tarih bilgisi.

Aşağıdaki örnekte verilen tarihin gün kısmı verilmektedir:

```
SELECT DAY('26/03/1999') AS 'Gün'
GO
```

Sonuç:

```
Gün
------------
26
```

MONTH (T-SQL)

_Bir tarih bilgisinin ay kısmını verir.

Kullanım Biçimi: **MONTH(tarih)**

Argümanları: Tarih datetime ya da smalldatetime tarih bilgisi.

Aşağıdaki örnekte verilen tarihin ay kısmı verilmektedir:

```
SELECT MONTH('26/03/1999') AS 'Ay'
GO
```

Sonuç:

```
Ay
----------
03
```

YEAR (T-SQL)

_Bir tarih bilgisinin yıl kısmını verir.

Kullanım Biçimi: YEAR(tarih)

Argümanları: tarih

D. KARAKER FONKSİYONLARI

String (karakter) alanları işlemek için yaygın kullandığımız fonksiyonlar bu alana girer. Transact-SQL dilinde kullanılan bazı karakter fonksiyonları şunlardır:

1. **ASCII**
2. **CHAR**
3. **CHARINDEX**
4. **DIFFERENCE**
5. **LEFT**
6. **LEN**
7. **LOWER**
8. **LTRIM**
9. **NCHAR**
10. **REPLICATE**
11. **REVERSE**
12. **SUBSTRING**
13. **QUOTENAME**
14. **STUFF**
15. **REPLACE**
16. **STR**
17. **SOUNDEX**
18. **PATINDEX**
19. **SPACE**
20. **RIGHT**
21. **RTRIM**
22. **UPPER**
23. **UNICODE**

ASCII : Bir karakter ifadenin en soldaki değerinin ASCII kodunu döndürür.

Kullanımı: ASCII (karakter ifade)

Karekter ifade char ya da varchar türündedir.

Döndürdüğü tür: Int

Örnek: Bizim Ev cümlesinin bütün karakterlerinin ASCII değerini döndürür:

```
SET TEXTSIZE 0
SET NOCOUNT ON
-- değişken oluştur.
DECLARE @konum int, @karakter char(15)
-- değişkenlere ilk değer ver.
SET @konum= 1
SET @karakter= 'Bizim Ev'
WHILE @konum<= DATALENGTH(@karakter)
```

```
  BEGIN
  SELECT ASCII(SUBSTRING(@karakter, @konum, 1)),
    CHAR(ASCII(SUBSTRING(@karakter, @konum, 1)))
   SET @konum= @konum + 1
  END
SET NOCOUNT OFF
GO
```

CHARINDEX

Bir karakter dize içinde belirtilen bir ifadenin (karakterin) başlangıç konumunu döndürür.
Kullanımı: **CHARINDEX (ifade1, ifade2[, başlangıç konumu])**
İfade1: aranacak karakterleri belirtir.
İfade2: İfade1'deki karakterlerin aranacağı karakterleri belirtir.
Başlangıç konumu ise aramanın başlanacağı konumu belirtir.
Döndürdüğü tür: Int
İfadelerden birisi NULL ise CHARINDEX fonksiyonu NULL değerini döndürür. İfade1, İfade2 içinde bulunamazda 0 değeri döner.
Örnek: ADI alanında BOYASI sözcüğünün başladığı konumu döndürür.

```
SELECT CHARINDEX('BOYASI', adi)
FROM urun
```

```
-- arama için başlangıkç konumu belirtmek istersek
SELECT CHARINDEX('BOYASI', adi, 5)
FROM urun
Yalnızca adlar listesi:    select left (adi,charindex(' ', adi)) from cariana
```

LEFT

Bir karakter dizesinin sol taraftan belirtilen sayı kadar keser.
Kullanımı: LEFT (karakter dize, tarmsayı)
Örnek: Adların soldan beş karakteri:

```
USE ornek
GO
SELECT LEFT(adi, 5)
FROM cariana
ORDER BY kodu
GO
```

LEN

Dize verinin uzunluğunu döndürür.
Kullanımı: LEN (karakter dize)
Örnek: Adı alanının uzunluğu:

```
SELECT LEN(adi) AS 'Uzunluk'
FROM cariana
use ornek
select substring (adi,charindex(' ',adi)+1,(len(adi)-charindex(' ',adi))) from cariana
-- soyadını ayırmak
```

LTRIM

Önündeki boşlukları siler.
Kullanımı: LTRIM (karakter dize)

RIGHT Belirtilen dizenin sağ tarafından keser.
Kullanımı: RIGHT (karakter dizesi, tamsayı)
Arguments
Örnek: Adı alanını sağlan 10 karakteri:

SELECT RIGHT(adi, 10)
FROM cariana

RTRIM Karakter dizesinin arkasındaki boşlukları kaldırır.
Kullanımı: RTRIM (karakter dize)
Örnek: Bir alan güncelleme: update deneme
set alan1 = rtrim(alan1) + rtrim('a')

STUFF Belirtilen uzunluktaki karakterleri siler ve yerine belirtilen diğer karakterleri ekler.
Syntax : STUFF (karaker dize, başlangıç, uzunluk, karakter dize)
Örnek: use ornek
SELECT STUFF(tel, 7, 1, '8') from cariana
-- telefon numarasında 7 karakteri 9 ile değiştirmek

SUBSTRING Bir karakter dizesinin içinden belli karakterleri seçer.
Kullanımı: SUBSTRING (karakter dize, başlangıç, uzunluk)
Örnek: Adı alanının içinde üçüncü karakter başlayım 4 tane karakteri döndürmek:

SELECT SUBSTRING(adi, 3, 4)
FROM cariana

Adı soyadı alanından soyadını çekmek:

select substring (adi,charindex(' ',adi)+1,(len(adi)-charindex(' ',adi))) from cariana

GÖZDEN GEÇİRME
1. SQL Dili fonksiyonlarının sınıflarını açıklayınız. 2. Bir alanın soldan üç karakterini seçmek için hangi fonksiyonları kullanabiliriz.

I. SQL DEYİM BLOKLARI

SQL deyimlerini işletirden bir grup deyimi bir arada işletmek gerekebilir. Bu olanak deyim bloklarıyla yapılır. Diğer bir olanak da IF, CASE ve WHILE gibi hem blok olarak hem de blok olmadan işletilecek deyimleri belli koşullara bağlamaktır.

A. BEGIN...END

BEGIN ve END deyimleri bir grup SQL deyimini bir blok içinde toplamayı sağlar.
Örneğin IF deyimi ile yalnızca bir deyimin işletilip işletilmemesi sınanırken, BEGIN ve END bloğu ile bir grup deyimin çalıştırılıp çalıştırılmaması sağlanır.
Örnek: WHILE (SELECT AVG(adet) FROM titles) < 30
-- döngü başlat
BEGIN UPDATE siparis SET adet= adet* 2
END
BEGIN ve END deyimleri genellikle şu durumlarda kullanılır.
WHILE ile döngü yapıldığında.
Bir CASE fonksiyonunun blok deyimi içermesi durumunda.
IF ve ELSE deyiminde.

