

Playlist: <https://www.youtube.com/playlist?list=PLeZr8VTNC1obmhNWF44Yz6vANbCd2ahk2>

Live SQL URL: <https://livesql.oracle.com>

SQL-1 SQL'e GİRİŞ

05.10.2020

Veritabanı genellikle elektronik olarak bir bilgisayar sisteminde depolanan yapılandırılmış bilgi veya veriden oluşan düzenli bir koleksiyondur.

Veri tabanı genellikle bir Veri Tabanı Yönetim Sistemi DBMS (**D**ata**B**ase**M**anagement**S**ystem) ile kontrol edilir.

Çoğu veri tabanında veri yazma ve sorgulama için yapılandırılmış sorgu dili **SQL** (**S**tructured **Q**uery **L**anguage) kullanılır.

SQL, verileri yönetmek ve tasarlamak için kullanılan bir dildir. SQL, kendisi bir programlama dili olmamasına rağmen birçok kişi tarafından programlama dili olarak bilinir. SQL herhangi bir veri tabanı ortamında kullanılan bir alt dildir. SQL ile yalnızca veri tabanı üzerinde işlem yapılabilir; veritabanlarında bulunan sistemlere bilgi ekleme, bilgi değiştirme, bilgi çıkarma ve bilgi sorgulama için kullanılmaktadır. Özellikle de ilişkisel veritabanı sistemleri üzerinde yoğun olarak kullanılmaktadır. SQL'e özgü cümleler kullanarak veri tabanına kayıt eklenebilir, olan kayıtlar değiştirilebilir, silinebilir ve bu kayıtlardan listeler oluşturulabilir.

Structured Query Language (SQL) - Yapılandırılmış Sorgu Dili - Strukturierte Abfragesprache.

Database' in Faydaları;

1. Yüksek miktarda bilgi depolanabilir.
2. Oluşturmak, okumak, değiştirme ve silme kolaylığı. **Create**, **Read**, **Update**, **Delete** (**CRUD**)
3. Girişin kolay ve kontrollü olması.
4. Dataya ulaşım kolaylığı.
5. Güvenlik

Dersler başlamadan; ilk olarak <https://livesql.oracle.com/> sitesinden kullanıcı hesabi oluşturulmalı.

Start Coding Now linkine tıklanır. Kullanıcı hesabi yoksa oluşturulur. Varsa Sign in yapılır.

SQL Worksheet penceresine aşağıdaki script yapıştırılır;

```
CREATE TABLE department
(
  id                number(5) constraint pk_department primary key,
  name              varchar2(50),
  monthly_budget    number(8,2),
  last_employee_id  number(5)
);

insert into department values (1,'ACCOUNTING',20000,8);
insert into department values (2,'MARKETING',15000,9);
insert into department values (3,'INFORMATION TECHNOLOGY',30000,10);
insert into department values (4,'HUMAN RESOURCES',25000,13);
insert into department values (5,'REGULATORY AFFAIRS',5000,null);
insert into department values (6,'CUSTOMER SERVICE',2000,null);
```

Ardından Run edilir. **Tablo** olarak görebilmek için **"select * from department;"** yazılıp run edilir.

ID	NAME	MONTHLY_BUDGET	LAST_EMPLOYEE_ID
1	ACCOUNTING	20000	8
2	MARKETING	15000	9
3	INFORMATION TECHNOLOGY	30000	10
4	HUMAN RESOURCES	25000	13
5	REGULATORY AFFAIRS	5000	-
6	CUSTOMER SERVICE	2000	-

Oluşan tabloyu silmek için **Schema**'ya gidilir, tablo tıklanır, Actions menüsünden **Drop** secilir. (Delete ya da remove kullanılmaz) Gelen code run edilirse tablo silinmiş olur.

```
select E.first_name, E.salary from employees E
join departments D on E.department_id = D.department_id
join locations L on D.location_id = L.location_id
where E.salary > (select max(salary) from employees E
                  join departments D on E.department_id = D.department_id
                  join locations L on D.location_id = L.location_id
                  where L.city = 'Toronto')
and L.city <> 'Toronto';
```

```
select max(salary) from employees E
                  join departments D on E.department_id = D.department_id
                  join locations L on D.location_id = L.location_id
                  where L.city = 'Toronto';
```

```
select E.first_name, E.salary from employees E
join departments D on E.depatment_id = D.department_id
join locations L on D.location_id = L.location_id
where E.salary > (select max(salary) from employees E
                  join departments D on E.depatment_id = D.department_id
                  join locations L on D.location_id = L.location_id
                  where L.city = 'Toronto')
and L.city <> 'Toronto';
```

```
select max(salary) from employees E
                  join departments D on E.depatment_id = D.department_id
                  join locations L on D.location_id = L.location_id
                  where L.city = 'Toronto';
```

Database Validation (Doğrulama) Testi

Tester olarak bizler; **Database Validation (Doğrulama) Testi** yaparız, database oluşturmayız, kullanıcılara yetki vermeyiz, raporlama yapmayız, tablo silmeyiz. Buna **End To End (E2E) Testing** de denir.

Datayı User Interface (UI) kullanarak, SQL kodlarını kullanarak veya API kodlarını kullanarak da yollasak; Üç çeşit Database Validation Testi vardır,

1. Datayı UI dan arama fonksiyonunu kullanarak doğrulama (Selenium)
2. Datayı SQL kodlarını kullanarak doğrulama (SQL + Selenium)
3. Datayı API kodlarını kullanarak doğrulama (API + Selenium)

Application Programming Interface (API) bir uygulamaya ait yeteneklerin, başka bir uygulamada da kullanılabilmesi için yeteneklerini paylasan uygulamanın sağladığı arayüzdür.

Data Base Management System (DBMS) Veri tabanlarını yönetmek, kullanmak, geliştirmek ve bakımını yapmak için kullanılan yazılımlara denir.

- Database'e erişimi düzenler
- Create, Read, Update ve Delete (CRUD) işlemlerini düzenler
- Data güvenliğini sağlar
- Formlar oluşturur ve işler
- Sorgular oluşturur ve iletir
- Raporlar oluşturur ve işletir
- Uygulamayı kontrol eder
- Diğer uygulamalarla (Application) iletişimi sağlar

SQL'de datalar **Table**'larda oluşturulur;

- Başlığa **Headers**,
- Satırlara **Record**,
- Sütunlara **Field** denilir.

Relational Database (İlişkili Tablolar)

- SQL tablolar, dataları ilişkili tablolarda depolar
- Tablolar arasında ilişkiler net olmalıdır
- Tablolar arasında geçiş kolay olmalıdır
- Tablolar ve ilişkilerin bütününe "**Schema**" denir.
- Relational Databases, SQL Databases (Structured Query Language) olarak da adlandırılır.

En çok kullanılan SQL'ler (Relational Databases);

Microsoft SQL Server, MySQL Server, PostgreSQL Server ve Oracle PL/SQL

Non Relational Database – NoSQL Database

- SQL veritabanı verilerle çalışırken yapısal sorgu dili kullanır. Veri yapısını belirlemek için önceden tanımlanmış şemalar gerektirir.
- Tablolar ile çalışmaz, onun yerine doküman dosyalarının içinde depolanır.
- **NoSQL veritabanı** ise verilerle çalışırken Yapılandırılmamış Sorgu Dili kullanır.

SQL komutları şunlardır:

1. Veri Tanımlama Dili (Data Definition Language- DDL)

DDL komutları ile veritabanı ve tabloları oluşturma, değiştirme ve silme işlemleri yapılır. (Bunu Tester'lar yapmaz)

CREATE TABLE tablo_adi

Yeni bir tablo oluşturmak için kullanılır. Alan isimleri yazılırken sona virgöl konulur ve son satır olan işlemimizde virgöl konmadan parantez kapatılır. Ör;

```
CREATE TABLE tabloilceler (  
  ilceNo      mediumint(8) unsigned DEFAULT '0' NOT NULL,  
  ilce        varchar(30) NOT NULL,  
  postakodu   varchar(5),
```

```

ilceTel      char(3),
plakaKodu    char(2) NOT NULL
)

```

ALTER TABLE tablo_adi

Yeni bir sütun eklemek, sütunun tipini veya uzunluğunu değiştirmek/güncellemek vb. yapısal değişiklikler yapılması için kullanılır.

DROP TABLE tablo_adi

Tabloyu içerisindeki verilerle birlikte siler.

// TRUNCATE TABLE tablo_adi

Tablodaki tüm verileri siler, tablo yapısını korur.:

//CREATE VIEW görüş_adi

Görüntü oluşturmak için kullanılır

//DROP VIEW görüş_adi

Görüntüyü siler

//CREATE INDEX indeks_adi

Tablonun (en azından bir) sütun adı üzerinde indeks oluşturmak için kullanılır.

//DROP INDEX indeks_adi

Oluşturulan indeksleri veri tabanından kaldırmak için kullanılır.

2. Veri Sorgulama Dili (Data Query Language- DQL)

DQL içindeki SELECT komutu ile veritabanında yer alan mevcut verilerin bir kısmını veya tamamını, tanımlanan koşullara bağlı olarak alır.

SELECT deyimi

```

SELECT ilçe, postakodu FROM tablolliceler WHERE plakaKodu = '34'
İstanbul'un ilçeleri ile posta kodlarını gösterir

```

3. Veri Kullanma Dili (Data Manipulation Language- DML)

DML komutları ile veritabanlarında bulunan verilere işlem yapılır. DML ile veritabanına yeni kayıt ekleme, mevcut kayıtları güncelleme ve silme işlemleri yapılır.

UPDATE deyimi

```

UPDATE tablolliceler SET postakodu = '06720' WHERE ilçe = 'Bala'
Bala'nın posta kodunu değiştirir/günceller

```

INSERT deyimi

```

INSERT INTO tablolliceler VALUES (, 'Yenişehir', , , '53')
Yeni veriler ekler

```

DELETE deyimi

```

DELETE FROM tablolliceler WHERE plakaKodu = '53'
plakaKodu 53 olan bütün verileri siler

```

4. Veri Kontrol Dili (Data Control Language- DCL)

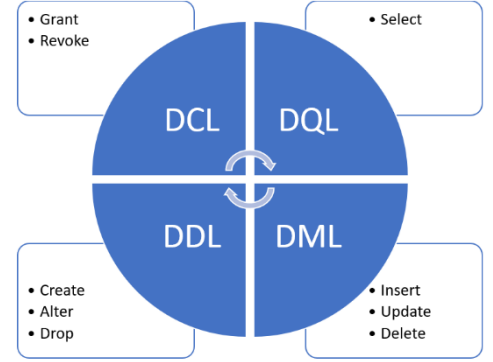
DCL komutları ile kullanıcılara veritabanı ve tablolar için yetki verilir veya geri alınır: (Bunu Tester'lar yapmaz)

GRANT: Bir kullanıcıya yetki vermek için kullanılır.

REVOKE: Bir kullanıcıya verilen yetkiyi geri almak için kullanılır.

Primary Key

- Her bir satır için eşsiz bir veridir. Primary Key tabloyu oluşturan kişi tarafından belirlenir.
- Unique 'dir ama her unique data **Primary Key** değildir.
- Duplication kabul etmez
- Null kabul etmez
- Bir tabloda yalnızca bir tane **Primary Key** olabilir.
- Her tabloda Primary Key olması zorunlu değildir.
- Primary Key her türlü datayı içerebilir (Sayı, String...).
- T.C. kimlik numarası, ISBN, email hesabi gibi gerçek verilere "**Natural Primary Key**" denir.
- Genel olarak kayıt eklenmeden önce üretilen sıra numarası gibi sayısal değerlere "**Surrogate Primary Key**" denir.
- Relational veri tabanlarında (relational database management system) mutlaka **Primary Key** olmalıdır.



Foreign Key

- İki tablo arasında relation (ilişki) oluşturmak için kullanılır.
- Başka bir tablodaki Primary Key ile ilişkilendirilmiş (bağlı) olmalıdır.
- Bir tabloda birden fazla Foreign Key olabilir.
- Foreign Key Null değeri kabul eder.
- Foreign Key olarak tanımlanan field'da tekrarlar (duplication) olabilir.
- Referenced table (bağlanılan tablo, Primary Key'in olduğu Tablo) "parent table" olarak adlandırılır. Foreign Key olan tabloya "child table" denir.
- *** "Parent Table" olmayan bir id'ye sahip datayı "Child Table"a ekleyemezsiniz.
- *** "Child Table" i silmeden "Parent Table" i silemezsiniz. Önce "Child Table" silinir, sonra "Parent Table" silinir.

SQL-2 Data Tipleri

06.10.2020

Composite Key birden fazla field (kolon)'in kombinasyonu ile oluşturulur. Tek başına bir kolon Primary Key olma özelliklerini taşııyorsa, bu özellikleri elde etmek için birden fazla kolon birleştirilerek **Primary Key** oluşturulur.

Primary Key'in Unique Key'den farkları;

1. Bir Tabloda sadece 1 tane olur
2. NULL değeri kabul etmez

Unique Key'in Primary Key'den farkları;

1. Bir tabloda birden fazla olabilir.
2. Sadece 1 tane NULL değeri kabul eder

Primary Key ve Unique Key'in ortak özelliği; Duplication (Çift Kullanım)'a izin vermezler.

Tablolarla arası Related üç şekilde olur;

- 1- One to One Relation
- 2- One to Many Relation
- 3- Many to Many Relation

String, Numeric, Date ve BLOB olmak üzere dört tane SQL Data Types vardır.

A. En çok kullanılan **String Data Types** 4 çeşittir; **char**(size), **nchar**(size), **varchar2**(size) ve **nvarchar2**(size)

char(size); TC-Kimlik, tel no gibi uzunluğu sabit String datalarda kullanılır.

nchar(size); genellikle farklı dillerdeki karakterler gibi Unicode datalarda kullanılır.

varchar2(size); isim gibi uzunluğu sabit olmayan String datalarda kullanılır.

nvarchar2(size); değişken uzunluktaki Stringlerin Unicode değerleri için kullanılır.

1. nchar'in dezavantajı iki kat ı byte kullanır.
2. Uzunluğu biliniyorsa Char kullanacağız (T.C. kimlik No gibi)
3. Uzunluğu sabit değilse varchar2 kullanacağız (isim, şehir vb)

B. Sayılar için Numeric Data Types kullanılır.

number(**p,s**) şeklinde kullanılır.

“**p**recision” (**p**) sayıdaki rakam sayısıdır.

“**s**cale” (**s**) virgülden sonra kaç rakam olduğunu belirler.

Örneğin: 1234,56 ==> Precision: 6, Scale: 2 => number(6,2)

C. Tarihler ve zamanı depolamak için Date Data Types kullanılır.

Saniyenin virgüllü kısmını da alır. Standart “Date Format”, “dd - MMM - yy”. Örneğin ‘13 - Apr – 20’

Tarih formatını “ALTER SESSION SET NLS_DATE_FORMAT = “YYYY-MM-DD” kodu kullanılarak değiştirilebilir. Koddan sonra tarih 2020 – 04 – 13 olur.

D. Resim, video, ses gibi dataları binary formatına çevirerek depolamak için BLOB Data Types kullanılır.

“**B**inary **L**arge **O**bjects” demektir.