B. DENETİM DEYİMLERİ

SQL dilinde programlama dilleri kadar olmasa da program akışını kontrol etmek için deyimler ve yapılar vardır. Bunların başında IF-ELSE, CASE ve WHILE yapısı gelir.
IF...ELSE Bir deyimin işletilmesini belli bir koşula bağlar.
Kullanımı:

```
IF ifade
   { deyim }
[ ELSE
   { deyim} ]
```

Örnek: Adet ortalamasının 20'den küçük olması durumunda çalıştırılacak deyimler:

```
IF (SELECT AVG(adet) FROM siparis) < 20
BEGIN
  --işlemler
```

END
ELSE
BEGIN
--işlemler
END
CASE: Bir değere göre daha fazla alternatifi yerine getirmeyi sağlar.
Kullanım biçimi: CASE değer
WHEN değer THEN işlem
WHEN değer THEN işlem
ELSE işlem
END

Tablodan Aktarma: INSERT INTO yenitablo (alanlar..)
SELECT (alan1, alan2, alan3,
CASE alan4 WHEN 'A' THEN '1' WHEN 'B' THEN '2' ELSE '3' END,
alan5 FROM eskitablo
WHERE isdate (tarih) <> 0
Örnek: Tablolar arasında aktama:
İNSERT İNTO KAMİL1
(KODU, ADİ, GRUBU, ADRESİ)
SELECT KODU, ADİ,
(CASE GRUBU WHEN 'A' THEN 'EMEKLI' WHEN 'B' THEN 'TERHIS' END)
,ADRESİ FROM KAMİL2
WHILE
SQL deyimlerinin döngü içinde yinelenmesini sağlar. WHILE ile belirtilen döngü koşulu yerine getirildiği sürece deyimler yerine getirilir.
Kullanımı: WHILE ifade
{ deyim ya da blok}
[BREAK]
{ deyim ya da blok}
[CONTINUE]
Örnek: Satış adetleri 50 oluncaya kadar adet alanının artır.
WHILE (SELECT AVG(adet) FROM siparis) < 50
BEGIN
UPDATE siparis
SET adet= adet* 2
SELECT MAX(adet) FROM siparis
IF (SELECT MAX(adet) FROM siparis) > 50
BREAK
ELSE
CONTINUE
END
PRINT 'adet değeri büyük'''

BREAK ve CONTINUE kullanmadan:
WHILE (SELECT AVG(adet) FROM siparis) < 50
BEGIN
UPDATE siparis
SET adet= adet* 2
END
PRINT 'adet değeri büyük'''
BREAK ve CONTINUE kullanmadan:
GÖZDEN GEÇİRME
1. SQL deyimlerini neden bir blok haline getiririz?

DERS 4: VIEW KULLANIMI

Amaçlar:
VIEW oluşturmayı ve kullanımını açıklamak.

I. VIEW NEDİR?

Tablolardaki verilere erişmenin bir diğer yolu da view'ler geliştirmektir. View'ler tabloların belli kolonların listelendiği ayrıca hesaplama işlemlerinin yapıldığı bir veri erişim yöntemidir.

View kullanmanın çok sayıda üsütünlüğü (kolaylığı) vardır. Bunların başında database üzerindeki çok sayıdaki tablo üzerinde özel görünümler yaratması ve kullanıcılara tablolar (görünümün altında yatan) üzerinde izin vermeden tablolar üzerinde işlem yapmalarını sağlar. Örneğin kullanıcı, tablonun sadece belli kolonlarını içeren bir View üzerinde çalışabilir.

A. VİEW OLUŞTURMAK

CREATE deyimi ile yaratılır. İçinde genellikle SELECT gibi bir cümle bulunur.

Kullanım Biçimi:

```
CREATE VIEW view_adı
   AS
select_deyimi
```

Aşağıdaki örnekte bir tablonun belli alanları üzerine bir View yaratılmaktadır.

```
CREATE VIEW musteri_view AS
SELECT Kodu, Adı, Soyadı, Grubu
FROM musteri
```

View'ların yaratılmasında SELECT deyimi kullanılır. Ancak bazı kısıtlamaları vardır:

- **ORDER BY, COMPUTE, ya da COMPUTE BY sözcükleri kullanılmaz.**
- **INTO kullanılmaz.**

B. VİEW'LARI ÇALIŞTIRMAK

Bir view'dan veri almak için genellikle SELECT kullanılır.

Aşağıdaki deyim yukarıda hazırladığımız view içindeki verileri görüntüler:

```
SELECT * from musteri_view
```

GÖZDEN GEÇİRME
1. View'ların amacı nedir?

I. STORED PROCEDURE

SQL Server'daki Stored procedure'lar aynı diğer programlama dillerindeki procedure'lara benzer.
SQL deyimlerini içeren komut doayaları hazırlanır ve sunucu üzerinde saklanır.
Stored procedure aracılığıyla şu işlemler yapılablir:
Input parametrelerini kabul ederek ve birçok değerin geri dönmesini sağlar.
Database içindeki işlemleri yapmak için programlama deyimleri içerir.

Stored procedure'lar genellikle rutin hale gelmiş işleri kolayca yapmak için geliştirilirler. SQL deyimleriyle yazılan stored procedure'lar sadece ilk kez çalıştırıldıklarında derlenirler. Daha sonraki çalıştırma işlemlerinde derlenmezler ve böylece hızlı bir biçimde çalışma sağlanmış olur.
Örnek bir stored procedure tasarımı:

```
USE Northwind
GO
CREATE PROC pahali_kitaplar
AS
  SELECT *
  FROM products
  WHERE unitprice > 30
 GO
```

Stored procedure'ı Çalıştırma:

```
EXEC pahalı_kitaplar
```

A. CREATE PROCEDURE Deyimi

Bir SQL Server stored procedure'ı CREATE PROCEDURE deyimi ile oluşturulur. İstenirse daha sonra ALTER PROCEDURE deyim ile değiştirilir. Bir stored procedure yaratma deyimi tek bir batch olarak düzenlenir. Diğer bir deyişle diğer SQL deyimleriyle aynı batch içinde yer alamaz.
Kullanım Biçimi: :

```
 CREATE PROC[EDURE] procedure_adı[;sayı]
[
{@parameter data_tipi} [VARYING] [= varsayım] [OUTPUT]
]
[,...n]
[WITH
{
RECOMPILE
| ENCRYPTION
| RECOMPILE, ENCRYPTION
}
]
[FOR REPLICATION]
AS
sql_deyimleri [...n]
```

Örnekler: Aşağıdaki stored procedure ile sadece belirtilen yazarın bilgileri ve kitapları listelenir.

```
USE pubs
GO
CREATE PROCEDURE yazar
@lastname varchar(40),
@firstname varchar(20)
AS
SELECT au_lname, au_fname, title, pub_name
FROM authors a INNER JOIN titleauthor ta
ON a.au_id = ta.au_id INNER JOIN titles t
ON t.title_id = ta.title_id INNER JOIN publishers p
ON t.pub_id = p.pub_id
WHERE au_fname = @firstname
AND au_lname = @lastname
GO
```

Çalıştırılması:

```
EXECUTE yazar 'White, 'Jhonson'
```

Ayrıca aşağıdaki biçimde de çalıştırılabilir:

```
EXECUTE yazar @lastname = 'White', @firstname = 'Johnson'
```

B. BİR STORED PROCEDURE'I İŞLETMEK

Bir stored procedure'ı işletmek için Transact-SQL EXECUTE deyimi kullanılır. Bunun dışında eğer stored procedure'ın parametresi varsa o da EXECUTE deyimi ile belirtilir.

EXECUTE (T-SQL) Deyimi

Bir stored procedure'ı işletmek için EXECUTE deyimi kullanılır:

Kullanım Biçimi:

[[EXEC[UTE]]
{
{procedure_adı[;sayı] | @procedure_adı_değişkeni
}
[[@parameter =] {değer| @değişken [OUTPUT] | [DEFAULT]]
[,...n]
[WITH RECOMPILE]

Örnekler: Aşağıdaki örnekte bir komut işletilir:

```
USE master
EXECUTE xp_cmdshell 'dir *.exe'
```

Aşağıdaki örnekte ise bir stored procedure parametre ile çağırılarak çalıştırılır:

```
EXECUTE yazar 'White, 'Jhonson'
```

Adı geçen ve telif haklı olan ürünler bilgi amaçlı olarak kullanılmıştır.

SQL

SQL (Structured Query Language) veri tabanlarındaki verileri işlemek için kullanılan yapısal sorgulama dilidir.

Bu dil yardımıyla veritabanlarındaki tüm işlemler yapılabilir. Backup almadan tutunda bir tabloya veri girmeye varıncaya kadar herşey.

SQL'i şu anda piyasada bulunan hemen hemen her veritabanında kullanabilirsiniz. SQL'de her veritabanında kullanılan ortak ifadeler olmasına karşın, veritabanlarının kendine özgü ifadeleri de vardır. Mesela Oracle'da SQL ile yapabildiğiniz bazı şeyleri başka veritabanlarında yapamayabilirsiniz.

SQL temel olarak şu ifadelerle kullanılır.

SELECT, FROM, WHERE, ORDER BY, GROUP BY, HAVING, UPDATE, DELETE, INSERT.

Burada kullandığımız SQL cümleleri ISCI adlı bir tablo üzerine yazılmıştır. Alanlar.

ISCI_NO	ISCI_ADI	YAS	GIRIS_TARIHI	MAAS

SELECT:

Tablodan seçmek istediğimiz alanları belirtmek için kullanılır. Eğer tablodan tüm alanları seçmek istiyorsak o zaman alan isimleri yerine * işareti konur.

FROM:

Üzerinde işlem yapılacak tablo/tabloları belirtmek için kullanılır.

WHERE:

Tablodan eğer tüm kayıtları değilde istediğimiz bazı kayıtları elde etmek istiyorsak, örnekte maaşı 250 milyondan fazla olan işçilerin numarası ve adi gibi, o zaman buraya istediğimiz kriteri yazarız.

```
SELECT ISCI_NO, ISCI_ADI FROM ISCI WHERE MAAS>250000000
```

DISTINCT:

Birbirinin aynı olan satırların listelenmemesi için bu ifade kullanılır. Mesela ISCI tablosunda bulunan birbirinin aynı olmayan isimleri listelemek istersek

```
SELECT DISTINCT ISCI_ADI FROM ISCI
```

şeklinde bir SQL ifadesi yazarız.

IN:

Koşul belirtirken kullanırız. Mesela ismi AHMET, ALİ veya MUSTAFA olan işçilerin bilgilerini listelemek için

```
SELECT * FROM ISCI WHERE ISCI_ADI='AHMET' OR ISCI_ADI='ALİ' OR ISCI_ADI='MUSTAFA'
```

şeklinde bir ifade kullanırız. Bunun yerine

```
SELECT * FROM ISCI WHERE ISCI_ADI IN ('AHMET' ,'ALİ' ,'MUSTAFA')
```

ifadesini de kullanabiliriz. Yani listenin içindeki herhangi bir değerin bulunması kayıtın seçilmesi için yeterlidir.

LIKE:

Eğer aradığımız kayıtın bulunması için tam bir karşılaştırma yapamıyorsak o zaman kullanırız. Mesela isminin baş harfi A ile başlayan isimleri bulmak için

```
SELECT * FROM ISCI WHERE ISCI_ADI LIKE 'A%'
```

ifadesi kullanılır.

% işareti uzunluğu önemsiz olmak üzere yazıldığı yere her türlü ifade gelebilir anlamındadır.

? işareti ise bir karakter olmak üzere her türlü değeri alabilir anlamındadır. Mesela isminin sondan üçüncü harfi A, ve son harfi Z olan kayıtları listelemek istersek sondan ikinci harfin ne olduğu önemli değildir. O zaman o harf yerine aşağıda görüldüğü üzere ? işaretini kullanırız.

```
SELECT * FROM ISCI WHERE ISCI_ADI LIKE '%A?Z'
```

ifadesi kullanılır.

BETWEEN:

Koşul belirtirken iki değer arasını belirtmek için kullanılır. Örnek: Yaşı 30 ile 40 arasındaki işçilerin kayıtlarını listelemek için

```
SELECT * FROM ISCI WHERE YAS BETWEEN 30 AND 40
```

ifadesi kullanılır. Bunu aynı zamanda aşağıdaki ifade ile de yapabilirsiniz. BETWEEN yazım kolaylığı sağlar.

```
SELECT * FROM ISCI WHERE YAS>=30 AND YAS<=40
```

SUM:

Seçilen değerlerin toplamını bulur. İşçilerin aldığı toplam ücreti görmek için

```
SELECT SUM(UCRET) FROM ISCI
```

ifadesi kullanılır.

MAX, MIN, AVG:

Verilen değerin en büyüğünü, en küçüğünü ve ortalamasını bulur. 1999 yılında giren işçilerin en yüksek ücretinin, en düşük ücretinin ve ortalamasının ne kadar olduğunu öğrenmek istersek aşağıdaki ifadeyi kullanırız.

```
SELECT MAX(UCRET), MIN(UCRET), AVG(UCRET) FROM ISCI WHERE GIRIS_TARIHI>'01.01.1999'
```

MAX en büyük değeri, MIN en küçük değeri, AVG ise seçilen değerlerin ortalmasını bulur.

ORDER BY:

Tablodan seçtiğimiz kayıtları sıralamak için kullanılır. Yukardaki örnekte isimleri alfabetik sıra ile görmek istersek

```
SELECT DISTINCT ISCI_ADI FROM ISCI ORDER BY ISCI_ADI
```

yazarız. Eğer sıralamayı tersine çevirmek istersek

```
SELECT DISTINCT ISCI_ADI FROM ISCI ORDER BY ISCI_ADI DESC
```

yazarız.

GROUP BY:

Genelde istatistik amaçlar için kullanılır. Mesela hangi tarihte kaç işçinin işe alındığını bulmak için

```
SELECT GIRIS_TARIHI,COUNT(*) FROM ISCI GROUP BY GIRIS_TARIHI
```

yazmanız yeterli olacaktır. Bu ifade size gün bazında kaç işçinin işe alındığını gösterecektir. Eğer belli bir tarihten önce ya da sonrasını isterseniz veya sadece sayının 10'dan büyük olduğu günleri görmek isterseniz o zaman ifadeyi şu şekilde yazmak gerekir.

```
SELECT GIRIS_TARIHI,COUNT(*) FROM ISCI WHERE GIRIS_TARIHI>'01.01.1999' GROUP BY
GIRIS_TARIHI HAVING COUNT(*)>10
```

HAVING, grup fonksiyonlarının kriterleri için kullanılır. SUM, COUNT vb. gibi.

UPDATE:

Tabloda bulunan bir istediğiniz bir veya daha fazla alanın güncellenmesi amacıyla kullanılır. Mesela işçilerin maaşlarına % 20 zam yapıldığını düşünürsek aşağıdaki ifade ile bunu tabloda gerçekleştirebiliriz.

```
UPDATE ISCI SET MAAS=MAAS*1.2
```

Eğer maaşlarla birlikte aldıkları primleri de %20 oranında artırmak isterseniz

```
UPDATE ISCI SET MAAS=MAAS*1.2 , PRIM=PRIM*1.2
```

şeklinde bir ifade kullanılır. Aynı zamanda WHERE ifadesini kullanarak sadece belli kayıtlar üzerinde güncelleme yapabilirsiniz.

DELETE:

Tabloda bulunan kayıtları silmek için kullanılır. Eğer

```
DELETE FROM ISCI
```

derseniz tüm kayıtları gönderirsiniz.

DELETE ifadesini kullanırken dikkatli olun. Buradada yine WHERE ifadesini kullanarak sadece belli kritere uyan kayıtların silinmesini sağlayabilirsiniz. Kötü bir örnek ama olsun, patron 45 yaşından büyük işçileri işten attı (burası Türkiye, olmaz demeyin) ve kayıtlarının silinmesi isteniyor. O zaman

```
DELETE FROM ISCI WHERE YAS>45
```

ifadesi kullanılır.