Practices;

-- P1 id, name, grade, adres ve last_update oluşan student_table oluşturunuz

```
CREATE TABLE student_table
(
  id char(11),
  name varchar2(50) NOT NULL,
  grade number(5,2),
  adres varchar2(100),
  last_update date,
  CONSTRAINT id_pk PRIMARY KEY (id)
);
```

--student_table'dan sadece name ve grade fieldlari alarak yeni bir tablo olusturun

```
CREATE TABLE student_grade
AS
SELECT name, grade
FROM student_table;
```

-- P2 tedarikci_id, tedarikci_ismi, tedarikci_adres, ve ulasim_tarihi olan tedarikçiler

```
CREATE TABLE tedarikciler
(
  tedarikci_id char(11),
  tedarikci_ismi varchar2(50),
  tedarikci_adres varchar2(100),
  ulasim_tarihi date
);
```

--

```
CREATE TABLE tedarikçi_ziyaret
AS
SELECT tedarikci_ismi, ulasim_tarihi
FROM tedarikçiler;
```

SQL-3 PRIMARY KEY und FOREIGN KEY

13.10.2020

- SQL'de Code yazarken bir field "boş bırakılmasın" diyorsa "NOT NULL",
- SQL'de Code yazarken bir field "tekrarlı değer kabul etmesin/tekrarsız" diyorsa "UNIQUE",
- SQL'de Code yazarken yorum yazılacaksa başına "--" işareti,
- Bir field Primary Key atanacaksa; Data Type'dan sonra "PRIMARY KEY" yazılır. (Ör; id number(9) PRIMARY KEY,)
- Birden fazla değer Primary Key atanacaksa;
CONSTRAINT tablo_ismi_pk **PRIMARY KEY** (field1, field2, id, ...) yazılır.
- Bir "Tabloya Data Ekleme" için;
INSERT INTO tablo_ismi **VALUES** (değer1, değer2,...); yazılır.
- Bir "Tabloda bazı field'lara Data Ekleme" için;
INSERT INTO tablo_ismi (column1, column2) **VALUES** (değer1, değer2); yazılır.
- **INSERT INTO** kodunu kullanarak bir tabloya data eklemek istediğinizde, **CONSTRAINT**'lere (kısıtlama) uymak zorundayız. Örneğin; **NOT NULL** yazan field'a bir değer atamak zorundayız aksi takdirde hata alınır.
- Oluşturulan bir tabloyu ekrana yazdırmak için; **SELECT * FROM** tablo_ismi; yazılır.

--P3 "şehirler" isimli bir Table oluşturun. Tabloda "alan_kodu", "isim", "nufus" field'lari olsun. Isim field'i boş bırakılmasın.

--1.Yöntemi kullanarak "alan_kodu" field'ini "Primary Key" yapın

```
CREATE TABLE sehirler
(
    alan_kodu char(3) PRIMARY KEY,
    isim varchar2(50) NOT NULL,
    nüfus number(7)
);
```

--P4 "öğretmenler" isimli bir Table oluşturun. Tabloda "id", "isim", "brans", "cinsiyet" field'lari olsun.

--Id field'i tekrarlı değer kabul etmesin.

--2.Yöntemi kullanarak "id ve isim" field'lerinin birleşimini "primary key" yapın

```
CREATE TABLE ogretmenler
(
    id char(10) UNIQUE,
    isim varchar2(50) NOT NULL,
    brans varchar2(20),
    cinsiyet varchar2(10),
    CONSTRAINT ogretmenler_pk PRIMARY KEY (id, isim)
);
```

- Bir tabloya Foreign Key atanacaksa;

CONSTRAINT tablo_ismi_fk **FOREIGN KEY** (field3) **REFERENCES** diğer_tablo_ismi (field3) yazılır.

--P5 "tedarikciler" isimli bir tablo oluşturun. Tabloda "tedarikci_id", "tedarikci_isim", "iletişim_isim" field'lari olsun ve
--"tedarikci_id" yi Primary Key yapın.

```

--"urunler" isminde baska bir tablo olusturun "tedarikci_id" ve "urun_id"
field'lari olsun ve
--"tedarikci_id" yi Foreign Key yapin.

CREATE TABLE tedarikciler
(
tedarikci_id char(10) PRIMARY KEY,
tedarikci_isim varchar2(50),
iletisim_isim varchar2(50),
);

CREATE TABLE urunler
(
tedarikci_id char(10),
urun_id char (10),
CONSTRAINT urunler_fk FOREIGN KEY (tedarikci_id) REFERENCES tedarikciler
(tedarikci_id)
);

-- P6 "tedarikciler" isimli bir Tablo olusturun. Icinde "tedarikci_id",
"tedarikci_isim", "iletisim_isim" field'lari olsun.
--"tedarikci_id" ve "tedarikci_isim" fieldlarini birlestirerek Primary Key
olusturun.
--"urunler" isminde baska bir tablo olusturun. Icinde "tedarikci_id" ve "urun_id"
fieldlari olsun.
--"tedarikci_id" ve "urun_id" fieldlarini birlestirerek Foreign Key olusturun
CREATE TABLE tedarikciler01
(
tedarikci_id char(10),
tedarikci_isim varchar2(50),
iletisim_isim varchar2(50),
CONSTRAINT tedarikciler01_pk PRIMARY KEY (tedarikci_id, tedarikci_isim)
);

CREATE TABLE urunler01
(
tedarikci_id char(10),
urun_id varchar2(10),
CONSTRAINT urunler01_fk FOREIGN KEY (tedarikci_id, urun_id) REFERENCES
tedarikciler01 (tedarikci_id, tedarikci_isim)
);

-- P
CREATE TABLE students
(
id number(9),
isim varchar2(50),
derece number(3),
adres varchar2(100),
last_modification date,
CONSTRAINT id_pk PRIMARY KEY(id)
);

(1- Tüm field lere data eklemek için;)
INSERT INTO students VALUES (123456789, 'Ali Can', 85, 'Paris, Louvre Museum', '13-
Oct-2020');

(2- Bazı field lere data eklemek için;)
INSERT INTO students (id, isim) VALUES (456123789, 'Veli Han');
SELECT * FROM students;

```


- Tablodaki Data Nasıl **Update** Edilir (**UPDATE SET**)?

-- **P** Bir "tedarikçiler" tablosu oluşturun. içinde id, isim ve iletisim_isim field'leri olsun. Id ve isim'i beraber Primary Key yapın.

```
CREATE TABLE tedarikciler
(
  id number(10),
  isim varchar2(50),
  iletisim_isim varchar2(50),
  CONSTRAINT tedarikci_pk PRIMARY KEY (id, isim)
);
-- Icine 3 kayıt ekleyin (1, 'ACB', 'Ali Can'), (2, 'RDB', 'Veli Gul'), (3, 'KMN', 'Ayse Gulmez').
INSERT INTO tedarikciler VALUES (1, 'ACB', 'Ali Can');
INSERT INTO tedarikciler VALUES (2, 'RDB', 'Veli Gul');
INSERT INTO tedarikciler VALUES (3, 'KMN', 'Ayse Gulmez');
-- id'si 1 olan tedarikcinin ismini 'KRM' ve iletisim_isim'ini 'Hasan Han' yapın
UPDATE tedarikçiler
SET isim = 'KRM', iletisim_isim = 'Hasan Han'
WHERE id = 1;
-- Ismi RDB olan tedarikcinin iletisim isim'ini Kemal Yasa yapın
UPDATE tedarikçiler
SET iletisim_isim = 'Kemal Yasa'
WHERE isim = 'RDB';
```

-- **P11** a) Urunler tablosundan Ali Can'ın aldığı urunun ismini, tedarikci tablosunda irtibat_isim Merve Temiz olan sirketin ismi ile değiştirin

--b) TV satın alan müşterinin ismini, Apple'ın irtibat_isim'i ile degistirin

```
CREATE TABLE tedarikci
(
  id number(5) PRIMARY KEY,
  isim varchar2(50),
  irtibat_isim varchar2(50)
);
INSERT INTO tedarikci VALUES (100, 'IBM', 'Ali Can');
INSERT INTO tedarikci VALUES (101, 'APPLE', 'Merve Temiz');
INSERT INTO tedarikci VALUES (102, 'SAMSUNG', 'Kemal Can');
INSERT INTO tedarikci VALUES (103, 'LG', 'Ali Can');

CREATE TABLE urunler
(
  tedarikci_id number(5),
  urun_id number(11),
  urun_isim varchar2(50),
  musteri_isim varchar2(50),
  CONSTRAINT urunler_fk FOREIGN KEY (tedarikci_id) REFERENCES tedarikci (id)
);
INSERT INTO urunler VALUES (100, 1001, 'Laptop', 'Suleyman');
INSERT INTO urunler VALUES (101, 1002, 'iPad', 'Fatma');
INSERT INTO urunler VALUES (102, 1003, 'TV', 'Ramazan');
INSERT INTO urunler VALUES (103, 1004, 'Phone', 'Ali Can');
```

```
a) UPDATE urunler
  SET urun_isim = (SELECT isim
                  FROM tedarikçi
                  WHERE irtibat_isim = 'Merve Temiz')
  WHERE musteri_isim = 'Ali Can'
```

```

b) UPDATE urunler
SET musteri_isim = (SELECT irtibat_isim
                     FROM tedarikci
                     WHERE isim = 'APPLE')
WHERE urun_ismi 'TV';

-- P Ogrenciler isminde bir tablo olusturun, icinde id, isim, not_ortalamasi, adres
ve son_degistirme_tarihi fieldleri olsun

CREATE TABLE ogrenciler
(
id char(11),
isim varchar2(50),
not_ortalamasi number(3),
adres varchar2(50),
son_degistirme_tarihi date
);

--123456789, Ali Can', 80, 'Istanbul,bakirkoy', '14-Oct-2020'

INSERT INTO ogrenciler VALUES('123456789', 'Ali Can', 80, 'Istanbul, bakirkoy',
'14-Oct-2020');
INSERT INTO ogrenciler VALUES('123456788', 'Veli Han', 83, 'Ankara, Cankaya', '12-
Oct-2020');

SELECT * FROM ogrenciler;

```

• SQL-4 Slayt Sayfa-8 önemli bir soru

<p>1) Ogrenciler tablosu olusturun. Icinde id,isim,veli_isim ve grade field'lari olsun. Id ve isim fieldlari birlikte Primary Key olsun.</p> <pre> CREATE TABLE ogrenciler (id char(3), isim varchar2(50), veli_isim varchar2(50), yazili_notu number(3), CONSTRAINT ogrenciler_pk PRIMARY KEY (id)); </pre>	<p>2) 3 kisiyi tabloya ekleyin. (123, 'Ali Can', 'Hasan',75), (124, 'Merve Gul', 'Ayse',85), (125, 'Kemal Yasa', 'Hasan',85).</p> <pre> INSERT INTO ogrenciler VALUES(123, 'Ali Can', 'Hasan',75); INSERT INTO ogrenciler VALUES(124, 'Merve Gul', 'Ayse',85); INSERT INTO ogrenciler VALUES(125, 'Kemal Yasa', 'Hasan',85); </pre>
<p>3) notlar tablosu olusturun. ogrenci_id,ders_adi,yazili_notu field'lari olsun, ogrenci_id field'i Foreign Key olsun</p> <pre> CREATE TABLE notlar (ogrenci_id char(3), ders_adi varchar2(30), yazili_notu number(3), CONSTRAINT notlar_fk FOREIGN KEY (ogrenci_id) REFERENCES ogrenciler (id)); </pre>	<p>4) notlar tablosuna 3 kayıt ekleyin ('123','kimya',75), ('124','fizik',65),('125','tarih',90)</p> <pre> INSERT INTO notlar VALUES ('123','kimya',75); INSERT INTO notlar VALUES ('124','fizik',65); INSERT INTO notlar VALUES ('125','tarih',90); </pre>
<p>5) Tum ogrencilerin yazili notlarini notlar tablosundaki ile update edin</p> <pre> UPDATE ogrenciler SET yazili_notu= (SELECT yazili_notu FROM notlar WHERE ogrenciler.id=notlar.ogrenci_id WHERE id>100; </pre>	

SQL-5 TABLODAN DATA SİLME

15.10.2020

```

-- mart_satislar isminde bir tablo olusturun. Icinde urun_id, musteri_isim,
urun_isim, urun_fiyat fieldlari olsun
CREATE TABLE mart_satislar
(
urun_id char(5),
musteri_isim varchar2(30),
urun_isim varchar2(50),
urun_fiyat number(9)
);
INSERT INTO mart_satislar VALUES (10, 'Ali', 'Honda', 75000);
INSERT INTO mart_satislar VALUES (10, 'Ayse', 'Honda', 80000);
INSERT INTO mart_satislar VALUES (20, 'Hasan', 'Toyota', 90000);

```

```

INSERT INTO mart_satislar VALUES (30, 'Veli', 'Ford',100000);
INSERT INTO mart_satislar VALUES (20, 'Ali', 'Toyota',110000);
INSERT INTO mart_satislar VALUES (10, 'Veli', 'Honda',120000);
INSERT INTO mart_satislar VALUES (40, 'Ayse', 'Hyundai',130000);
INSERT INTO mart_satislar VALUES (20, 'Ali', 'Toyota',140000);

SELECT * FROM mart_satislar;

--musteri ismi Hasan olan satisinin urun_isim 'ini Honda yapiniz
UPDATE mart_satislar
SET urun_isim = 'Honda'
WHERE musteri_isim = 'Hasan';

--urun_isim = 'Toyota' olanlari urun_id = '50' yapin
UPDATE mart_satislar
SET urun_id = '50'
WHERE urun_isim = 'Toyota';

-- urun_isim 'i 'Honda' olanlari urun_fiyat larini %10 artirin
UPDATE mart_satislar
SET urun_fiyat = urun_fiyat * 1.1
WHERE urun_isim = 'Honda';

--musteri_isim 'leri Ayse olanlara %10 indirim yapin
UPDATE mart_satislar
SET urun_fiyat = urun_fiyat * 0.9
WHERE musteri_isim = 'Ayse';

```

Tablodan Data Nasıl Silinir (DELETE)?

```

CREATE TABLE ogrenciler
(
id char(3),
isim varchar2(50),
veli_isim varchar2(50),
yazili_notu number(3),
CONSTRAINT ogrenciler_pk PRIMARY KEY (id)
);

INSERT INTO ogrenciler VALUES (123, 'Ali Can', 'Hasan', 75);
INSERT INTO ogrenciler VALUES (124, 'Merve Gul', 'Ayse', 85);
INSERT INTO ogrenciler VALUES (125, 'Kemal Yasa', 'Hasan', 85);

SELECT * FROM ogrenciler;

DELETE ogrenciler;

```

- **DELETE** tablo_ismi yazarsak tablodaki tüm rekord'ları (dataları) siler.
- Tüm kayıtlar silindikten sonra bos bir tablo kalır. Eğer tabloyu görmek isterseniz **“no data found”** yazar, tabloyu göstermez.
- **DELETE** komutu tabloyu silmez, sadece kayıtları siler.

```

--isim 'i 'Ali Can' olan kaydı silin
DELETE ogrenciler
WHERE isim = 'Ali Can';

-- yazili_notu 85 olanlari siliniz
DELETE ogrenciler
WHERE yazili_notu = 85;

--veli_isim Hasan veya Ayse olan kayitlari sil
DELETE ogrenciler

```

```
WHERE veli_isim = 'Hasan' OR veli_isim = 'Ayse';

--veli_isim Ayse veya yazili_notu = 75 olan kayitlari sil
DELETE ogrenciler
WHERE veli_isim = 'Ayse' OR yazili_notu = 75;

--veli_isim Hasan ve yazili_notu = 75 olan kayitlari sil
DELETE ogrenciler
WHERE veli_isim = 'Hasan' AND yazili_notu = 75;

--yazili_notu = 85 olmayan kayitlari sil (!= veya <>)
DELETE ogrenciler
WHERE yazili_notu <> 85;
```

Tablodan Data Nasıl Silinir (TRUNCATE)?

```
TRUNCATE TABLE ogrenciler;
```

- Truncate ve Delete komutlarının ikisi de bir tabloda bulunan kayıtları silmek için kullanılır.
- İki komutta sadece belirtilen tablodaki kayıtları siler.
- En belirgin farkı ise DELETE komutu ile belli bir aralığı silebilirken TRUNCATE komutu ile tablonun tamamı silinmektedir.
- “Truncate” kodu kullanılarak bir tablo silinirse dataların geri getirilme ihtimali olmaz.
- “Truncate” kodu geri getirilmesini (rolling back) istemeyeceğiniz tabloları silmek için kullanılır.
- DELETE FROM ile sildiğimiz kayıtları geri getirebiliriz ama TRUNCATE ile silinen kayıtlar geri getirilemez.

```
TRUNCATE TABLE ogrenciler;
DELETE ogrenciler; yada
DELETE ogrenciler
WHERE veli_isim;

SELECT * FROM ogrenciler;
```

Tablodan Data Nasıl Silinir (DROP)?

```
DROP TABLE ogrenciler;
```

- DROP TABLE tüm tabloyu siler ve RECYCLEBIN 'e gönderir.
- DROP TABLE ile silinen tablolar FLASHBACK TABLE ile geri getirilebilir.

```
FLASHBACK TABLE ogrenciler TO BEFORE DROP;
```

- Bir tabloyu geri getirilmemek üzere silmek istiyorsak iki yöntemle yapabiliriz; **(sf 8)**

```
-- 1) Önce DROP TABLE ile silip PURGE TABLE ile RECYCLEBIN 'den de silinebilir
```

```
DROP TABLE ogrenciler;
PURGE TABLE ogrenciler;
```

```
-- 2) DROP TABLE ve PURGE TABLE komutlarını beraber kullanıp geri getirilmeyecek şekilde silebiliriz
```

```
DROP TABLE mart_satislar PURGE;
FLASHBACK TABLE ogrenciler TO BEFORE DROP;
```

- DROP TABLE ile silinmeyen bir tablo direkt PURGE TABLE yapılamaz.
- ```
-- direkt PURGE TABLE ile silmeye çalışırsanız hata alırsınız.
PURGE TABLE ogrenciler;
```

**UYARI:** Purge kullandığımızda Tabloyu ve dataları geri getirmek mümkün değildir.

**Purge Kullanmanın Amacı:** Hassas bilgileri silmek istediğinizde başka insanların o bilgiye ulaşamayacağından emin olursunuz.

```
SELECT * FROM mart_satislar;
```

```
INSERT INTO ogrenciler VALUES (123, 'Ali Can', 'Hasan', 75);
INSERT INTO ogrenciler VALUES (124, 'Merve Gul', 'Ayse', 85);
INSERT INTO ogrenciler VALUES (125, 'Kemal Yasa', 'Hasan', 85);
```

## SELECT KOMUTU

- **SELECT** komutu bize var olan bir dosyadaki istediğimiz dataları getirir

-- **1) Tüm Dataları çağırma;**

```
SELECT *
FROM ogrenciler;
```

-- **2) Sadece bir field**

```
-- ogrenciler tablosundaki tüm isim'leri yazdırın
SELECT isim
FROM ogrenciler;
```

- Kayıtlar arasında filtreleme yapmak için **WHERE** komutu kullanılır.

--ogrenciler tablosundan yazili notu 85 olanların veli isimlerini getirin

```
SELECT veli_isim
FROM ogrenciler
WHERE yazili_notu = 85;
```


--**3) Tablodan birden fazla field'i çağırma**

```
--veli_isim 'i 'Hasan' olanların isim ve yazili_notu 'larını listeleyen bir sorgu yapınız
SELECT isim, yazili_notu
FROM ogrenciler
WHERE veli_isim = 'Hasan';
```

--**4) Bir kayıta ait birden fazla sütunu listeleme**

```
--id'si 124 olan öğrencinin isim ve yazili_notu
SELECT isim, yazili_notu
FROM ogrenciler
WHERE id = 124;
```

--**WHERE** komutu **AND**, **OR**, **<**, **=** ve **>** gibi mantıksal operatörler ile kullanılabilir.



|     |                                   |
|-----|-----------------------------------|
| =   | ==> Equal to sign                 |
| >   | ==> Greater than sign             |
| <   | ==> Less than sign                |
| >=  | ==> Greater than or equal to sign |
| <=  | ==> Less than or equal to sign    |
| < > | ==> Not Equal to sign             |
| AND | ==> And operator                  |
| OR  | ==> Or operator                   |

## IN CONDITION

**IN** Condition birden fazla mantıksal ifade ile tanımlayabileceğimiz durumları (**Condition**) tek komutla yazabilme imkânı verir.

```
CREATE TABLE musteriler
(
urun_id number(10),
musteri_isim varchar2(50),
urun_isim varchar2(50)
);
```

```
INSERT INTO musteriler VALUES (10, 'Mark', 'Orange');
INSERT INTO musteriler VALUES (10, 'Mark', 'Orange');
INSERT INTO musteriler VALUES (20, 'John', 'Apple');
INSERT INTO musteriler VALUES (30, 'Amy', 'Palm');
INSERT INTO musteriler VALUES (20, 'Mark', 'Apple');
INSERT INTO musteriler VALUES (10, 'Adem', 'Orange');
INSERT INTO musteriler VALUES (40, 'John', 'Apricot');
INSERT INTO musteriler VALUES (20, 'Eddie', 'Apple');
```

```
SELECT * FROM musteriler;
```

--Orange veya Apple alan müşteri isimlerini yazdırın

```
SELECT müşteri_isim
FROM musteriler
WHERE ürün_isim = 'Orange' OR ürün_isim = 'Apple';
```

--Orange veya Apple veya Apricot alan müşteri isimlerini yazdırın

```
SELECT müşteri_isim
FROM musteriler
WHERE ürün_isim = 'Orange' OR ürün_isim = 'Apple' OR ürün_isim = 'Apricot';
```

```
SELECT müşteri_isim
FROM musteriler
WHERE ürün_isim IN ('Orange', 'Apple', 'Apricot');
```

--müşteri\_isim 'Mark' 'John' olanların aldıkları ürün\_isim 'lerini yazdırın

```
SELECT ürün_isim
FROM musteriler
WHERE müşteri_isim IN ('Mark', 'John');
```

-- ürün\_id si 20'den büyük ve ürün\_id si 40'dan küçük olan ürünleri alan

```
müşteri_isim 'leri
SELECT müşteri_isim
FROM musteriler
WHERE ürün_id >20 AND ürün_id<40;
```

-- ürün\_id si 20'ye eşit ve büyük veya ürün\_id si 40'a eşit ve 40'dan küçük olan ürünleri alan müşteri isimleri

```
SELECT müşteri_isim
FROM musteriler
WHERE ürün_id >= 20 AND ürün_id <= 40;
```

## BETWEEN CONDITION

**BETWEEN** Condition iki mantıksal ifade ile tanımlayabileceğimiz durumları tek komutla yazabilme imkânı verir. **BETWEEN** komutunda yazdığımız iki sınır da aralığa dahildir (**INCLUSIVE**)

```
SELECT muster_i_isim
FROM musteriler
WHERE urun_id BETWEEN 20 AND 40;
```

## NOT BETWEEN CONDITION

**NOT BETWEEN** Condition iki mantıksal ifade ile tanımlayabileceğimiz durumları tek komutla yazabilme imkânı verir. **NOT BETWEEN** komutunda yazdığımız 2 sınır da aralığa hariçtir (**EXCLUSIVE**)

```
--urun_id si 20'den küçük veya 30'dan büyük olan urun_isim 'lerini
SELECT urun_isim
FROM musteriler
WHERE urun_id < 20 OR urun_id > 30;
```

```
SELECT urun_isim
FROM musteriler
WHERE urun_id NOT BETWEEN 20 AND 30;
```

"Truncate ve Delete komutlarının ikisi de bir tabloda bulunan kayıtları silmek için kullanılır. İki komutta sadece belirtilen tablodaki kayıtları siler. En belirgin farkı ise DELETE komutu ile belli bir aralığı silebilirken TRUNCATE komutu ile tablonun tamamı silinmektedir."

1. DELETE ile TRUNCATE arasındaki fark nedir?

- A) TRUNCATE tüm kayıtları siler, DELETE istersek tüm kayıtları, istersek belirli kayıtları siler
- B) DELETE ile sildiğimiz dataları ROLLBACK yapabiliriz, TRUNCATE ile silinenler geri getirilemez
- C) DELETE ile WHERE komutunu kullanabiliriz ama TRUNCATE ile kullanamayız

2. DELETE ile DROP arasındaki fark nedir?

DELETE kayıtları siler, DROP ise tabloları.

3. DROP ile DROP PURGE arasındaki fark nedir?

DROP ile sildiğimiz dosyalar RECYCLEBIN 'e gider. PURGE RECYCLEBIN 'deki dosyaları geri getirilmeyecek şekilde siler. DROP PURGE beraber kullanılırsa geri getirilmeyecek şekilde silinir.

## SQL-6 SUBQUERIES

20.10.2020

**EXISTS Condition subquery**'ler ile kullanılır. IN ifadesinin kullanımına benzer olarak, EXISTS ve NOT EXISTS ifadeleri de alt sorgudan getirilen değerlerin içerisinde bir değerın olması veya olmaması durumunda işlem yapılmasını sağlar.

```
--Personel isminde bir tablo oluşturun. İçinde id, isim, şehir, maaş ve şirket
field'leri olsun.
```

```
--Id'yi 2.yöntemle PK yapın
```

```
CREATE TABLE personel
(
id number(9),
isim varchar2(50),
```

```

sehir varchar2(50),
maas number(20),
sirket varchar2(20),
CONSTRAINT personel_pk PRIMARY KEY (id)
);

INSERT INTO personel VALUES(123456789, 'Ali Yilmaz', 'Istanbul', 5500, 'Honda');
INSERT INTO personel VALUES(234567890, 'Veli Sahin', 'Istanbul', 4500, 'Toyota');
INSERT INTO personel VALUES(345678901, 'Mehmet Ozturk', 'Ankara', 3500, 'Honda');
INSERT INTO personel VALUES(456789012, 'Mehmet Ozturk', 'Izmir', 6000, 'Ford');
INSERT INTO personel VALUES(567890123, 'Mehmet Ozturk', 'Ankara', 7000, 'Tofas');
INSERT INTO personel VALUES(456715012, 'Veli Sahin', 'Ankara', 4500, 'Ford');
INSERT INTO personel VALUES(123456710, 'Hatice Sahin', 'Bursa', 4500, 'Honda');

SELECT * FROM personel;

--Personel_bilgi isminde bir tablo olusturun. Icinde id, tel ve cocuk_sayisi
field'leri olsun.
--Id'yi FK yapin ve personel tablosu ile relation kurun

CREATE TABLE personel_bilgi
(
id number(9),
tel char(10) UNIQUE ,
cocuk_sayisi number(2),
CONSTRAINT personel_bilgi_fk FOREIGN KEY (id) REFERENCES personel (id)
);

INSERT INTO personel_bilgi VALUES(123456789, '5302345678', 5);
INSERT INTO personel_bilgi VALUES(234567890, '5422345678', 4);
INSERT INTO personel_bilgi VALUES(345678901, '5354561245', 3);
INSERT INTO personel_bilgi VALUES(456789012, '5411452659', 3);
INSERT INTO personel_bilgi VALUES(567890123, '5551253698', 2);
INSERT INTO personel_bilgi VALUES(456789012, '5524578574', 2);
INSERT INTO personel_bilgi VALUES(123456710, '5537488585', 1);

SELECT * FROM personel_bilgi;

--SORU 1) Personel_bilgi tablosundan 5 cocugu olan kisinin cocuk sayisini 2 yapin
UPDATE personel_bilgi
SET cocuk_sayisi = 2
WHERE cocuk_sayisi = 5;

--SORU 2) Personel tablosundan ucreti 4500 veya 5000 olanlari maaslarini %10
artirin
UPDATE personel
SET maas = maas *1.1
WHERE maas IN (4500, 5000);

--SORU 3) Personel tablosundan maasi 4950 olanlari silin
DELETE personel
WHERE maas = 4950;

Parent-Child relation'i oluřturulan tablolarda child datalar silinmeden parent datalar silinemez,
SQL ORA-02292 hatası verir.

--SORU 4) cocuk sayisi 3 veya 4 olanlari siliniz.
DELETE personel_bilgi
WHERE cocuk_sayisi IN (3,4);

--SORU 5) Honda da calisip maasi 4500 ve üzeri olanlari silin.
DELETE personel
WHERE sirket = 'HONDA' AND maas = 3500;

```



```
--SORU 6) personel_bilgi den datalari geri gelmeyecek sekilde silin.
TRUNCATE TABLE personel_bilgi;

--SORU 7) personel tablosundan maasi 4000 ile 5000 arasinda olanlari silin.
DELETE personel
WHERE maas BETWEEN 4000 AND 5000;

--SORU 8) Personel tablosundan maasi 5000 ile 6000 arasında olmayanlari silin.
DELETE personel
WHERE maas NOT BETWEEN 5000 AND 6000;

--SORU 9) Personel tablosunu geri getirilemeyecek şekilde silin.
```

- Her ne kadar personel\_bilgi tablosundan silmiş olsak da tablo duruyor
- Child tablo boş olarak dururken parent tablodan kayıtları rahatlıkla silebilirsiniz
- Ancak child tablo silinmeden parent tablo silinemez.

DROP TABLE personel PURGE; hata verir.