INSERT:

Tablolara veri girişi yapmak amacıyla kullanılır.

```
INSERT INTO ISCI (ISCI_NO,ADI,SOYADI) VALUES (1000,'AHMET','SAVAŞ');
```

Eğer giriş yaparken tablonun bütün alanları kullanılacaksa alan isimlerini vermeye gerek yoktur.

Not:

UPDATE, DELETE ve INSERT ifadelerini kullanırken dikkatli olmalısınız. Eğer SQL tabanlı bir veri tabanı kullanıyorsanız bu ifadeleri veritabanlarının kendi tool'ları üzerinde kullanın. Çünkü Delphi ile gelen SQL Explorer'da işaretine basmayı unutursanız yaptığınız işlemin geri dönüşü olmayabilir. Yani en son yaptığınız işlemi Rollback yapamazsınız ve eğer gerçek veritabanı üzerinde yaptıysanız işlemi başınız bayağı ağrıyabilir veya o iş yerinde yazdığınız son SQL ifadesi olabilir. :-))

İKİ TABLODAN BİRDEN KAYIT SEÇMEK:

İşçilerin kimlik bilgilerinin ISCI_KIMLIK tablosunda tutulduğunu kabul ederek bizden ÇORUM doğumlu olanların listesinin istendiğini varsayalım. Tablolar birbirine ISCI_NO alanı üzerinden ilişkili olsun.

```
SELECT A.ISCI_NO, A.ISCI_ADI, B.DOGUM_YERI FROM ISCI A, ISCI_KIMLIK B WHERE
B.DOGUM_YERI='ÇORUM' AND A.ISCI_NO=B.ISCI_NO
```

şeklinde bir ifade yazarak listemizi elde edebiliriz. Burada WHERE koşuluna yazdığınız sıranın pek bir önemi yoktur. Her şartta aynı sonuçları elde ederseniz. Fakat performans açısından biraz farkeder. Yukardaki ifade

```
SELECT A.ISCI_NO, A.ISCI_ADI, B.DOGUM_YERI FROM ISCI A, ISCI_KIMLIK B WHERE
A.ISCI_NO=B.ISCI_NO B.DOGUM_YERI='ÇORUM'
```

ifadesinden daha hızlı çalışır. Çünkü ilk ifadede önce doğum yeri ÇORUM olan kayıtlar seçilir daha bu kayıtlara işçi tablosu birleştirilir. Sonraki ifadede ise önce tüm kayıtlar birleştirilir, bunların arasından doğum yeri ÇORUM olanlar seçilir.

DISTINCT TEKRARSIZ

TANIM: SQL'de tablo içinde birbirinin aynı datalar bulunabilir.Aynı satırların listeleme esnasında bir kez yazılması

ÖRNEK: 1)Par _sat dosyasından sat_no'ları tekrarsız olarak listelenecektir.

ORDER BY SIRALA

TANIM: Tablodaki sütunlardan ,belirli bir sütuna göre listelemek için SEÇ komutuna, SIRALA eklenir.

ÖRNEK: 1)Personel dosyasından,sicil,ad,soyad,brüt sütunlarını seç ve brüt(maaşa)göre büyükten küçüğe sırala.

```
SELECT sicil,ad,soyad,brüt          SEÇ  sicil,ad,soyad,brüt
FROM personel                       GELİŞ  personel
ORDER BY brüt ASC;                  SIRALA brüt  B-K;
```

DESC Küçükten büyüğe sırala ASC Büyükten küçüğe sırala

İİ)BİRDEN ÇOK ALANA GÖRE SIRALAMA:

TANIM:Bir tablo içinde ,birden fazla sütundan aynı anda sıralamak için kullanılır.

ÖRNEK 1)Personel dosyasından seçilen sütunlarını aynı anda hem ad,hem de otomatik olarak sıralar.

```
SELECT sicil,ad,soyad,brüt          SEÇ sicil,ad,soyad,brüt
FROM personel                       GELİŞ  personel
ORDER BY ad,brüt;                   SIRALA ad,brüt;
```

ÖRNEK 2)Personel tablosundan seçili sütunları öncelik adda olmak üzere (B-K) adı bozmadan soyadı (K-B) sıralı listeler.

```
SELECT sicil,ad,soyad,brüt          SEÇ sicil,ad,soyad,brüt
FROM  personel                      GELİŞ  personel
```

ORDER BY ad ASC,soyad DESC, SIRALA ad B-K,soyad K-B,
brüt ASC; brüt B-K;
veya;
SELECT sicil,ad,soyad,brüt SEÇ sicil,ad,soyad,brüt
FROM personel GELİŞ personel
ORDER BY ad,soyad DESC,brüt; SIRALA ad,soyad K-B,brüt;
DESC'li durumda yanına yazıp belirtilir,yazılmazsa ASC direct kabul edilir.

KOŞULA BAĞLI OLARAK LİSTELEME:

WHERE OLAN
TANIM:Verilen koşulu sağlayanlar listelenir.İki veri birbiriyle karşılaştırılmaktadır. Karşılaştırılan verilerin türü aynı olmalıdır.
SELECT * SEÇ *
FROM personel GELİŞ personel
WHERE brüt > 5000000; OLAN brüt > 5000000;

KARŞILAŞTIRMA OPERATÖRLERİ:

OPERATÖR	ANLAMI
<	...den daha küçük
>	...den daha büyük
=	Eşit
<=	Küçük veya eşit
>=	Büyük veya eşit
<>	Eşit değil
!=	Eşit değil
!<	...den küçük değil
!>	...den büyük değil

ÇEŞİTLİ VERİ TİPLERİ İÇİN BASİT SORGULAMALAR:

i)NÜMERİK VERİ TİPLERİ:

ÖRNEK: Maaşı 8000000TL'den fazla olmayan personeli listele.
SELECT * SEÇ *
FROM personel GELİŞ personel
WHERE brüt <= 8000000 OLAN brüt <= 8000000;

ii)KARAKTER VERİ TİPLERİ (CHAR):

Karakter çift veya tek tırnak ile gösterilir.
ÖRNEK: Adı Ali olmayan personele ait kayıtları listele.
SELECT * SEÇ *
FROM personel GELİŞ personel
WHERE ad <> "Ali"; OLAN ad <> "Ali";

İİİ)TARİH VERİ TİPİ:

Tarih veri tipleri { } sembolleri içinde yazılır.
ÖRNEK: Hangi personelin doğum tarihi 1960 yılından daha öncedir?
SELECT * SEÇ *
FROM personel GELİŞ personel
WHERE dog_tar <={12/31/59} OLAN dog_tar <={12/31/59};

MANTIKSAL (LOJİK) VERİ TİPİ:

Mantıksal veriler için mümkün olabilen sadece iki değer sözkonusudur.DOĞRU D(TRUE T) , YANLIŞ Y (FALSE F) ile simgelenir.
ÖRNEK: Personel tablosunda personelin cinsiyetini belirten cins adlı alan mantıksal(logical) olarak tanımlanmıştır.Cinsiyeti erkek olanları D,kadın olanları y ile tanımlarsak erkek olanları listele.
SELECT * SEÇ *
FROM personel GELİŞ personel
WHERE cins = .T.; OLAN cins =.D.;

BİRDEN ÇOK KOŞULA DAYALI SORGULAMALAR: (NOT,AND,OR)

TANIM:Mantıksal operatörlerin yardımı ile birden çok koşulun gerçekleştirmesine bağlı olarak ifade edilebilecek (karmaşık yada birleşik koşullu listelemeleri gerçekleştirilmektedir.)
AND VE

ÖRNEK:Maaşı 5000000'dan fazla olan ve cinsiyeti erkek olan personelin listelenmesi istenir yani iki koşul verilmektedir ve ikisininde olması istenir.

```
SELECT *                                SEÇ  *
FROM personel                           GELİŞ  personel
WHERE  brüt >5000000 AND cins =.T.;     OLAN brüt > 5000000 AND cins =.D.
NOT          DEĞİL
OR           VEYA
```

ÖRNEKLER:

i)Doğum tarihi 1960'dan önce olan maaşı 6000000 – 10000000 arasındaki bayan personelin listele.

```
SELECT  *                               SEÇ  *
FROM dog_tar <  {01/01/60} AND          GELİŞ dog_tar < {01/01/60} VE
brüt > = 6000000 AND brüt < =10000000   brüt > = 6000000 VE brüt < =10000000
AND cins = .F.;                         VE cins =.Y.;
```

İİ)Satış bölümüyle muhasebe bölümündekiler kimlerdir?
(Satış bölümünün böl_no'sunun 1 ve muhasebe bölümünün böl_no'sunun 2 olduğu varsayılmaktadır.)

```
SELECT *                                SEÇ  *
FROM  personel                          GELİŞ  personel
WHERE  bol_no =1 OR  bol_no = 2;        OLAN bol_no = 1  VEYA bol_no =2;
```

İİİ)Bölümü Satış yada Muhasebe olamayan 1960'dan sonra doğmuş bayan personeli listele.

1.YAZILIM:

```
SELECT *                                SEÇ  *
FROM  personel                          GELİŞ  personel
WHERE  NOT (böl_no =1 OR                OLAN DEĞİL (böl_no =1 VEYA
böl_no =2) AND dog_tar > ={01/01/60}        böl_no =2)VE dog_tar >={01/01/60}
AND cins=.F.;                           VE cins=.Y.;
```

2.YAZILIM:

```
SELECT  *                               SEÇ  *
FROM  personel                          FROM  personel
WHERE  böl_no  <> 1 AND                 OLAN böl_no <> 1 VE
böl_no <> 2 AND dog_tar > ={01/01/60}       böl_no <> 2 AND dog_tar > = {01/01/60}
AND  cins =.F.;                         VE cins =.Y.;
```

BİR VERİ KÜMESİNDE ARAMA –IN OPERATÖRÜ

IN İÇİNDE

"IN" operatörü DEĞİL(NOT) ile kullanılılabilir.

ÖRNEK:i) Bölümü 1,2,3 olmayan personel kimlerden oluşmaktadır?

```
SELECT *                                SEÇ  *
FROM  personel                          GELİŞ  personel
WHERE  bol_no  NOT IN (1,2,3);          OLAN böl_no  DEĞİL İÇİNDE (1,2,3);
```

ÖRNEK:ii) Böl_no'su 1,2 yada 3 olan personeli listele.

```
SELECT  *                               SEÇ  *
FROM  personel                          GELİŞ  personel
WHERE böl_no = 1  OR  böl_no= 2  OR     OLAN böl_no =1 VEYA böl_no =2 VEYA
böl_no=3;                               böl_no = 3;
```

Bu örneğin IN ile yapılmış şekli daha kısadır.

```
SELECT *                                SEÇ  *
FROM   personel                         GELİŞ  personel
WHERE  NOT  böl_no  IN (1,2,3);         OLAN DEĞİL böl_no İÇİNDE (1,2,3);
```

ARALIK SORGULAMA SÖZCÜĞÜ:

BETWEEN ARASINDA

ÖRNEK:Maaşı 5- 10 milyon arasında olan personel kimlerdir?

```
SELECT  *                               SEÇ  *
FROM  personel                          GELİŞ  personel
WHERE  brüt  > =5000000  AND            OLAN brüt > =5000000  VE
brüt < = 10000000;                      brüt < = 10000000;
```

BETWEEN (ARASINDA) komutu ile daha kısa olacaktır.

SELECT * SEÇ *
FROM personel GELİŞ personel
WHERE brüt BETWEEN 5000000 OLAN brüt ARASINDA 5000000
AND 10000000; VE 10000000;

KARAKTER TÜRÜ BİLGİ İÇİNDE ARAMA YAPMA –LIKE SÖZCÜĞÜ:

TANIM ÖRNEĞİ: Adres sutunu içerisinde semt bölümüne ait ayrıca bir sutun olmadığını varsayarak semt adı adres sutunu içerisinde yer alır ve buradan da LIKE (BULUNAN) komutuyla adres sutunu içerisinde Taksim semtinde oturan personeli listele.

SELECT * SEÇ *
FROM personel GELİŞ personel
WHERE adres LIKE '% TAKSİM %' ; OLAN adres LIKE '% TAKSİM%' ;

Adres LIKE '%TAKSİM%' ifadesi adres içinde her hangi bir yerde TAKSİM yazan yerde oturan personeli listeleyecektir.

LIKE sözcüğünü ,alt çizgi (-) sembolü ile birlikte kullanmakta mümkündür.

SELECT * SEÇ *
FROM personel GELİŞ personel
WHERE ad LIKE 'Mehmet -----'; OLAN ad BULUNAN 'Mehmet ----';

Şekildeki komut ile ad alanı "Mehmet " ile başlayan ve ad alanı uzunluğu 10 karakter olan isimlere sahip personeli listeleyecektir."Mehmet Ali","Mehmet Can"- "Mehmetcik" gibi isimler listeleyecektir.Anlaşılacağı gibi - sembolü , tek karakterlik bir bilgiyi temsil etmektedir.

SQL'DE ARİTMETİKSEL İFADELER VE FNKSİYONLAR :
KÜME FONKSİYONLARI:

SUM FONKSİYONU:

SUM TOPLA

Fonksiyonun argümanı olarak belirtilen sütun ile ilişkili olana toplama işlemini gerçekleştirir.

ÖRNEK:İşletmedeki personelin brüt maaşlar toplamı ne kadardır?

SELECT SUMbrüt) SEÇ TOPLA(brüt)
FROM personel; GELİŞ personel;

AVG FONKSİYONU:

AVG ORT

Aritmetiksel ortalama (average) hesaplamak için kullanılır.

SELECT AVG(brüt) SEÇ ORT (brüt)
FROM personel; GELİŞ personel;

MAX FONKSİYONU:

MAX ÜST

Tablo içinde ,belirtilen sutun (alan)içindeki en büyük değeri bulur.

ÖRNEK:İşletme içindeki en yüksek maaş ne kadardır?

SELECT MAX (brüt) SEÇ ÜST (brüt)
FROM personel; GELİŞ personel;

MIN FONKSİYONU:

MIN ALT

Tablo içinde,belirlenen sutun alan içindeki en küçük değeri bulur.

ÖRNEK:İşletme içinde 4 Mayıs 1970'den önce doğanlar için,asgari ücret nedir?

SELECT MIN(brüt) SEÇ ALT(brüt)
FROM personel GELİŞ personel
WHERE dog_tar<05/04/70}; OLAN dog_tar < {05/04/70};

COUNT FONKSİYONU:

COUNT SAY

Tablo içinde ,her hangi bir sayma işlemi gerçekleştirmek için kullanılır.

ÖRNEK:Ücreti 6000000'dan olan personel sayısı nedir?

SELECT COUNT (*) SEÇ SAY(*)
FROM personel GELİŞ personel
WHERE brüt>6000000; OLAN brüt > 6000000;

COUNT (SAY) fonksiyonu DISTINCT (TEKRARSIZ)sözcüğü ile de kullanılır.

ÖRNEK:Personel tablosunda mevcut personelin işletme içinde kaç tane farklı bölümde çalıştığını bul.