- Önce personel\_bilgi tablosu silinmeli
- ```
DROP TABLE personel_bilgi;
```
- Sonra personel tablosu silinmeli
- ```
DROP TABLE personel PURGE;
```

## SUBQUERIES

**SUBQUERY** başka bir SORGU (**query**)'nün içinde çalışan SORGU 'dur.

1. **WHERE** den sonra kullanılabilir.

```
--personel tablosu
CREATE TABLE personel
(
id number(9),
isim varchar2(50),
sehir varchar2(50),
maas number(20),
sirket varchar2(20)
);

INSERT INTO personel VALUES(123456789, 'Ali Seker', 'Istanbul', 2500, 'Honda');
INSERT INTO personel VALUES(234567890, 'Ayse Gul', 'Istanbul', 1500, 'Toyota');
INSERT INTO personel VALUES(345678901, 'Veli Yilmaz', 'Ankara', 3000, 'Honda');
INSERT INTO personel VALUES(456789012, 'Veli Yilmaz', 'Izmir', 1000, 'Ford');
INSERT INTO personel VALUES(567890123, 'Veli Yilmaz', 'Ankara', 7000, 'Hyundai');
INSERT INTO personel VALUES(456789012, 'Ayse Gul', 'Ankara', 1500, 'Ford');
INSERT INTO personel VALUES(123456710, 'Fatma Yasa', 'Bursa', 2500, 'Honda');

SELECT * FROM personel;
```

| ID        | ISIM        | SEHIR    | MAAS | SIRKET  |
|-----------|-------------|----------|------|---------|
| 123456789 | Ali Seker   | Istanbul | 2500 | Honda   |
| 234567890 | Ayşe Gul    | Istanbul | 1500 | Toyota  |
| 345678901 | Veli Yilmaz | Ankara   | 3000 | Honda   |
| 456789012 | Veli Yilmaz | Izmir    | 1000 | Ford    |
| 567890123 | Veli Yilmaz | Ankara   | 7000 | Hyundai |
| 456789012 | Ayşe Gul    | Ankara   | 1500 | Ford    |
| 123456710 | Fatma Yasa  | Bursa    | 2500 | Honda   |

```
--sirketler tablosu
CREATE TABLE sirketler
(
sirket_id number(9),
sirket varchar2(20),
personel_sayisi number(20)
);
```

```
INSERT INTO sirketler VALUES(100, 'Honda', 12000);
INSERT INTO sirketler VALUES(101, 'Ford', 18000);
INSERT INTO sirketler VALUES(102, 'Hyundai', 10000);
INSERT INTO sirketler VALUES(103, 'Toyota', 21000);
```

| SIRKET_ID | SIRKET  | PERSONEL_SAYISI |
|-----------|---------|-----------------|
| 100       | Honda   | 12000           |
| 101       | Ford    | 18000           |
| 102       | Hyundai | 10000           |
| 103       | Toyota  | 21000           |

```
SELECT * FROM sirketler;
```

- SUBQUERY en çok **WHERE** ve **SELECT** ile kullanılabilir

--**WHERE** soruları

--**Soru 1**) Personel sayisi 15.000'den cok olan sirketlerin  
--isimlerini ve bu şirkette çalışan personelin isimlerini listeleyin

```
SELECT sirket, isim
FROM personel
WHERE sirket IN ('Ford', 'Toyota');
```

| SIRKET | ISIM        |
|--------|-------------|
| Toyota | Ayse Gul    |
| Ford   | Veli Vilmaz |
| Ford   | Ayse Gul    |

```
--Personel sayisi 15.000'den cok olan sirketler
SELECT sirket
FROM sirketler
WHERE personel_sayisi > 15000;
```

| SIRKET |
|--------|
| Ford   |
| Toyota |

--iki QUERY'i iççice yazıyoruz

```
SELECT sirket, isim
FROM personel
WHERE sirket IN (SELECT sirket
FROM sirketler
WHERE personel_sayisi > 15000);
```

| SIRKET | ISIM        |
|--------|-------------|
| Ford   | Veli Vilmaz |
| Ford   | Ayse Gul    |
| Toyota | Ayse Gul    |

--**Soru 2**) sirket\_id 'si 101'den büyük olan şirketlerin maas larini ve sehir lerini  
listeleyiniz

```
SELECT maas, sehir
FROM personel
WHERE sirket IN (SELECT sirket
FROM sirketler
WHERE sirket_id > 101);
```

| MAAS | SEHIR    |
|------|----------|
| 7000 | Ankara   |
| 1500 | Istanbul |

--**Soru 3**) 'Ankara' daki sirket lerin (**Honda, Hyundai, Ford**) sirket\_id ve  
personel\_sayisi ni listeleyiniz

```
SELECT sirket_id, personel_sayisi
FROM sirketler
WHERE sirket (ortak) IN (SELECT sirket
FROM personel
WHERE sehir='Ankara');
```

| SIRKET_ID | PERSONEL_SAYISI |
|-----------|-----------------|
| 100       | 12000           |
| 102       | 10000           |
| 101       | 18000           |

```
SELECT sirket_id, personel_sayisi
FROM sirketler
WHERE sirket IN ('Honda', 'Hyundai', 'Ford');
```

## SELECT ile SUBQUERY kullanımı

- WHEN satırında yazdığımız QUERY'lerde SELECT satırında field isimleri kullanıyoruz.
- Dolayısıyla eğer SELECT satırında bir SUBQUERY yazacaksak sonucunun tek bir field ismi olması gerekir.
- Ancak SELECT CLAUSE da kullanılan SUBQUERY sadece 1 değer dönmelidir.
- Dolayısıyla SELECT satırında SUBQUERY yazacaksak SUM, COUNT, MIN, MAX ve AVG gibi fonksiyonlar kullanılır.
- Bu fonksiyonlara AGGREGATE FUNCTION (hesaplama fonksiyonları) denir.

| Fonksiyon | Kullanımı           |
|-----------|---------------------|
| SUM       | Toplam              |
| AVG       | Ortalama            |
| MAX       | En büyük değer      |
| MIN       | En küçük değer      |
| COUNT     | Toplam Kayıt Sayısı |

### Aggregate Function (Hesaplama Fonksiyonları)

2. **SELECT'** den sonra kullanılabilir.

```
--SORU 1- Her sirketin ismini, personel_sayisi ni ve personelin ortalama maasini
listelezen bir QUERY yazin.
```

```
SELECT sirket, personel_sayisi, (
 SELECT AVG (maas)
 FROM personel
 WHERE sirketler.sirket = personel.sirket
) ortalama_maas
FROM sirketler;
```

| SIRKET  | PERSONEL_SAYISI | ORTALAMA_MAAS                     |
|---------|-----------------|-----------------------------------|
| Honda   | 12000           | 2666.6666666666666666666666666667 |
| Ford    | 18000           | 1250                              |
| Hyundai | 10000           | 7000                              |
| Toyota  | 21000           | 1500                              |

\*tablodaki sütuna isim vermek için parantezin dışına verilmek istenen isim (**ortalama maas**) yazılır.

--SORU 2- Her şirketin ismini ve personelin aldığı max. maaşı listeleyen bir QUERY yazın.

```
SELECT sirket, (
 SELECT MAX (maas)
 FROM personel
 WHERE sirketler.sirket = personel.sirket
) max_maas
FROM sirketler;
```

| SIRKET  | MAX_MAAS |
|---------|----------|
| Honda   | 3000     |
| Ford    | 1500     |
| Hyundai | 7000     |
| Toyota  | 1500     |

--SORU 3- Her sirketin id'sini, ismini ve toplam kac sehirde bulunduğunu listeleyen bir QUERY yaziniz.

```
SELECT sirket_id, sirket, (
 SELECT COUNT (sehir)
 FROM personel
 WHERE sirketler.sirket = personel.sirket
) bulundugu_sehir_sayisi
FROM sirketler;
```

| SIRKET_ID | SIRKET  | BULUNDUGU_SEHIR_SAYISI |
|-----------|---------|------------------------|
| 100       | Honda   | 3                      |
| 101       | Ford    | 2                      |
| 102       | Hyundai | 1                      |
| 103       | Toyota  | 1                      |

--SORU 3,5- id si 101'den büyük olan şirketlerin id'sini, ismini ve toplam kaç şehirde bulunduğunu listeleyen bir QUERY yazınız.

```
SELECT sirket_id, sirket, (
 SELECT COUNT (sehir)
 FROM personel
 WHERE sirketler.sirket = personel.sirket
) bulunduгу_sehir_sayisi
FROM sirketler
WHERE sirket_id>101;
```

| SIRKET_ID | SIRKET  | BULUNDUGU_SEHIR_SAYISI |
|-----------|---------|------------------------|
| 102       | Hyundai | 1                      |
| 103       | Toyota  | 1                      |

--SORU 4- Her şirketin ismini, personel sayısını ve personelin aldığı max. ve min. maaşı listeleyen bir QUERY yazın.

```
SELECT sirket, personel_sayisi, (
 SELECT MAX (maas)
 FROM personel
 WHERE sirketler.sirket = personel.sirket
) max_maas,
(
 SELECT MIN (maas)
 FROM personel
 WHERE sirketler.sirket = personel.sirket
) min_maas
FROM sirketler;
```

| SIRKET  | PERSONEL_SAYISI | MAX_MAAS | MIN_MAAS |
|---------|-----------------|----------|----------|
| Honda   | 12000           | 3000     | 2500     |
| Ford    | 18000           | 1500     | 1000     |
| Hyundai | 10000           | 7000     | 7000     |
| Toyota  | 21000           | 1500     | 1500     |

--SORU 5- Her şirketin ismini ve personel sayısını ve işçilere ödediği toplam maaşı listeleyen bir QUERY yazın.

```
SELECT sirket, personel_sayisi, (
 SELECT SUM (maas)
 FROM personel
 WHERE sirketler.sirket = personel.sirket
) toplam_maas
FROM sirketler;
```

| SIRKET  | PERSONEL_SAYISI | TOPLAM_MAAS |
|---------|-----------------|-------------|
| Honda   | 12000           | 8000        |
| Ford    | 18000           | 2500        |
| Hyundai | 10000           | 7000        |
| Toyota  | 21000           | 1500        |

## SQL-7 EXISTS, IS NULL, ORDER BY, GROUP BY

21.10.2020

```
--mart_satislar tablosu
CREATE TABLE mart_satislar
(
 urun_id number(10),
 musteri_isim varchar2(50),
 urun_isim varchar2(50)
);
```

```
INSERT INTO mart_satislar VALUES (10, 'Mark', 'Honda');
INSERT INTO mart_satislar VALUES (10, 'Mark', 'Honda');
INSERT INTO mart_satislar VALUES (20, 'John', 'Toyota');
INSERT INTO mart_satislar VALUES (30, 'Amy', 'Ford');
INSERT INTO mart_satislar VALUES (20, 'Mark', 'Toyota');
INSERT INTO mart_satislar VALUES (10, 'Adem', 'Honda');
INSERT INTO mart_satislar VALUES (40, 'John', 'Hyundai');
INSERT INTO mart_satislar VALUES (20, 'Eddie', 'Toyota');
```

```
SELECT * FROM mart_satislar;
```

| URUN_ID | MUSTERI_ISIM | URUN_ISIM |
|---------|--------------|-----------|
| 10      | Mark         | Honda     |
| 10      | Mark         | Honda     |
| 20      | John         | Toyota    |
| 30      | Amy          | Ford      |
| 20      | Mark         | Toyota    |
| 10      | Adem         | Honda     |
| 40      | John         | Hyundai   |
| 20      | Eddie        | Toyota    |

```
--nisan_satislar tablosu
CREATE TABLE nisan_satislar
(
urun_id number(10),
musteri_isim varchar2(50),
urun_isim varchar2(50)
);
```

```
INSERT INTO nisan_satislar VALUES (10, 'Hasan', 'Honda');
INSERT INTO nisan_satislar VALUES (10, 'Kemal', 'Honda');
INSERT INTO nisan_satislar VALUES (20, 'Ayse', 'Toyota');
INSERT INTO nisan_satislar VALUES (50, 'Yasar', 'Volvo');
INSERT INTO nisan_satislar VALUES (20, 'Mine', 'Toyota');
```

```
SELECT * FROM nisan_satislar;
```

| URUN_ID | MUSTERI_ISIM | URUN_ISIM |
|---------|--------------|-----------|
| 10      | Hasan        | Honda     |
| 10      | Kemal        | Honda     |
| 20      | Ayse         | Toyota    |
| 50      | Yasar        | Volvo     |
| 20      | Mine         | Toyota    |

## EXISTS (var olmak) Condition

**EXISTS** Condition Subquery'ler ile kullanılır. **IN** ifadesinin kullanımına benzer olarak, EXISTS ve NOT EXISTS ifadeleri de alt sorgudan getirilen değerlerin içerisinde bir değerin olması veya olmaması durumunda işlem yapılmasını sağlar.

- EXISTS, bir alt sorgudaki herhangi bir kaydın varlığını test eder.
- EXISTS, bir alt sorgu veya daha fazla kayıt döndürürse doğru döndürür.
- EXISTS koşulu, genellikle ilişkili alt sorgularla kullanılır.

```
--Her iki ayda da aynı id ile satılan ürünlerin ürün_id'lerini ve
--ürünleri mart ayında alanların isimlerini getiren bir query yazınız.
SELECT ürün_id, müşteri_isim
FROM mart_satislar
WHERE ürün_isim IN ('Honda', 'Toyota');
Veya
SELECT ürün_id, müşteri_isim
FROM mart_satislar
WHERE ürün_isim IN (SELECT ürün_isim
 FROM nisan_satislar
 WHERE mart_satislar.ürün_isim = nisan_satislar.ürün_isim);
Veya
SELECT ürün_id, müşteri_isim
FROM mart_satislar
WHERE EXISTS (SELECT ürün_isim
 FROM nisan_satislar
 WHERE mart_satislar.ürün_isim=nisan_satislar.ürün_isim);
veya
SELECT ürün_id, müşteri_isim
FROM mart_satislar
WHERE EXISTS (SELECT ürün_id
 FROM nisan_satislar
 WHERE mart_satislar.ürün_id = nisan_satislar.ürün_id);
```

| URUN_ID | MUSTERI_ISIM |
|---------|--------------|
| 10      | Mark         |
| 10      | Mark         |
| 20      | John         |
| 20      | Mark         |
| 10      | Adem         |
| 20      | Eddie        |

```
--Her iki ayda da satılan aynı ürünlerin ürün_isim lerini ve
--bu ürünleri nisan ayında alanların müşteri_isim lerini getiren bir query yazınız.
SELECT ürün_isim, müşteri_isim
```

```
FROM nisan_satislar
WHERE EXISTS (SELECT urun_isim
 FROM mart_satislar
 WHERE mart_satislar.urun_isim=nisan_satislar.urun_isim);
```

| URUN_ISIM | MUSTERI_ISIM |
|-----------|--------------|
| Honda     | Hasan        |
| Honda     | Kemal        |
| Toyota    | Ayşe         |
| Toyota    | Mine         |

```
-- Nisan ayında satılan ancak her iki ayda ortak olarak satılmamış olan urun_isim
'lerini
```

```
-- musteri_isim isimlerini ve
-- urun_id 'sini listeleyen QUERY yazınız
```

```
SELECT urun_isim, musteri_isim, urun_id
FROM nisan_satislar
```

```
WHERE NOT EXISTS (SELECT urun_isim
 FROM mart_satislar
 WHERE mart_satislar.urun_isim = nisan_satislar.urun_isim);
```

| MUSTERI_ISIM | URUN_ISIM | URUN_ID |
|--------------|-----------|---------|
| Yasar        | Volvo     | 50      |

## IS NULL CONDITION

- Arama yapılan field’da NULL değeri (boş) almış kayıtları (dataları) getirir.

```
CREATE TABLE insanlar
```

```
(
 ssn char(9),
 isim varchar2(50),
 adres varchar2(50)
);
```

```
INSERT INTO insanlar VALUES(123456789, 'Ali Can', 'Istanbul');
INSERT INTO insanlar VALUES(234567890, 'Veli Cem', 'Ankara');
INSERT INTO insanlar VALUES(345678901, 'Mine Bulut', 'Izmir');
INSERT INTO insanlar (ssn, adres) VALUES(456789012, 'Bursa');
INSERT INTO insanlar (ssn, adres) VALUES(567890123, 'Denizli');
```

```
SELECT * FROM insanlar;
```

| SSN       | ISIM       | ADRES    |
|-----------|------------|----------|
| 123456789 | Ali Can    | Istanbul |
| 234567890 | Veli Cem   | Ankara   |
| 345678901 | Mine Bulut | Izmir    |
| 456789012 | -          | Bursa    |
| 567890123 | -          | Denizli  |

```
--isim field'i boş olan kayıtları getiren Query yazınız
```

```
SELECT *
FROM insanlar
WHERE isim IS NULL;
```

| SSN       | ISIM | ADRES   |
|-----------|------|---------|
| 456789012 | -    | Bursa   |
| 567890123 | -    | Denizli |

**\* Boş olan kayıt** yoksa “no data found” alırız.

```
--isim field'i bos olan kayıtlara isim olarak "Isim girilmemistir" yazdırın
```

```
UPDATE insanlar
SET isim = 'Isim girilmemistir'
WHERE isim IS NULL;
```

| SSN       | ISIM               | ADRES    |
|-----------|--------------------|----------|
| 123456789 | Ali Can            | Istanbul |
| 234567890 | Veli Cem           | Ankara   |
| 345678901 | Mine Bulut         | Izmir    |
| 456789012 | Isim girilmemistir | Bursa    |
| 567890123 | Isim girilmemistir | Denizli  |

```
--adres istanbul ve ankara olanlari ssn'lerini siliniz
UPDATE insanlar
SET ssn = NULL
WHERE adres IN ('Ankara', 'Istanbul');
```

| SSN       | ISIM               | ADRES    |
|-----------|--------------------|----------|
| -         | Ali Can            | Istanbul |
| -         | Veli Cem           | Ankara   |
| 345678901 | Mine Bulut         | Izmir    |
| 456789012 | Isim girilmemistir | Bursa    |
| 567890123 | Isim girilmemistir | Denizli  |

```
--ssn field'i bos olmayanlari listeleyen Query yaziniz
SELECT *
FROM insanlar
WHERE ssn IS NOT NULL;
```

| SSN       | ISIM               | ADRES   |
|-----------|--------------------|---------|
| 345678901 | Mine Bulut         | Izmir   |
| 456789012 | Isim girilmemistir | Bursa   |
| 567890123 | Isim girilmemistir | Denizli |

```
-- insanlar tablosunu siliniz
DROP TABLE insanlar;
=> ORA-00942: table or view does not exist
```

-- yeni insanlar tablosu

```
CREATE TABLE insanlar
(
ssn char(9),
isim varchar2(50),
soyisim varchar2(50),
adres varchar2(50)
);
```

```
INSERT INTO insanlar VALUES(123456789, 'Ali','Can', 'Istanbul');
INSERT INTO insanlar VALUES(234567890, 'Veli','Cem', 'Ankara');
INSERT INTO insanlar VALUES(345678901, 'Mine','Bulut', 'Ankara');
INSERT INTO insanlar VALUES(256789012, 'Mahmut','Bulut', 'Istanbul');
INSERT INTO insanlar VALUES(344678901, 'Mine','Yasa', 'Ankara');
INSERT INTO insanlar VALUES(256789562, 'Veli','Yilmaz', 'Istanbul');
SELECT * FROM insanlar;
```

| SSN       | ISIM   | SOYISIM | ADRES    |
|-----------|--------|---------|----------|
| 123456789 | Ali    | Can     | Istanbul |
| 234567890 | Veli   | Cem     | Ankara   |
| 345678901 | Mine   | Bulut   | Ankara   |
| 256789012 | Mahmut | Bulut   | Istanbul |
| 344678901 | Mine   | Yasa    | Ankara   |
| 256789562 | Veli   | Yilmaz  | Istanbul |

## ORDER BY CLAUSE (komutu)

- **ORDER BY** komutu belli bir field'a göre NATURAL ORDER olarak sıralama yapmak için kullanılır.
- **ORDER BY** komutu sadece **SELECT** komutu ile kullanılır.

```
SELECT *
FROM insanlar
ORDER BY adres;
```

| SSN       | ISIM   | SOYISIM | ADRES    |
|-----------|--------|---------|----------|
| 345678901 | Mine   | Bulut   | Ankara   |
| 344678901 | Mine   | Yasa    | Ankara   |
| 234567890 | Veli   | Cem     | Ankara   |
| 256789562 | Veli   | Yilmaz  | Istanbul |
| 256789012 | Mahmut | Bulut   | Istanbul |
| 123456789 | Ali    | Can     | Istanbul |

```
--isim 'i Mine olanlari soyisim olarak siralayan Query yaziniz
SELECT *
FROM insanlar
WHERE isim = 'Mine'
ORDER BY soyisim;
```

| SSN       | ISIM | SOYISIM | ADRES  |
|-----------|------|---------|--------|
| 345678901 | Mine | Bulut   | Ankara |
| 344678901 | Mine | Yasa    | Ankara |

```
--isim 'i Mine olanlari ssn olarak siralayan Query yaziniz
SELECT *
FROM insanlar
WHERE isim = 'Mine'
ORDER BY ssn;
```

| SSN       | ISIM | SOYISIM | ADRES  |
|-----------|------|---------|--------|
| 344678901 | Mine | Yasa    | Ankara |
| 345678901 | Mine | Bulut   | Ankara |

```
-- soyisim i 'Bulut' olanlari isim olarak siralayan Query yaziniz
SELECT *
FROM insanlar
WHERE soyisim = 'Bulut'
ORDER BY isim;
Yada
```

- **ORDER BY** komutunda özel olarak sıralama yapacağımız field ismi (isim) yerine field numarası (2) da yazabiliriz.

```
SELECT *
FROM insanlar
WHERE soyisim = 'Bulut'
ORDER BY 2;
```

| SSN       | ISIM   | SOYISIM | ADRES    |
|-----------|--------|---------|----------|
| 256789012 | Mahmut | Bulut   | Istanbul |
| 345678901 | Mine   | Bulut   | Ankara   |

```
--'Mine' 'Bulut' adini 'mine' ve soyisim ini 'bulut' yapiniz
UPDATE insanlar
SET isim = 'mine', soyisim = 'bulut'
WHERE isim = 'Mine' AND soyisim = 'Bulut';
```

| SSN       | ISIM        | SOYISIM | ADRES    |
|-----------|-------------|---------|----------|
| 123456789 | Ali         | Can     | Istanbul |
| 234567890 | Veli        | Cem     | Ankara   |
| 345678901 | <u>mine</u> | bulut   | Ankara   |
| 256789012 | Mahmut      | Bulut   | Istanbul |
| 344678901 | <u>Mine</u> | Yasa    | Ankara   |
| 256789562 | Veli        | Yilmaz  | Istanbul |

```
--tüm tabloyu isim e göre sirali yazdirin
SELECT *
FROM insanlar
ORDER BY isim;
```

| SSN       | ISIM        | SOYISIM | ADRES    |
|-----------|-------------|---------|----------|
| 123456789 | Ali         | Can     | Istanbul |
| 256789012 | Mahmut      | Bulut   | Istanbul |
| 344678901 | Mine        | Yasa    | Ankara   |
| 234567890 | Veli        | Cem     | Ankara   |
| 256789562 | Veli        | Yilmaz  | Istanbul |
| 345678901 | <u>mine</u> | bulut   | Ankara   |



## ORDER BY field\_name ASC or DESC Clause

- İsimleri natural order 'a göre sıralamak için sorgunun sonuna ORDER BY (field name) yazmamız yeterli.
- İsimleri ters sıralama ile yazdırmak isterseniz field ismi sonuna **DESC** yazıyoruz.

A>>>> 65      a>>>>97

ascending (from min to max)

descending (from max to min)

```
--İnsanlar tablosundaki tum kayitlari SSN numarasi büyükten küçüğe olarak sıralayin
SELECT *
FROM insanlar
ORDER BY ssn DESC;
```

| SSN       | ISIM   | SOYISIM | ADRES    |
|-----------|--------|---------|----------|
| 345678901 | Mine   | Bulut   | Ankara   |
| 344678901 | Mine   | Yasa    | Ankara   |
| 256789562 | Veli   | Yilmaz  | Istanbul |
| 256789012 | Mahmut | Bulut   | Istanbul |
| 234567890 | Veli   | Cem     | Ankara   |
| 123456789 | Ali    | Can     | Istanbul |

- Eğer birden fazla field sıralanacaksa **DESC** istenilmeyen field'in sonuna **ASC** yazılır.

```
--İnsanlar tablosundaki tum kayitlari isim ler natural sirali, soyisim leri ters
sirali olarak listeleyin
```

```
SELECT *
FROM insanlar
ORDER BY isim ASC, soyisim DESC;
```

| SSN       | ISIM   | SOYISIM | ADRES    |
|-----------|--------|---------|----------|
| 123456789 | Ali    | Can     | Istanbul |
| 256789012 | Mahmut | Bulut   | Istanbul |
| 344678901 | Mine   | Yasa    | Ankara   |
| 345678901 | Mine   | Bulut   | Ankara   |
| 256789562 | Veli   | Yilmaz  | Istanbul |
| 234567890 | Veli   | Cem     | Ankara   |

## ALIASES

- Aliases** kodu ile tablo yazdırırken, field isimleri sadece o çıktı için değiştirilebilir.
- AS** olarak field\_name ile kod arasına yazılır. **SELECT** ile kullanılır.

```
CREATE TABLE employees
(
employee_id char(9),
employee_name varchar2(50),
employee_birth_city varchar2(50)
);
```

```
INSERT INTO employees VALUES (123456789, 'Ali Can', 'Istanbul');
INSERT INTO employees VALUES (234567890, 'Veli Cem', 'Ankara');
INSERT INTO employees VALUES (345678901, 'Mine Bulut', 'Izmir');
```

```
SELECT * FROM employees;
```

| EMPLOYEE_ID | EMPLOYEE_NAME | EMPLOYEE_BIRTH_CITY |
|-------------|---------------|---------------------|
| 123456789   | Ali Can       | Istanbul            |
| 234567890   | Veli Cem      | Ankara              |
| 345678901   | Mine Bulut    | Izmir               |

- Field isimlerini sadeleştirip gösterme

```
SELECT employee_id AS id, employee_name AS isim, employee_birth_city AS dogum_yeri
FROM employees;
```

| <u>ID</u> | <u>ISIM</u> | <u>DOGUM_YERI</u> |
|-----------|-------------|-------------------|
| 123456789 | Ali Can     | Istanbul          |
| 234567890 | Veli Cem    | Ankara            |
| 345678901 | Mine Bulut  | Izmir             |

- İki field 'daki dataları birleştirip tek field 'da raporlamak istiyorsak
- İki field adının arasına || işareti konulur.
- Sonuna da **AS** diyerek isim verilir.

```
-- id sutunu olsun, bir de isim_dogum_yeri
```

```
SELECT employee_id AS id, employee_name || employee_birth_city AS isim_dogum_yeri
FROM employees;
```

| <u>ID</u> | <u>ISIM_DOGUM_YERI</u> |
|-----------|------------------------|
| 123456789 | Ali CanIstanbul        |
| 234567890 | Veli CemAnkara         |
| 345678901 | Mine BulutIzmir        |

## GRUP BY Clause

**GROUP BY** komutu sonuçları bir veya daha fazla sütuna göre gruplamak için **SELECT** komutuyla birlikte kullanılır.

```
CREATE TABLE manav
```

```
(
 isim varchar2(50),
 Urun_adi varchar2(50),
 Urun_miktar number(9)
);
```

```
INSERT INTO manav VALUES ('Ali', 'Elma', 5);
INSERT INTO manav VALUES ('Ayse', 'Armut', 3);
INSERT INTO manav VALUES ('Veli', 'Elma', 2);
INSERT INTO manav VALUES ('Hasan', 'Uzum', 4);
INSERT INTO manav VALUES ('Ali', 'Armut', 2);
INSERT INTO manav VALUES ('Ayse', 'Elma', 3);
INSERT INTO manav VALUES ('Veli', 'Uzum', 5);
INSERT INTO manav VALUES ('Ali', 'Armut', 2);
INSERT INTO manav VALUES ('Veli', 'Elma', 3);
INSERT INTO manav VALUES ('Ayse', 'Uzum', 2);
```

```
SELECT * FROM manav;
```

| ISIM  | URUN_ADI | URUN_MIKTAR |
|-------|----------|-------------|
| Ali   | Elma     | 5           |
| Ayse  | Armut    | 3           |
| Veli  | Elma     | 2           |
| Hasan | Uzum     | 4           |
| Ali   | Armut    | 2           |
| Ayse  | Elma     | 3           |
| Veli  | Uzum     | 5           |
| Ali   | Armut    | 2           |
| Veli  | Elma     | 3           |
| Ayse  | Uzum     | 2           |

```
--1) isim 'e göre alınan toplam Urun_miktar ini bulunuz
SELECT isim, SUM (Urun_miktar) AS alınan_toplam_meyve
FROM manav
GROUP BY isim;
```

| ISIM  | ALINAN_TOPLAM_MEYVE |
|-------|---------------------|
| Veli  | 10                  |
| Ayşe  | 8                   |
| Ali   | 9                   |
| Hasan | 4                   |

```
--2) Urun_adi na göre urunu alan toplam kisi sayisini yazdiriniz.
SELECT Urun_adi, COUNT (isim) AS toplam_kisi_sayisi
FROM manav
GROUP BY Urun_adi;
```

| URUN_ADI | TOPLAM_KISI_SAYISI |
|----------|--------------------|
| Elma     | 4                  |
| Uzum     | 3                  |
| Armut    | 3                  |

```
--3) Alınan kilo miktarina göre musteri sayisini yazdiriniz.
SELECT Urun_miktar, COUNT (isim) AS urun_alan_musteri_sayisi
FROM manav
GROUP BY Urun_miktar;
```

| URUN_MIKTAR | URUN_ALAN_MUSTERI_SAYISI |
|-------------|--------------------------|
| 2           | 4                        |
| 5           | 2                        |
| 4           | 1                        |
| 3           | 3                        |

## SQL-8 HAVING, UNION, UNION ALL, INTERSECT, MINUS 22.10.2020

```
CREATE TABLE personel
(
id number(9),
isim varchar2(50),
sehir varchar2(50),
maas number(20),
sirket varchar2(20)
);
INSERT INTO personel VALUES(123456789, 'Ali Yilmaz', 'Istanbul', 5500, 'Honda');
INSERT INTO personel VALUES(234567890, 'Veli Sahin', 'Istanbul', 4500, 'Toyota');
INSERT INTO personel VALUES(345678901, 'Mehmet Ozturk', 'Ankara', 3500, 'Honda');
INSERT INTO personel VALUES(456789012, 'Mehmet Ozturk', 'Izmir', 6000, 'Ford');
INSERT INTO personel VALUES(567890123, 'Mehmet Ozturk', 'Ankara', 7000, 'Tofas');
INSERT INTO personel VALUES(456789012, 'Veli Sahin', 'Ankara', 4500, 'Ford');
INSERT INTO personel VALUES(123456710, 'Hatice Sahin', 'Bursa', 4500, 'Honda');
```

| ID        | ISIM          | SEHIR    | MAAS | SIRKET |
|-----------|---------------|----------|------|--------|
| 123456789 | Ali Yilmaz    | Istanbul | 5500 | Honda  |
| 234567890 | Veli Sahin    | Istanbul | 4500 | Toyota |
| 345678901 | Mehmet Ozturk | Ankara   | 3500 | Honda  |
| 456789012 | Mehmet Ozturk | Izmir    | 6000 | Ford   |
| 567890123 | Mehmet Ozturk | Ankara   | 7000 | Tofas  |
| 456789012 | Veli Sahin    | Ankara   | 4500 | Ford   |
| 123456710 | Hatice Sahin  | Bursa    | 4500 | Honda  |

```
-- 1) isim e göre toplam maas lari bulun
SELECT isim, SUM (maas) AS toplam_maas
FROM personel
GROUP BY isim;
```

| ISIM          | TOPLAM_MAAS |
|---------------|-------------|
| Hatice Sahin  | 4500        |
| Veli Sahin    | 9000        |
| Ali Yilmaz    | 5500        |
| Mehmet Ozturk | 16500       |

```
-- 2) şehir e göre toplam personel sayisini bulun
SELECT şehir, COUNT (isim) AS toplam_personel
FROM personel
GROUP BY şehir;
```

| ŞEHİR    | TOPLAM_PERSONEL |
|----------|-----------------|
| Izmir    | 1               |
| Bursa    | 1               |
| Istanbul | 2               |
| Ankara   | 3               |

```
-- 3) şirket lere göre maası 5000 liradan fazla olan personel sayisini bulun
SELECT şirket, COUNT (isim) AS beyaz_yakali_personel
FROM personel
WHERE maas > 5000
GROUP BY şirket;
```

| SİRKET | BEYAZ_YAKALI_PERSONEL |
|--------|-----------------------|
| Honda  | 1                     |
| Ford   | 1                     |
| Tofas  | 1                     |

```
-- 4) Her şirket için Min ve Max maas i bulun
SELECT şirket, MAX (maas) max_maas, MIN (maas) min_maas
FROM personel
GROUP BY şirket;
```

| SİRKET | MAX_MAAS | MIN_MAAS |
|--------|----------|----------|
| Honda  | 5500     | 3500     |
| Ford   | 6000     | 4500     |
| Toyota | 4500     | 4500     |
| Tofas  | 7000     | 7000     |