SELECT COUNT(DISTINCT böl_no) SEÇ SAY (TEKRARSIZ böl_no)
FROM personel; GELİŞ personel;
COUNT (böl_no) SAY (böl_no)

GRUPLANDIRARAK İŞLEM YAPMA:

GROUP BY GRUPLA
ÖRNEK: Her bölümdeki ortalama maaş nedir?
SELECT böl_no,AVG (brüt) SEÇ böl_no
FROM personel GELİŞ personel
GOUP BY böl_no; GRUPLA böl_no;
HAVING:
HAVING SAHİP
Gruplandırarak kümeleme fonksiyonunu uygularken koşulda verilebilir.Bu durumda grup üzerindeki hesaplamalarla ilgili koşul belirtilirken HAVING (SAHİP) sözcüğü kullanılır.
ÖRNEK:En yüksek maaşın 9000000'dan fazla olduğu bölümlerdeki personele ait ortalama maaşları listele.
SELECT böl_no,AVG (brüt) SEÇ böl_no, ORT(brüt)
FROM personel GELİŞ personel
GROUP BY böl_no GRUPLA böl_no
HAVING AVG(brüt)> 9000000; SAHİP ORT(brüt)> 9000000;
HAVING(SAHİP) sözcüğü SELECT(SEÇ) konusunda GROUP BY(GRUPLA) bulunmadığı zaman geçersizdir.HAVING(SAHİP) sözcüğünü izleyen ifade içinde ,SUM(TOPLA), COUNT(*)(SAY),AVG(ORT),MAX(ÜST) yada MIN(ALT) fonksiyonlarından en az biri bulunmalıdır.
HAVING (SAHİP) sözcüğü sadece gruplanmış veriler üzerindeki işlemlerde geçerlidir.
WHERE (OLAN) sözcüğü bir tablonun tek tek satırları üzerinde işlem yapan koşullar içinde geçerlidir.
Bazı durumlarda HAVING(SAHİP) ve WHERE(OLAN) sözcükleri ile birlikte SELECT(SEÇ) komutu içinde kullanılabilir.
ÖRNEK:Personel tablosu içinde her bölümde erkek personele ait maaşlar için ortalamanın 9000000'dan fazla olduğu bölümleri listele.
SELECT böl_no, AVG(brüt) SEÇ böl_no, ORT (brüt)
FROM personel GELİŞ personel
WHERE cins= .T. OLAN cins= .D.
GROUP BY böl_no GRUPLA böl_no
HAVING AVG (brüt) > 9000000; SAHİP ORT(brüt) > 9000000;

BİRDEN FAZLA TABLOYU İLİŞKİLENDİRMEK:

JOIN İLİŞKİLENDİR
ÖRNEK: Personel ve bölüm adlı 2 tablo bulunmaktadır.
Çalışan her personel ve personelin yöneticisi ile ilişkili bilgiler nelerdir?
SELECT * SEÇ *
FROM personel,bölüm GELİŞ personel,bölüm
WHERE personel .böl_no=bölüm.bölüm_no ; OLAN personel.böl_no = bölüm.bölüm_no;
SELECT sicil,ad,soyad,böl_no,yön_s_g_n SEÇ sicil,ad,soyad,böl_no,yön_s_g_n
FROM personel,bölüm GELİŞ personel,bölüm
WHERE personel .böl_no = bölüm .bölüm_no; OLAN personel .böl_no = bölüm.bölüm_no;
SELF-JOIN: KENDİSİYLE -İLİŞKİLENDİR
TANIM:Bir tablonun kendisi ile birleştirilmesine "KENDİSİYLE-İLİŞKiLENDİR" denir.(SELF-JOIN)
SELECT A. sicil , A.ad , A.soyad,B .ad , B.soyad , B.dog_tar
SEÇ A. sicil , A.ad , A.soyad, B .ad , B.soyad , B.dog_tar
FROM personel A , personel B GELİŞ personel A , personel B
WHERE A. yon_sos_g_n =B .sosy_g_no; OLAN A. yon_sos_g_n =B .sosy_g_no;

NESTED SELECTS:

İÇİÇE
TANIM:İç içe geçmiş SELECT(SEÇ)komutlarından oluşur.İçteki. seç komutunun bulduğu sonucu dış takı SEÇ komutumuz işlevini yerine getirmesi için kullanılır.
ÖRNEK:Parça numarası 24 olan parçayı ,projelerde kullanan çalışan personeli listele.
SELECT * SEÇ *

```
FROM  personel                          GELİŞ personel
WHERE sosy_g_no                         OLAN sosy_g_no
IN(SELECT per_s_g_no                    İÇİNDE(SEÇ per_s_g_no
FROM  parça,proje,çalışma               GELİŞ parça,proje,çalışma
WHERE pr_no = proj_no AND                                  OLAN pr_no = proj_no
VE
      proj_no =proj_no AND                          proj_no = proj_no VE
      par_no =24);                                    par_no =24);
```

ÖRNEK: Fatih'te oturan personelin çalıştığı projelerin adlarını ve yerlerini listele.

```
SELECT   proj_ad,yer                    SEÇ  proj_ad,yer
FROM  proje                             GELİŞ  proje
WHERE  proj_no  IN                      OLAN proj_no İÇİNDE
(SELECT proje_no                        (SEÇ proje_no
FROM personel,çalışma                   GELİŞ sosy_g_no = per_s_g_no
WHERE  sosy_g_no = per_s_g_no           OLAN  sosy_g_no = per_s_g_no
AND adres LIKE "% fatih %");            VE adres BULUNAN "% fatih %);
```

UNION SÖZCÜĞÜ:

UNION BİRLEŞİM

TANIM:İki ayrı SEÇ komutunun sonucunda elde edilen tabloların birleşimi işlemini gerçekleştirir.

ÖRNEK:Adı Ahmet ve Soyadı Caner olan kişi yada kişileri işletmenin yürüttüğü projelerde çalışan bir kişi (sıradan bir personel yada bölüm yöneticisi)olarak bulunduran projelerin isimlerini ve projelerin yürütüldüğü yerleri listele.

```
(SELECT proj_ad,yer                     (SEÇ proj_ad,yer
FROM proj,bölüm,personel                GELİŞ  proj,bölüm,personel
WHERE  bl_no=bölüm_no AND               OLAN bl_no=bölüm_no VE
       y_sos gno = sosy_g_no                  y_sos gno = sosy_g_no
       AND ad ="Ahmet"AND soyad ="Caner")          VE ad ="Ahmet" VE  soyad ="Caner")
UNION (SELECT proj_ad,yer               BİRLEŞİM (SEÇ proj_ad,yer
FROM proje,çalışma,personel             GELİŞ proje,çalışma,personel
WHERE proj_no = proje_no AND            OLAN proj_no = proje_no VE
Per_s_g_no = sosy_g_no AND ad ="Ahmet"  Per_s_g_no = sosy_g_no VE  ad "Ahmet"
            AND soyad="Caner")                    VE  soyad ="Caner")
```

KOŞULLAR:

UNION (BİRLEŞİM) sözcüğü ile ,iki yada daha çok kişi SELECT (SEÇ)'in sonucu olan tabloların küme birleşimi işlemine tabi tutulması için 2 koşul gereklidir.

1)SELECT (SEÇ) komutları sonucunda elde edilecek tablolar aynı sayıda kolon içermelidirler.

2)Sonuç tabloları karşılıklı olarak kolonların aynı veri tipi ve aynı genişlikte olmalıdır.

ANY :

ANY HER HANGİ BİRİ

ÖRNEK:Satış bölümünde çalışan personelin her hangi birinden daha düşük maaş alan ve mühendislik bölümündeki kişileri listele.

```
SELECT  *                                SEÇ  *
FROM personel                            GELİŞ personel
WHERE  brüt < ANY                        OLAN brüt < HER HANGİ BİRİ
(SELECT brüt                             (SEÇ brüt
FROM personel                            GELİŞ personel
WHERE böl_no = 2)  AND böl_no = 1;       OLAN böl_no = 2) VE böl_no =1;
```

EŞ DEĞERİ İFADE:

```
SELECT  *                                SEÇ  *
FROM  personel                           GELİŞ  personel
WHERE brüt  <  (SELECT MAX (brüt )       OLAN brüt < (SEÇ ÜST (brüt )
FROM personel                            GELİŞ personel
WHERE böl_no = 2) AND  böl_no =1;        OLAN böl_no = 2) VE  böl_no =1;
```

ALL:

ALL HEPSİ

ÖRNEK:Satış bölümünde çalışan ve mühendislik bölümündeki personelin hepsinden daha fazla maaş alan personeli listele.Bu örnekte satış bölümü kodu = 2 ve mühendislik bölümü kodu = 1 alınmıştır.

YAPILIŞ YOLU:

```
1)SELECT *                                  SEÇ *
  FROM personel                               GELİŞ personel
 WHERE brüt >                                 OLAN brüt >
 ALL (SELECT brüt                            HEPSİ (SEÇ brüt
 FROM personel                                 GELİŞ personel
 WHERE böl_no = 1) AND böl_no = 2;             OLAN böl_no =1) VE böl_no =2;
2)SELECT *                                  SEÇ *
  FROM personel                               GELİŞ personel
 WHERE brüt >                                 OLAN brüt >
(SELECT MAX (brüt)                          (SEÇ ÜST (brüt)
FROM personel                                 GELİŞ personel
WHERE böl_no = 1) AND böl_no =2;              OLAN böl_no = 1) VE böl_no =2;
```

EXISTS:

EXISTS MEVCUT

VE ,VEYA ,DEĞİL operatörleri ile kullanılabilir.

ÖRNEK: 27 no'lu parçayı satan satıcılarla ilişkili tüm bilgileri listele.

```
SELECT *                                    SEÇ *
FROM satıcı                                   GELİŞ satıcı
WHERE EXISTS                                OLAN MEVCUT
   (SELECT *                                  (SEÇ *
   FROM par_sat                                GELİŞ par_sat
   WHERE sat_no = satıcı_n                     OLAN sat_no = satıcı_n
   AND parça_n =27);                           VE parça_n = 27);
```

NOT EXISTS:

NOT EXISTS MEVCUT DEĞİL

VE ,VEYA ,DEĞİL operatörleri ile kullanılabilir.

ÖRNEK: 27 no'lu parçayı satmayan satıcılar kimlerdir?

```
SELECT *                                    SEÇ *
FROM satıcı                                   GELİŞ satıcı
WHERE NOT EXISTS                            OLAN MEVCUT DEĞİL
   (SELECT *                                    (SEÇ *
   FROM par_sat                                  GELİŞ par_sat
   WHERE sat_no = satıcı_n                       OLAN sat_no = satıcı_n
   AND parça_n =27);                             VE parça_n = 27);
```

EXCEPT:

EXCEPT FARKLI

Tablo-1 - Tablo-2 işlemi sonuç(iki kümenin farkı) elde edilecek tabloda,Tablo-1'de bulunup, Tablo-2'de bulunmayan veriler mevcut olacaktır.

ÖRNEK:Satış bölümündeki personel adlarından,mühendislik bölümünde bulunmayanları listele.

```
SELECT * FROM                               SEÇ * GELİŞ
(SELECT ad FROM personel                      (SEÇ ad GELİŞ personel
WHERE bol_no=1                                OLAN böl_no = 1
EXCEPT                                      FARKLI
SELECT ad FROM personel                       SEÇ ad GELİŞ personel
WHERE bol_no =2);                             OLAN böl_no =2);
```

INTERSECT:

INTERSECT KESİŞİM

ÖRNEK: Hem Ankara'da,hem de İstanbul'daki projelerde görev alan bölümleri listele.

```
SELECT * FROM                               SEÇ * GELİŞ
(SELECT bl_no FROM proje                      (SEÇ bl_no GELİŞ proje
WHERE yer LIKE "%Ankara%"                   OLAN yer BULUNAN "%Ankara%"
INTERSECT                                   KESİŞİM
SELECT bl_no FROM proje                       SEÇ bl_no GELİŞ proje
```

WHERE yer LIKE “%İstanbul%”); OLAN yer BULUNAN “%İstanbul%”);
SAVE TO TEMP:
SAVE TO TEMP SAKLA
ÖRNEK: Bayan personeli,bayan adlı bir tablo içinde sakla.
SELECT * SEÇ *
FROM personel GELİŞ personel
WHERE cins =.F. SAVE TO TEMP bayan; OLAN cins =.Y. SAKLA bayan;
KEEP:
KEEP KALICI
ÖRNEK:
SELECT * SEÇ *
FROM personel GELİŞ personel
WHERE cins = .F. OLAN cins =.Y.
SAVE TO TEMP bayan KEEP; GEÇİCİ SAKLA bayan KALICI;
TABLOLARDA DEĞİŞİKLİK YAPMAK:
INSERT:
INSERT EKLE
INTO İÇİNE
VALUES DEĞERLER
ÖRNEK:Bir personel tablosuna sicil_no’su 275 olan personel ile ilişkili bilgileri ekle.
INSERT INTO personel(sicil, EKLE İÇİNE personel(sicil,
sosy_g_no,ad,soyad,doğ_tar sosy_g_no,ad,soyad,doğ_tar
adres,cins,brüt,böl_no,yön_s_g_no adres,cins,brüt,böl_no,yön_s_g_no
VALUES(‘275’,’27652418’,’Ali’,’Caner’, DEĞERLER (‘275’,’27652418’,’Ali’,’Caner’,
{10/05/1962},’Merkez caddesi 46 –Fatih-İstanbul’, {10/05/1962},’Merkez caddesi 46 –Fatih-İstanbul’,
.T.,27000000,2,’876215342’); .D.,27000000,2,’876215342’);
DELETE:
DELETE SİL
ÖRNEK:2 no’lu bölümdeki personelin tümü tablodan sil.
DELETE FROM personel SİL GELİŞ personel
WHERE böl_no = 2; OLAN böl_no = 2;
5 ROWS DELETED 5 SATIR SİLİNDİ
ÖRNEK:Brüt maaş alanı boş olmayan tüm personeli sil.
DELETE FROM personel SİL GELİŞ personel
WHERE brüt IS NOT NULL; OLAN brüt DEĞERSİZ;
25 ROWS DELETED 25 SATIR SİLİNDİ
UPDATE :
UPDATE GÜNCELLE
SET YAP
ÖRNEK:2’inci bölümün yürüttüğü projelerde kullanılan tüm parçaların fiyatlarını % 7 zam yap.
UPDATE parça GÜNCELLE parça
SET fiyat = fiyat *1,07 YAP fiyat = fiyat *1,07
WHERE pr_no IN OLAN pr_no İÇİNDE
(SELECT proj_no (SEÇ proj_no
FROM proje GELİŞ proje
WHERE bl_no = 2; OLAN bl_no =2 ;
CREATE INDEX:
CREATE INDEX INDEKS YARAT
ON İÇİN
CREATE INDEX ındeks adı INDEKS YARAT ındeks adı
ON tablo adı(kolon adı 1,kolon adı 2,.,.kolon adı n);
İÇİN tablo adı(kolon adı 1,kolon adı 2,.,.kolon adı n);

TEK BİR ALAN A GÖRE ARTAN SIRADA İNDEKSLEME :

ÖRNEK:İşletmede çalışan personeli brüt maaşlarına göre artan sırada listele.(Brüt alana göre bir indeks oluşturmalıyız)
CREATE INDEX pers_maas INDEKS YARAT pers_maas

ON personel(brüt); İÇİN personel(brüt);
INDEX CREATED 127 ROWS İNDEKS YARATILDI 127 SATIR
127 satırlık personel tablosu ile ilişkili olarak brüt kolonu indeks anahtarı olarak kullanan pers_maas adlı indeks oluşturulmuştur.Bu durumda;
SELECT * SEÇ *
FROM personel; GELİŞ personel;
Şeklinde listeleme komutu sonucunda personel tablosundaki tüm personel, brüt maaşlarına göre sıralı olarak listelenecektir.