## HAVING CLAUSE

- HAVING, AGGREGATE Function'lar ile birlikte kullanılan FİLTRELEME komutudur.

```
-- 1a) Her şirket in MIN maas larini göster
SELECT şirket, MIN (maas) AS en_az_maas
FROM personel
GROUP BY şirket;
```

| SİRKET | MIN(MAAS) |
|--------|-----------|
| Honda  | 3500      |
| Ford   | 4500      |
| Toyota | 4500      |
| Tofas  | 7000      |

```
-- 1) Her şirket in MIN maas larini eger 4000'den büyükse göster
SELECT şirket, MIN (maas)
FROM personel
GROUP BY şirket
HAVING MIN (maas) > 4000;
```

| SİRKET | MIN(MAAS) |
|--------|-----------|
| Ford   | 4500      |
| Toyota | 4500      |
| Tofas  | 7000      |

- AGGREGATE function larin field larindan süzme yaparsak **WHERE** yerine **HAVING** kullanılır

```
-- 1b) Maasi 4000'den büyükten olanlar icerisinden en düşük maas i bulunuz
SELECT sirket, MIN (maas)
FROM personel
WHERE maas > 4000
GROUP BY sirket;
```

| SIRKET | MIN(MAAS) |
|--------|-----------|
| Honda  | 4500      |
| Ford   | 4500      |
| Toyota | 4500      |
| Tofas  | 7000      |

```
-- 2) Toplam geliri 10000 liradan fazla olan isimleri gösteren sorgu yaziniz
SELECT isim, SUM (maas) AS toplam_maas
FROM personel
GROUP BY isim
HAVING SUM (maas) > 10000;
```

| ISIM          | TOPLAM_MAAS |
|---------------|-------------|
| Mehmet Ozturk | 16500       |

```
-- 3) Eger bir sehir de calisan personel sayisi 1'den coksa sehir ismini ve
personel sayisini veren sorgu yaziniz
SELECT sehir, COUNT (isim) AS toplam_personel_sayisi
FROM personel
GROUP BY sehir
HAVING count (isim) > 1;
```

| SEHIR    | TOPLAM_PERSONEL_SAYISI |
|----------|------------------------|
| Istanbul | 2                      |
| Ankara   | 3                      |

```
-- 4) Eger bir sehirde alinan MAX maas 5000'den dusukse sehir ismini ve MAX maasi
veren sorgu yaziniz
SELECT sehir, MAX (maas) AS max_maas
FROM personel
GROUP BY sehir
HAVING MAX (maas) < 5000;
```

| SEHIR | MAX_MAAS |
|-------|----------|
| Bursa | 4500     |

## UNION OPERATOR

- UNION Fonksiyonu iki farklı sorgulamanın sonuçlarını aynı tabloda birleştirir.
- Seçilen Field SAYISI ve DATA TYPE'i aynı olmalıdır.
- UNION komutu küme gibi calisir. Yani her iki sorgudan ortak sonuçlar varsa 1 kere yazar.
- UNION ile birleştirme yaparken her iki sorgudan gelen sütun sayısı ve sütunların data tipleri aynı olmalıdır.

```
-- maas i 4500 den cok olanlari isimlerini ve maaslarini yazdirin
SELECT sirket, COUNT (isim) AS beyaz_yakali_personel
FROM personel
WHERE maas > 5000
GROUP BY sirket;
```

| SIRKET | BEYAZ_YAKALI_PERSONEL |
|--------|-----------------------|
| Honda  | 1                     |
| Ford   | 1                     |
| Tofas  | 1                     |

```
--maas 5000 den az ise sehir adi ve maas yazdirin
SELECT sehir, maas
FROM personel
WHERE maas < 5000;
```

| SEHIR    | MAAS |
|----------|------|
| Istanbul | 4500 |
| Ankara   | 3500 |
| Ankara   | 4500 |
| Bursa    | 4500 |

-- alınan maas 5000'den az olan şehirler veya maasi 4500'den çok olan personeli ve alınan maasi yazdırın

```
SELECT isim AS isim_veya_sehir, maas
FROM personel
WHERE maas > 4500
UNION
SELECT sehir, maas
FROM personel
WHERE maas < 5000;
```

| ISIM_VEYA_SEHIR | MAAS |
|-----------------|------|
| Ali Yılmaz      | 5500 |
| Ankara          | 3500 |
| Ankara          | 4500 |
| Bursa           | 4500 |
| İstanbul        | 4500 |
| Mehmet Ozturk   | 6000 |
| Mehmet Ozturk   | 7000 |

--Ankarada çalışanları veya maasi 4000'den fazla olanların isim ve maaşlarını yazdırın

```
SELECT isim, maas
FROM personel
WHERE sehir = 'Ankara'
UNION
SELECT isim, maas
FROM personel
WHERE maas > 4000;
```

| ISIM          | MAAS |
|---------------|------|
| Ali Yılmaz    | 5500 |
| Hatice Şahin  | 4500 |
| Mehmet Ozturk | 3500 |
| Mehmet Ozturk | 6000 |
| Mehmet Ozturk | 7000 |
| Veli Şahin    | 4500 |

-- 1) Maasi 3000'den fazla olan şehir ve işçi isimlerini gösteren sorguyu yazınız

```
SELECT sehir AS isci_veya_sehir_ismi, maas
FROM personel
WHERE maas > 4000
UNION
SELECT isim AS isci_veya_sehir_ismi, maas
FROM personel
WHERE maas > 4000;
```

| ISCI_VEYA_SEHIR_ISMI | MAAS |
|----------------------|------|
| Ali Yılmaz           | 5500 |
| Ankara               | 4500 |
| Ankara               | 7000 |
| Bursa                | 4500 |
| Hatice Şahin         | 4500 |
| İstanbul             | 4500 |
| İstanbul             | 5500 |
| İzmir                | 6000 |
| Mehmet Ozturk        | 6000 |
| Mehmet Ozturk        | 7000 |
| Veli Şahin           | 4500 |

-- 2) Mehmet Ozturk ismindeki personelin aldığı maaşları ve

-- İstanbul'daki personelin maaşlarını bir tabloda gösteren sorgu yazınız

```
SELECT sehir AS isci_veya_sehir_ismi, maas
FROM personel
WHERE sehir = 'İstanbul'
UNION
SELECT isim AS isci_veya_sehir_ismi, maas
FROM personel
WHERE isim = 'Mehmet Ozturk';
```

| ISCI_VEYA_SEHIR_ISMI | MAAS |
|----------------------|------|
| İstanbul             | 4500 |
| İstanbul             | 5500 |
| Mehmet Ozturk        | 3500 |
| Mehmet Ozturk        | 6000 |
| Mehmet Ozturk        | 7000 |

-- NOT: 2.sorgunun sonuna ORDER BY komutunu kullanırsanız tüm tabloyu istediğiniz sıralamaya göre sıralar

```
SELECT sehir AS isci_veya_sehir_ismi, maas
FROM personel
WHERE sehir = 'İstanbul'
UNION
SELECT isim AS isci_veya_sehir_ismi, maas
FROM personel
WHERE isim = 'Mehmet Ozturk'
ORDER BY maas;
```

| ISCI_VEYA_SEHIR_ISMI | MAAS |
|----------------------|------|
| Mehmet Ozturk        | 3500 |
| İstanbul             | 4500 |
| İstanbul             | 5500 |
| Mehmet Ozturk        | 6000 |
| Mehmet Ozturk        | 7000 |

```
-- 3) Sehirlerden odenen ucret 3000'den fazla olan ve personelden ucreti 5000'den
-- az olanlari bir tabloda gosteren sorguyu yaziniz
SELECT sehir AS isci_veya_sehir_ismi, maas
FROM personel
WHERE maas > 3000
UNION
SELECT isim AS isci_veya_sehir_ismi, maas
FROM personel
WHERE maas < 5000;
```

| ISCI_VEYA_SEHIR_ISMI | MAAS |
|----------------------|------|
| Ankara               | 3500 |
| Ankara               | 4500 |
| Ankara               | 7000 |
| Bursa                | 4500 |
| Hatice Sahin         | 4500 |
| Istanbul             | 4500 |
| Istanbul             | 5500 |
| Izmir                | 6000 |
| Mehmet Ozturk        | 3500 |
| Veli Sahin           | 4500 |

## UNION OPERATOR ile 2 Tablodan Data Birleştirme

```
CREATE TABLE personel
(
id number(9),
isim varchar2(50),
sehir varchar2(50),
maas number(20),
sirket varchar2(20),
CONSTRAINT personel_pk PRIMARY KEY (id)
);

INSERT INTO personel VALUES(123456789, 'Ali Yilmaz', 'Istanbul', 5500, 'Honda');
INSERT INTO personel VALUES(234567890, 'Veli Sahin', 'Istanbul', 4500, 'Toyota');
INSERT INTO personel VALUES(345678901, 'Mehmet Ozturk', 'Ankara', 3500, 'Honda');
INSERT INTO personel VALUES(456789012, 'Mehmet Ozturk', 'Izmir', 6000, 'Ford');
INSERT INTO personel VALUES(567890123, 'Mehmet Ozturk', 'Ankara', 7000, 'Tofas');
INSERT INTO personel VALUES(456715012, 'Veli Sahin', 'Ankara', 4500, 'Ford');
INSERT INTO personel VALUES(123456710, 'Hatice Sahin', 'Bursa', 4500, 'Honda');
```

```
SELECT * FROM personel;
```

| ID        | ISIM          | SEHIR    | MAAS | SIRKET |
|-----------|---------------|----------|------|--------|
| 123456789 | Ali Yilmaz    | Istanbul | 5500 | Honda  |
| 234567890 | Veli Sahin    | Istanbul | 4500 | Toyota |
| 345678901 | Mehmet Ozturk | Ankara   | 3500 | Honda  |
| 456789012 | Mehmet Ozturk | Izmir    | 6000 | Ford   |
| 567890123 | Mehmet Ozturk | Ankara   | 7000 | Tofas  |
| 456715012 | Veli Sahin    | Ankara   | 4500 | Ford   |
| 123456710 | Hatice Sahin  | Bursa    | 4500 | Honda  |

```
CREATE TABLE personel_bilgi
(
id number(9),
tel char(10) UNIQUE ,
cocuk_sayisi number(2),
CONSTRAINT personel_bilgi_fk FOREIGN KEY (id) REFERENCES personel(id)
);
```

```

INSERT INTO personel_bilgi VALUES(123456789, '5302345678', 5);
INSERT INTO personel_bilgi VALUES(234567890, '5422345678', 4);
INSERT INTO personel_bilgi VALUES(345678901, '5354561245', 3);
INSERT INTO personel_bilgi VALUES(456789012, '5411452659', 3);
INSERT INTO personel_bilgi VALUES(567890123, '5551253698', 2);
INSERT INTO personel_bilgi VALUES(456789012, '5524578574', 2);
INSERT INTO personel_bilgi VALUES(123456710, '5537488585', 1);

```

```
SELECT * FROM personel_bilgi;
```

| ID        | TEL        | COCUK_SAYISI |
|-----------|------------|--------------|
| 123456789 | 5302345678 | 5            |
| 234567890 | 5422345678 | 4            |
| 345678901 | 5354561245 | 3            |
| 456789012 | 5411452659 | 3            |
| 567890123 | 5551253698 | 2            |
| 456789012 | 5524578574 | 2            |
| 123456710 | 5537488585 | 1            |

```

--id'si 12345678 olan personelin
--Personel tablosundan sehir ve maasini,
--personel_bilgi tablosundan da tel ve cocuk sayisini yazdirin
SELECT sehir AS tel_sehir, maas AS cocuk_sayisi_maas
FROM personel
WHERE id = 123456789
UNION
SELECT tel, cocuk_sayisi
FROM personel_bilgi
WHERE id = 123456789;

```

| TEL_SEHIR  | COCUK_SAYISI_MAAS |
|------------|-------------------|
| 5302345678 | 5                 |
| Istanbul   | 5500              |

**NOT:** Union işlemi yaparken

- 1) Her 2 QUERY'den elde edeceğiniz tabloların sütun sayıları eşit olmalı.
- 2) Alt alta gelecek sütunların data type'leri aynı olmalı.

```

-- personel tablosundan
-- maasi 4500 olanların isim ve id'lerini ve
-- personel_bilgi tablosundan 2 cocugu olanların tel ve id'lerini tek tabloda
-- listeleyin
SELECT isim AS isim_veya_tel, id
FROM personel
WHERE maas = 4500
UNION
SELECT tel, id
FROM personel_bilgi
WHERE cocuk_sayisi = 2;

```

| ISIM_VEYA_TEL | ID        |
|---------------|-----------|
| 5524578574    | 456789012 |
| 5551253698    | 567890123 |
| Hatice Sahin  | 123456710 |
| Veli Sahin    | 234567890 |
| Veli Sahin    | 456715012 |

## UNION ALL OPERATOR

- UNION ALL, UNION dan farklı olarak her iki sorgudan gelen TÜM KAYITLARI listeler.
- UNION ALL yaparken de sütun sayısı ve sütunların data type i uyumu gereklidir.

```

--maasi 5000'den az olanları yazdirin
SELECT *
FROM personel
WHERE maas < 5000;