TEK BİR ALANA GÖRE AZALAN SIRADA İNDEKSLEME :

DESC Küçükten büyüğe (K-B)
ÖRNEK:İşletmede çalışan personeli brüt maaşlarına göre azalan sırada (yüksek maaştan düşük maaşa doğru)listelemek istersek ,brüt alanına göre aşağıdaki şekilde oluşturmak gerekir.
CREATE INDEX INDEKS YARAT
ON personel (brüt DESC); İÇİN PERSONEL(BRÜT K-B);

BİRDEN FAZLA ALANA GÖRE İNDEKSLEME :

ÖRNEK:İşletmedeki personelin öncelikle adlarına göre,aynı adda olanların soyadlarına göre ,hem adı hemde soyadı aynı olanların maaşlarına göre sıralanmış olarak listele.
CREATE INDEX p_ad_soy_m İNDEKS YARAT p_ad_soy_m
ON personel(ad,soyad,brüt); İÇİN personel (ad,soyad,brüt);
Bu durumda;
SELECT * SEÇ *
FROM personel; GELİŞ personel; ile tablo görüntülenir.

UNİQUE SÖZCÜĞÜ:

UNIQUE TEK
Bir tablo,seçilen bir sutüna (alana) göre indekslenirken , indeksleme alanı olarak seçilen sutündaki verilerintekrarlanmasına müsaade edilmesi istenmiyorsa,indeksleme yapılırken ,CREATE ,INDEX komutu iinde UNİQUE sözcüğü kullanılmalıdır.
CREATE UNİQUE İNDEX pers_sicil TEK INDEKS YARAT pers_sicil
ON personel (sicil); İÇİN personel (sicil);

EKLEME İÇİN:

Personel tablosuna
INSERT INTO Personel EKLE İÇİNE Personel
VALUES(53768 ,'27241685','ayşe', DEĞERLER (53768 ,'27241685','ayşe' ,
'şen'{01/04/63},'Merkez cad. 82 – 'şen'{01/04/63},'Merkez cad. 82 –
Kadıköy'.F. ,27000000 ,2, '34261578'); Kadıköy'.Y. ,27000000 ,2, '34261578');

MEVCUT BİR İNDEKSİN SİLİNMESİ:

DROP IPTAL
DROP INDEX pers_in; İPTAL İNDEKS pers_in;
Komutu ile
INDEX DROPPED (İNDEKS SİLİNDİ)
TABLONUN YAPISINDA DEĞİŞİKLİK YAPMAK:
ALTER TABLE TABLO DEĞİŞTİR

MEVCUT BİR TABLOYA KOLON EKLEMEK:

ADD EKLE
DATE TARİH
ALTER TABLE (TABLO DEĞİŞTİR) komutu içinde ADD (EKLE) ile satır ekle.
ÖRNEK:Personel tablosuna ,işe başlama tarihini belirten bir kolon ekle
ALTER TABLE personel TABLO DEĞİŞTİR personel
ADD iş_baş_tar DATE; EKLE iş_baş_tar TARİH;
ADD (EKLE)iş_baş_tar DATE NOT NULL (TARİH DEĞERSİZ) bu şekilde kullanılsaydı bu kolon satırı gene boş kalırdı ; fakat bu kolon ile ilişkili yeni boş değerler eklemek istendiğinde buna müsaade edilmeyecekti.

MEVCUT BİR TABLONUN KOLONLARINDA DEĞİŞİKLİK YAPMAK :

MODIFY KOMUTU:
MODIFY ONAR

MEVCUT BİR TABLODAN BİR KOLON SİLMEK:

DROP KOMUTU :

DROP İPTAL
ÖRNEK:Personel tablosundan iş_baş_tar kolonunu sil.
ALTER TABLE personel TABLO DEĞİŞTİR personel
DROP iş_baş_tar ; İPTAL iş_baş_tar;
Birden fazla kolonda silinebilir.Birden fazla kolon silmek için virgülle ayrılarak silinir.

BİR TABLONUN ADINI DEĞİŞTİRMEK:

RENAME KOMUTU:
RENAME TABLO YENİ AD
ALTER TABLE personel TABLO DEĞİŞTİR personel
RENAME TABLE elemanlar; TABLO YENİ AD elemanlar;

MEVCUT Bİ TABLONUN BİR KOLONUNUN ADININ DEĞİŞTİRİLMESİ:

RENAME:
RENAME YENİ AD
ALTER TABLE personel
RENAME brüt br-maaş;

MEVCUT BİR TABLONUN TÜMÜYLE SİLİNMESİ

DROP TABLE TABLO İPTAL
ÖRNEK:Proje tablosunu sil.
DROP TABLE proje; TABLO İPTAL proje;

VERİ GÜVENLİĞİ:

CREATE VİEW GÖRÜŞ ALANI YARAT
ÖRNEK:Personel adlı temel tablodan persview adlı bir view oluştur.
CREATE VİEW perswiew GÖRÜŞ ALANI YARAT persview
AS SELECT sicil,sos_g_no,ad,soyad,doğ_tar, GİBİ SEÇ sicil,sos_g_no,ad,soyad,doğ_tar,
adres,cins,böl_no,yon_s_g_no adres,cins,böl_no,yon_s_g_no
FROM personel; GELİŞ personel;

VERİ BÜTÜNLÜĞÜNÜN SAĞLANMASI:

WITH CHECK OPTİON KONTROLLÜ
CREATE VIEW UST_PER_ VIEW Önce bir vıew oluşturulsun
AS SELECT FROM personel
WHERE brüt >25000000
WITH CHECK OPTİON;
GÖRÜŞ ALANI YARAT UST_PER_VIEW
GİBİ SEÇ GELİŞ personel
OLAN brüt >25000000
KONTROLLÜ;
Burada, maaşı 25000000'ün üzerinde olan personelden oluşan bir UST_PER_VIEW adlı view oluşturulmuştur.Bu vıew'a brüt maaşı 13000000 olan bir personel eklemek istediği zaman hata mesajı verecektir.
CHECK opsiyonu kullanılmasaydı hata mesajı alınmadan bu veri VİEW içine yükleyecekti.
EKLEME
INSERT INTO UST_PER_VIEW EKLE İÇİNE UST_PER_VIEW
VALUES (27521 ,'27865427','ayşe', DEĞERLER (27521 ,'27865427','ayşe',
'okan' ,{01/05/1962}'Cumh. Cad. 46 – Taksim', 'okan' ,{01/05/1962}'Cumh. Cad. 46 – Taksim',
.F.,13000000 ,1 ,'27651112'); .F.,13000000 ,1 ,'27651112');
VIEW İÇİNDE SATIR SİLME:
ÖRNEK:UST_PER_VIEW içinden,maaşı 2500000'den az olan kişileri sil.
DELETE FROM UST_PER_VIEW SİL GELİŞ UST_PER_VIEW
WHERE brüt < 25000000; OLAN brüt < 25000000;

VIEW SATIRLARI ÜZERİNDE GÜNCELLEME :

ÖRNEK: UST_PER_VIEW adlı view'de sicili 27251 olan kişnin maaşını 37000000 olarak değiştir.
UPDATE UST_PER_VIEW GÜNCELLE UST_PER_VIEW
SET brüt = 37000000 YAP brüt = 37000000
WHERE sicil = 27251; OLAN sicil = 27251;
BİR VIEW'U SİLMEK:
DROP VIEW GÖRÜŞ ALANI İPTALİ

DROP VIEW UST_PER_VIEW; GÖRÜŞ ALANI İPTALİ UST_PER_VIEW;
SQL'in amacı bu komutları yan yana yazdığımızda bir cümlenin ortaya çıkmasıdır.
SELECT .... FROM .... WHERE ....,ORDER BY ....,GROUP BY .....HAVING....
.....(DEN).....GELİŞ.....OLAN.. ,.....SIRALA....,GRUPLA.....SAHİP....SEÇ