```

| ID        | ISIM          | SEHIR    | MAAS | SIRKET |
|-----------|---------------|----------|------|--------|
| 234567890 | Veli Sahin    | Istanbul | 4500 | Toyota |
| 345678901 | Mehmet Ozturk | Ankara   | 3500 | Honda  |
| 456715012 | Veli Sahin    | Ankara   | 4500 | Ford   |
| 123456710 | Hatice Sahin  | Bursa    | 4500 | Honda  |



```
--ayni sorguyu 2 kere yazip UNION ile birlestirin
SELECT *
FROM personel
WHERE maas < 5000
UNION
SELECT *
FROM personel
WHERE maas < 5000;
```

| ID        | ISIM          | SEHIR    | MAAS | SIRKET |
|-----------|---------------|----------|------|--------|
| 234567890 | Veli Sahin    | Istanbul | 4500 | Toyota |
| 345678901 | Mehmet Ozturk | Ankara   | 3500 | Honda  |
| 456715012 | Veli Sahin    | Ankara   | 4500 | Ford   |
| 123456710 | Hatice Sahin  | Bursa    | 4500 | Honda  |

```
--ayni sorguyu 2 kere yazip UNION ALL ile birlestirin
SELECT *
FROM personel
WHERE maas < 5000
UNION ALL
SELECT *
FROM personel
WHERE maas < 5000;
```

| ID        | ISIM          | SEHIR    | MAAS | SIRKET |
|-----------|---------------|----------|------|--------|
| 234567890 | Veli Sahin    | Istanbul | 4500 | Toyota |
| 345678901 | Mehmet Ozturk | Ankara   | 3500 | Honda  |
| 456715012 | Veli Sahin    | Ankara   | 4500 | Ford   |
| 123456710 | Hatice Sahin  | Bursa    | 4500 | Honda  |
| 234567890 | Veli Sahin    | Istanbul | 4500 | Toyota |
| 345678901 | Mehmet Ozturk | Ankara   | 3500 | Honda  |
| 456715012 | Veli Sahin    | Ankara   | 4500 | Ford   |
| 123456710 | Hatice Sahin  | Bursa    | 4500 | Honda  |

```
--maasi 5000'den az olanlari isim ve maas larini yazdirin
SELECT isim, maas
FROM personel
WHERE maas < 5000;
```

| ISIM          | MAAS |
|---------------|------|
| Veli Sahin    | 4500 |
| Mehmet Ozturk | 3500 |
| Veli Sahin    | 4500 |
| Hatice Sahin  | 4500 |

```
--
SELECT isim, maas
FROM personel
WHERE maas < 5000
UNION
SELECT isim, maas
FROM personel
WHERE maas < 5000;
```

| ISIM          | MAAS |
|---------------|------|
| Hatice Sahin  | 4500 |
| Mehmet Ozturk | 3500 |
| Veli Sahin    | 4500 |

```
--
SELECT isim, maas
FROM personel
WHERE maas < 5000
UNION ALL
SELECT isim, maas
FROM personel
WHERE maas < 5000;
```

| ISIM          | MAAS |
|---------------|------|
| Veli Sahin    | 4500 |
| Mehmet Ozturk | 3500 |
| Veli Sahin    | 4500 |
| Hatice Sahin  | 4500 |
| Veli Sahin    | 4500 |
| Mehmet Ozturk | 3500 |
| Veli Sahin    | 4500 |
| Hatice Sahin  | 4500 |

- Eğer sonuç tablosu sıralı görmek isterseniz son satıra ORDER BY eklemek yeterli olur.

```
SELECT isim, maas
FROM personel
WHERE maas < 5000
UNION ALL
SELECT isim, maas
FROM personel
WHERE maas < 5000
ORDER BY maas;
--yada ORDER BY 2;
```

| ISIM          | MAAS |
|---------------|------|
| Mehmet Ozturk | 3500 |
| Mehmet Ozturk | 3500 |
| Hatice Sahin  | 4500 |
| Veli Sahin    | 4500 |
| Veli Sahin    | 4500 |
| Veli Sahin    | 4500 |
| Veli Sahin    | 4500 |
| Hatice Sahin  | 4500 |

```
--1) Personel tablosundan Istanbul veya Ankara'da calisanlari id'lerini yazdir
SELECT id
FROM personel
WHERE sehir IN ('Istanbul', 'Ankara');
```

| ID        |
|-----------|
| 123456789 |
| 234567890 |
| 345678901 |
| 567890123 |
| 456715012 |

```
--2) Personel_bilgi tablosundan 2 veya 3 cocugu olanlari id lerini yazdirin
SELECT id
FROM personel_bilgi
WHERE cocuk_sayisi IN (2, 3);
```

| ID        |
|-----------|
| 345678901 |
| 456789012 |
| 567890123 |
| 456789012 |

- **UNION** işlemi 2 veya daha çok **SELECT** işleminin sonuç **KÜMELERİNİ** birleştirmek için kullanılır, Aynı kayıt birden fazla olursa, sadece bir tanesini alır.
- **UNION ALL** ise tekrarlı elemanları, tekrar sayısınca yazar.

**NOT:** UNION ALL ile birleştirmelerde de

- 1) Her 2 QUERY'den elde edeceğiniz tabloların sütun sayıları eşit olmalı.
- 2) Alt alta gelecek sütunların data type'leri aynı olmalı.

```
-- 1) Tabloda personel maasi 4000'den çok olan tüm şehirleri ve maaşları yazdırın
SELECT şehir, maaş
FROM personel
WHERE maaş > 4000;
```

| ŞEHİR    | MAAŞ |
|----------|------|
| Istanbul | 5500 |
| Istanbul | 4500 |
| Izmir    | 6000 |
| Ankara   | 7000 |
| Ankara   | 4500 |
| Bursa    | 4500 |

```
-- 2) Tabloda personel maasi 5000'den az olan tüm isimleri ve maaşları yazdırın
SELECT isim, maaş
FROM personel
WHERE maaş < 5000;
```

| İSİM          | MAAŞ |
|---------------|------|
| Veli Şahin    | 4500 |
| Mehmet Öztürk | 3500 |
| Veli Şahin    | 4500 |
| Hatice Şahin  | 4500 |

```
-- 3) İki sorguyu UNION ve UNION ALL ile birleştirin
```

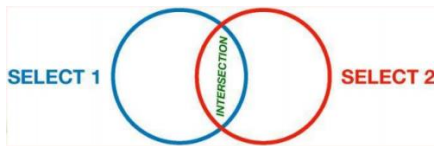
```
SELECT şehir, maaş
FROM personel
WHERE maaş > 4000
UNION
SELECT isim, maaş
FROM personel
WHERE maaş < 5000;
```

```
SELECT şehir, maaş
FROM personel
WHERE maaş > 4000
UNION ALL
SELECT isim, maaş
FROM personel
WHERE maaş < 5000;
```

| ŞEHİR         | MAAŞ |
|---------------|------|
| Ankara        | 4500 |
| Ankara        | 7000 |
| Bursa         | 4500 |
| Hatice Şahin  | 4500 |
| Istanbul      | 4500 |
| Istanbul      | 5500 |
| Izmir         | 6000 |
| Mehmet Öztürk | 3500 |
| Veli Şahin    | 4500 |

| ŞEHİR         | MAAŞ |
|---------------|------|
| Istanbul      | 5500 |
| Istanbul      | 4500 |
| Izmir         | 6000 |
| Ankara        | 7000 |
| Ankara        | 4500 |
| Bursa         | 4500 |
| Veli Şahin    | 4500 |
| Mehmet Öztürk | 3500 |
| Veli Şahin    | 4500 |
| Hatice Şahin  | 4500 |

## INTERSECT OPERATOR



-- 1) hem ankara veya istanbulda calisan hem de 2 veya 3 cocugu olan personelin  
--id'lerini yazdrrz

```
SELECT id
FROM personel
WHERE sehir IN ('Istanbul', 'Ankara')
INTERSECT
SELECT id
FROM personel_bilgi
WHERE cocuk_sayisi IN (2, 3);
```

| ID        |
|-----------|
| 345678901 |
| 567890123 |

-- 2a) Maasi 4800'den az olanlar veya 5000'den cok olanlarin id'lerini listeleyin

```
SELECT id
FROM personel
WHERE maas NOT BETWEEN 4800 AND 5500;
```

| ID        |
|-----------|
| 234567890 |
| 345678901 |
| 456789012 |
| 567890123 |
| 456715012 |
| 123456710 |

-- 2b) Personel\_bilgi tablosundan 2 veya 3 cocugu olanlarin id lerini yazdirin

```
SELECT id
FROM personel_bilgi
WHERE cocuk_sayisi IN (2,3);
```

| ID        |
|-----------|
| 345678901 |
| 456789012 |
| 567890123 |
| 456789012 |

-- 2c) Hem maas i 4800'den az olanlar veya 5000'den cok olanlar ile hem de 2 veya 3  
-- cocugu olanlarin id lerini yazdirin

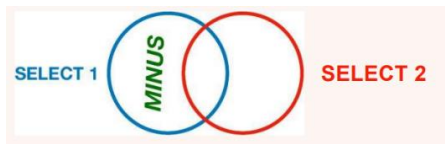
```
SELECT id
FROM personel
WHERE maas NOT BETWEEN 4800 AND 5500
INTERSECT
SELECT id
FROM personel_bilgi
WHERE cocuk_sayisi IN (2,3);
```

| ID        |
|-----------|
| 345678901 |
| 456789012 |
| 567890123 |

```
-- 3) Honda,Ford ve Tofas'ta calisan ayni isimde personel varsa listeleyin
SELECT isim
FROM personel
WHERE sirket = 'Honda'
INTERSECT
SELECT isim
FROM personel
WHERE sirket = 'Ford'
INTERSECT
SELECT isim
FROM personel
WHERE sirket = 'Tofas';
```

| ISIM          |
|---------------|
| Mehmet Ozturk |

## MINUS OPERATOR



```
-- 1) 5000'den az maas alip Honda'da calismayanlari yazdirin
SELECT *
FROM personel
WHERE maas < 5000
MINUS
SELECT *
FROM personel
WHERE sirket = 'Honda';
```

| ID        | ISIM       | SEHIR    | MAAS | SIRKET |
|-----------|------------|----------|------|--------|
| 234567890 | Veli Sahin | Istanbul | 4500 | Toyota |
| 456715012 | Veli Sahin | Ankara   | 4500 | Ford   |

```
-- 2) isim i Mehmet Ozturk olup Istanbul'da calismayanlarin
-- isimlerini ve sehirlerini listeleyin
SELECT isim, sehir
FROM personel
WHERE isim = 'Mehmet Ozturk'
MINUS
SELECT isim, sehir
FROM personel
WHERE sehir = 'Istanbul';
```

| ISIM          | SEHIR  |
|---------------|--------|
| Mehmet Ozturk | Ankara |
| Mehmet Ozturk | Izmir  |

## SQL-9 JOINS, LIKE, NOT LIKE

28.10.2020

```
CREATE TABLE sirketler
(
sirket_id number(9),
sirket_isim varchar2(20)
);
```

```
INSERT INTO sirketler VALUES(100, 'Toyota');
INSERT INTO sirketler VALUES(101, 'Honda');
INSERT INTO sirketler VALUES(102, 'Ford');
INSERT INTO sirketler VALUES(103, 'Hyundai');
```

```
SELECT * FROM sirketler;
```

| SIRKET_ID | SIRKET_ISIM |
|-----------|-------------|
| 100       | Toyota      |
| 101       | Honda       |
| 102       | Ford        |
| 103       | Hyundai     |

```
CREATE TABLE siparisler
(
siparis_id number(9),
sirket_id number(9),
siparis_tarihi date
);
```

```
INSERT INTO siparisler VALUES(11, 101, '17-Apr-2020');
INSERT INTO siparisler VALUES(22, 102, '18-Apr-2020');
INSERT INTO siparisler VALUES(33, 103, '19-Apr-2020');
INSERT INTO siparisler VALUES(44, 104, '20-Apr-2020');
INSERT INTO siparisler VALUES(55, 105, '21-Apr-2020');
```

| SIPARIS_ID | SIRKET_ID | SIPARIS_TARIHI |
|------------|-----------|----------------|
| 11         | 101       | 17-APR-20      |
| 22         | 102       | 18-APR-20      |
| 33         | 103       | 19-APR-20      |
| 44         | 104       | 20-APR-20      |
| 55         | 105       | 21-APR-20      |

```
SELECT * FROM siparisler;
```

## JOIN CONDITIONS

- **JOINS** iki tablodaki dataları birleştirmek için kullanılır.
- Şu ana kadar gördüğümüz **Union**, **Intersect** ve **Minus** sorgu sonuçları için kullanılır. Tablolar için ise JOIN kullanılır. 5 Çeşit Join vardır

- 1) **INNER JOIN** iki tablodaki ortak dataları gösterir.
- 2) **LEFT JOIN** ilk tabloda olan tüm rekordları gösterir.
- 3) **RIGHT JOIN** ikinci tabloda olan tüm rekordları gösterir.
- 4) **FULL JOIN** iki tablodaki tüm rekordları gösterir.
- 5) **SELF JOIN** bir tablonun kendi içinde Join edilmesi ile oluşur.

- 1) **INNER JOIN** iki tablodaki ortak dataları gösterir;

- Select'ten sonra tabloda görmek istediğiniz sütunları yazarken Tablo\_adi.field\_adi şeklinde yazın.
- From'dan sonra tablo ismi yazarken 1.Tablo ismi + **INNER JOIN** + 2.Tablo ismi yazmalıyız.
- Join'i hangi kurala göre yapacağınızı belirtmelisiniz. Bunun için ON+ kuralımız yazılmalı.

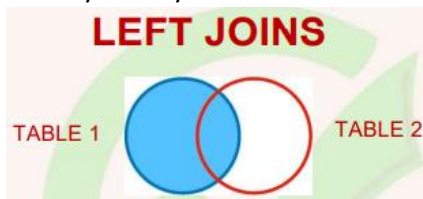


```
-- İki Tabloda sirket_id 'si aynı olanların sirket_isim, siparis_id ve
-- siparis_tarih leri ile yeni bir tablo oluşturun
SELECT sirketler.sirket_isim, siparisler.siparis_id, siparisler.siparis_tarihi
FROM sirketler INNER JOIN siparisler
ON sirketler.sirket_id = siparisler.sirket_id;
```

| SIRKET_ISIM | SIPARIS_ID | SIPARIS_TARIHI |
|-------------|------------|----------------|
| Honda       | 11         | 17-APR-20      |
| Ford        | 22         | 18-APR-20      |
| Hyundai     | 33         | 19-APR-20      |

- 2) **LEFT JOIN** ilk tabloda olan tüm rekordları gösterir;

- İlk tablodaki datalara 2.tablodan gelen ek datalar varsa bu ek datalar ortak datalar için gösterilir ancak ortak olmayan datalar için o kısımlar boş kalır
- İlk yazdığınız tablonun tamamını aldığı için hangi tabloyu istediğimize karar verip önce onu yazmalıyız



```
SELECT sirketler.sirket_isim, siparisler.siparis_id, siparisler.siparis_tarihi
FROM sirketler LEFT JOIN siparisler
ON sirketler.sirket_id = siparisler.sirket_id;
```

| SIRKET_ISIM | SIPARIS_ID | SIPARIS_TARIHI |
|-------------|------------|----------------|
| Honda       | 11         | 17-APR-20      |
| Ford        | 22         | 18-APR-20      |
| Hyundai     | 33         | 19-APR-20      |
| Toyota      | -          | -              |

```
SELECT sirketler.sirket_id, siparisler.siparis_id, siparisler.siparis_tarihi
FROM sirketler LEFT JOIN siparisler
ON sirketler.sirket_id = siparisler.sirket_id;
```

| SIRKET_ID | SIPARIS_ID | SIPARIS_TARIHI |
|-----------|------------|----------------|
| 101       | 11         | 17-APR-20      |
| 102       | 22         | 18-APR-20      |
| 103       | 33         | 19-APR-20      |
| 100       | -          | -              |

### 3) **RIGHT JOIN** ikinci tabloda olan tüm rekordları gösterir;

- İkinci tablodaki datalara 1.tablodan gelen ek datalar varsa bu ek datalar ortak datalar için gösterilir ancak ortak olmayan datalar için o kısımlar boş kalır.



```
SELECT sirketler.sirket_isim, siparisler.siparis_id, siparisler.siparis_tarihi
FROM sirketler RIGHT JOIN siparisler
ON sirketler.sirket_id = siparisler.sirket_id;
```

| SIRKET_ISIM | SIPARIS_ID | SIPARIS_TARIHI |
|-------------|------------|----------------|
| Honda       | 11         | 17-APR-20      |
| Ford        | 22         | 18-APR-20      |
| Hyundai     | 33         | 19-APR-20      |
| -           | 55         | 21-APR-20      |
| -           | 44         | 20-APR-20      |

### 4) **FULL JOIN** iki tabloda var olan tüm rekordları gösterir;

- Bir tabloda olup ötekinde olmayan datalar boş kalır

```
SELECT sirketler.sirket_isim, siparisler.siparis_id, siparisler.siparis_tarihi
FROM sirketler FULL JOIN siparisler
ON sirketler.sirket_id = siparisler.sirket_id;
```

| SIRKET_ISIM | SIPARIS_ID | SIPARIS_TARIHI |
|-------------|------------|----------------|
| Honda       | 11         | 17-APR-20      |
| Ford        | 22         | 18-APR-20      |
| Hyundai     | 33         | 19-APR-20      |
| -           | 44         | 20-APR-20      |
| -           | 55         | 21-APR-20      |
| Toyota      | -          | -              |

### 5) **SELF JOIN** bir tablonun kendi içinde Join edilmesi ile oluşur;

```
CREATE TABLE personel
(
id number(2),
isim varchar2(20),
title varchar2(60),
yonetici_id number(2)
);
```

```

INSERT INTO personel VALUES(1, 'Ali Can', 'SDET', 2);
INSERT INTO personel VALUES(2, 'Veli Cem', 'QA', 3);
INSERT INTO personel VALUES(3, 'Ayse Gul', 'QA Lead', 4);
INSERT INTO personel VALUES(4, 'Fatma Can', 'CEO', 5);

```

```
SELECT * FROM personel;
```

| ID | ISIM      | TITLE   | YONETICI_ID |
|----|-----------|---------|-------------|
| 1  | Ali Can   | SDET    | 2           |
| 2  | Veli Cem  | QA      | 3           |
| 3  | Ayşe Gul  | QA Lead | 4           |
| 4  | Fatma Can | CEO     | 5           |

```

-- Her personelin yanına yönetici ismini yazdıran bir tablo oluşturun
SELECT p1.isim AS personel_ismi, p2.isim AS yönetici_ismi
FROM personel p1 INNER JOIN personel p2
ON p1.yoneticici_id = p2.id;

```

| PERSONEL_ISMI | YONETICI_ISMI |
|---------------|---------------|
| Ali Can       | Veli Cem      |
| Veli Cem      | Ayşe Gul      |
| Ayşe Gul      | Fatma Can     |

## LIKE CONDITIONS

LIKE condition WHERE ile kullanılarak SELECT, INSERT, UPDATE, veya DELETE statement ile çalışan **wildcards**'a izin verir. Ve bize **pattern matching** yapma imkânı sunar.

```

CREATE TABLE musteriler
(
id number(10) UNIQUE,
isim varchar2(50) NOT NULL,
gelir number(6)
);

```

```

INSERT INTO musteriler (id, isim, gelir) VALUES (1001, 'Ali', 62000);
INSERT INTO musteriler (id, isim, gelir) VALUES (1002, 'Ayşe', 57500);
INSERT INTO musteriler (id, isim, gelir) VALUES (1003, 'Feride', 71000);
INSERT INTO musteriler (id, isim, gelir) VALUES (1004, 'Fatma', 42000);
INSERT INTO musteriler (id, isim, gelir) VALUES (1005, 'Kasim', 44000);

```

```
SELECT * FROM musteriler;
```

| ID   | ISIM   | GELİR |
|------|--------|-------|
| 1001 | Ali    | 62000 |
| 1002 | Ayşe   | 57500 |
| 1003 | Feride | 71000 |
| 1004 | Fatma  | 42000 |
| 1005 | Kasim  | 44000 |

```

-- Ismi Ali olan musterilerin tüm bilgilerini yazdıran QUERY yazın
SELECT *
FROM musteriler
WHERE isim = 'Ali';

```

| ID   | ISIM | GELİR |
|------|------|-------|
| 1001 | Ali  | 62000 |



- **%** => 0 veya birden fazla karakter belirtir

```
-- Ismi A harfi ile baslayan musterilerin tum bilgilerini yazdiran QUERY yazin
SELECT *
FROM musteriler
WHERE isim LIKE 'A%';
```

| ID   | ISIM | GELIR |
|------|------|-------|
| 1001 | Ali  | 62000 |
| 1002 | Ayse | 57500 |

```
-- Ismi e harfi ile biten musterilerin isimlerini ve gelir'lerini yazdiran QUERY
SELECT isim, gelir
FROM musteriler
WHERE isim LIKE '%e';
```

| ISIM   | GELIR |
|--------|-------|
| Ayse   | 57500 |
| Feride | 71000 |

```
-- Isminin icinde er olan musterilerin isimlerini ve gelir'lerini yazdiran QUERY
SELECT isim, gelir
FROM musteriler
WHERE isim LIKE '%er%';
```

| ISIM   | GELIR |
|--------|-------|
| Feride | 71000 |

- **\_** => sadece bir karakteri gösterir.

```
-- Ismi 5 harfli olup son 4 harfi atma olan musterilerin tum bilgilerini yazdiran QUERY
SELECT *
FROM musteriler
WHERE isim LIKE '_atma';
```

| ID   | ISIM  | GELIR |
|------|-------|-------|
| 1004 | Fatma | 42000 |

```
-- Ikinci harfi a olan musterilerin tum bilgilerini yazdiran QUERY yazin
SELECT *
FROM musteriler
WHERE isim LIKE '_a%';
```

| ID   | ISIM  | GELIR |
|------|-------|-------|
| 1004 | Fatma | 42000 |
| 1005 | Kasim | 44000 |

```
-- Ucuncu harfi s olan musterilerin tum bilgilerini yazdiran QUERY yazin
SELECT *
FROM musteriler
WHERE isim LIKE '__s%';
```

| ID   | ISIM  | GELIR |
|------|-------|-------|
| 1002 | Ayse  | 57500 |
| 1005 | Kasim | 44000 |

```
-- İlk harfi F olan en az 4 harfli musterilerin tum bilgilerini yazdiran QUERY
SELECT *
FROM musteriler
WHERE isim LIKE 'F_%_%_%';
```

| ID   | ISIM   | GELIR |
|------|--------|-------|
| 1003 | Feride | 71000 |
| 1004 | Fatma  | 42000 |

```
-- Ikinci harfi a, 4.harfi m olan musterilerin tum bilgilerini yazdiran QUERY yazin
SELECT *
FROM musteriler
WHERE isim LIKE '_a_m%';
```

| ID   | ISIM  | GELIR |
|------|-------|-------|
| 1004 | Fatma | 42000 |

```
CREATE TABLE kelimeler
(
id number(10) UNIQUE,
kelime varchar2(50) NOT NULL,
Harf_sayisi number(6)
);
```

```
INSERT INTO kelimeler VALUES (1001, 'hot', 3);
INSERT INTO kelimeler VALUES (1002, 'hat', 3);
INSERT INTO kelimeler VALUES (1003, 'hit', 3);
INSERT INTO kelimeler VALUES (1004, 'hbt', 3);
INSERT INTO kelimeler VALUES (1008, 'hct', 3);
INSERT INTO kelimeler VALUES (1005, 'adem', 4);
INSERT INTO kelimeler VALUES (1006, 'selim', 5);
INSERT INTO kelimeler VALUES (1007, 'yusuf', 5);
```

```
SELECT * FROM kelimeler;
```

| ID   | KELIME | HARF_SAYISI |
|------|--------|-------------|
| 1001 | hot    | 3           |
| 1002 | hat    | 3           |
| 1003 | hit    | 3           |
| 1004 | hbt    | 3           |
| 1008 | hct    | 3           |
| 1005 | adem   | 4           |
| 1006 | selim  | 5           |
| 1007 | yusuf  | 5           |

```
-- h ile baslayan kelimeleri yazdiran QUERY
SELECT kelime
FROM kelimeler
WHERE kelime LIKE 'h%';
```

| KELIME |
|--------|
| hot    |
| hat    |
| hit    |
| hbt    |
| hct    |

- **[ ] REGEXP\_LIKE** => sadece bir karakteri gösterir

-- İlk harfi h, son harfi t olup 2.harfi a veya i olan 3 harfli kelimelerin tum bilgilerini yazdiran QUERY yazin

```
SELECT *
FROM kelimeler
WHERE REGEXP_LIKE (kelime, 'h[ai]t');
```

| ID   | KELIME | HARF_SAYISI |
|------|--------|-------------|
| 1002 | hat    | 3           |
| 1003 | hit    | 3           |

-- İlk harfi h, son harfi t olup 2.harfi a ile k arasinda olan 3 harfli kelimelerin tum bilgilerini yazdiran QUERY yazin

```
SELECT *
FROM kelimeler
WHERE REGEXP_LIKE (kelime, 'h[a-k]t');
```

| ID   | KELIME | HARF_SAYISI |
|------|--------|-------------|
| 1002 | hat    | 3           |
| 1003 | hit    | 3           |
| 1004 | hbt    | 3           |
| 1008 | hct    | 3           |

-- Icinde m veya i olan musterilerin tum bilgilerini yazdiran QUERY yazin

```
SELECT *
FROM kelimeler
WHERE REGEXP_LIKE (kelime, '[mi](*)');
```

| ID   | KELIME | HARF_SAYISI |
|------|--------|-------------|
| 1003 | hit    | 3           |
| 1005 | adem   | 4           |
| 1006 | selim  | 5           |

-- a veya s ile baslayan kelimelerin tum bilgilerini yazdiran QUERY yazin

```
SELECT *
FROM kelimeler
WHERE REGEXP_LIKE (kelime, '^[as]');
```

| ID   | KELIME | HARF_SAYISI |
|------|--------|-------------|
| 1005 | adem   | 4           |
| 1006 | selim  | 5           |

-- m veya f ile biten kelimelerin tum bilgilerini yazdiran QUERY yazin

```
SELECT *
FROM kelimeler
WHERE REGEXP_LIKE (kelime, '[mf]$');
```

| ID   | KELIME | HARF_SAYISI |
|------|--------|-------------|
| 1005 | adem   | 4           |
| 1006 | selim  | 5           |
| 1007 | yusuf  | 5           |

## NOT LIKE CONDITIONS

-- ilk harfi h olmayan kelime lerin tum bilgilerini yazdiran QUERY yazin

```
SELECT *
FROM kelimeler
WHERE kelime NOT LIKE 'h%';
```

| ID   | KELIME | HARF_SAYISI |
|------|--------|-------------|
| 1005 | adem   | 4           |
| 1006 | selim  | 5           |
| 1007 | yusuf  | 5           |

```
-- a harfi icermeyen kelime lerin tum bilgilerini yazdiran QUERY yazin
SELECT *
FROM kelimeler
WHERE kelime NOT LIKE '%a%';
```

| ID   | KELIME | HARF_SAYISI |
|------|--------|-------------|
| 1001 | hot    | 3           |
| 1003 | hit    | 3           |
| 1004 | hbt    | 3           |
| 1008 | hct    | 3           |
| 1006 | selim  | 5           |
| 1007 | yusuf  | 5           |

```
-- ikinci ve ucuncu harfi 'de' olmayan kelimelerin tum bilgilerini yazdiran QUERY yazin
SELECT *
FROM kelimeler
WHERE kelime NOT LIKE '_de%';
```

| ID   | KELIME | HARF_SAYISI |
|------|--------|-------------|
| 1001 | hot    | 3           |
| 1002 | hat    | 3           |
| 1003 | hit    | 3           |
| 1004 | hbt    | 3           |
| 1008 | hct    | 3           |
| 1006 | selim  | 5           |
| 1007 | yusuf  | 5           |

```
-- Icinde usu olmayan kelime lerin tum bilgilerini yazdiran QUERY yazin
SELECT *
FROM kelimeler
WHERE kelime NOT LIKE '%usu%';
```

| ID   | KELIME | HARF_SAYISI |
|------|--------|-------------|
| 1001 | hot    | 3           |
| 1002 | hat    | 3           |
| 1003 | hit    | 3           |
| 1004 | hbt    | 3           |
| 1008 | hct    | 3           |
| 1005 | adem   | 4           |
| 1006 | selim  | 5           |

```
-- 2. harfi e, i veya o olmayan kelimelerin tum bilgilerini yazdiran QUERY yazin
SELECT *
FROM kelimeler
WHERE NOT REGEXP_LIKE (kelime, '[_eio]');
```

| ID   | KELIME | HARF_SAYISI |
|------|--------|-------------|
| 1002 | hat    | 3           |
| 1004 | hbt    | 3           |
| 1008 | hct    | 3           |
| 1007 | yusuf  | 5           |

## UPPER – LOWER – INITCAP

Tabloları yazdırırken dataları büyük harf, küçük harf veya ilk harfleri büyük diğerleri küçük harf yazdırmak için kullanırız

```
-- tablodaki tum kelime leri yazdiran QUERY yaziniz
SELECT kelime
FROM kelimeler;
```

```
-- tablodaki tüm kelime leri BÜYÜK harf olarak yazdıran QUERY yazdırın
SELECT UPPER (kelime)
FROM kelimeler;

-- tablodaki tüm kelime leri küçük harf olarak yazdıran QUERY yazdırın
SELECT LOWER (kelime)
FROM kelimeler;

-- tablodaki tüm kelime leri ilk harflerini BÜYÜK olarak yazdıran QUERY yazdırın
SELECT INITCAP (kelime)
FROM kelimeler;
```

| UPPER(KELIME) | LOWER(KELIME) | INITCAP(KELIME) |
|---------------|---------------|-----------------|
| HOT           | hot           | Hot             |
| HAT           | hat           | Hat             |
| HIT           | hit           | Hit             |
| HBT           | hbt           | Hbt             |
| HCT           | hct           | Hct             |
| ADEM          | adem          | Adem            |
| SELİM         | selim         | Selim           |
| YUSUF         | yusuf         | Yusuf           |

## DISTINCT

```
CREATE TABLE muster_i_urun
(
urun_id number(10),
muster_i_isim varchar2(50),
urun_isim varchar2(50)
);

INSERT INTO muster_i_urun VALUES (10, 'Ali', 'Portakal');
INSERT INTO muster_i_urun VALUES (10, 'Ali', 'Portakal');
INSERT INTO muster_i_urun VALUES (20, 'Veli', 'Elma');
INSERT INTO muster_i_urun VALUES (30, 'Ayse', 'Armut');
INSERT INTO muster_i_urun VALUES (20, 'Ali', 'Elma');
INSERT INTO muster_i_urun VALUES (10, 'Adem', 'Portakal');
INSERT INTO muster_i_urun VALUES (40, 'Veli', 'Kaysi');
INSERT INTO muster_i_urun VALUES (20, 'Elif', 'Elma');
```

```
SELECT * FROM muster_i_urun;
```

| URUN_ID | MUSTERI_ISIM | URUN_ISIM |
|---------|--------------|-----------|
| 10      | Ali          | Portakal  |
| 10      | Ali          | Portakal  |
| 20      | Veli         | Elma      |
| 30      | Ayse         | Armut     |
| 20      | Ali          | Elma      |
| 10      | Adem         | Portakal  |
| 40      | Veli         | Kaysi     |
| 20      | Elif         | Elma      |

```
-- satılan tüm ürünleri veren QUERY
SELECT urun_isim
FROM muster_i_urun;
-- satılan tüm ürünleri, tekrarsız yazdıran QUERY
SELECT DISTINCT urun_isim
FROM muster_i_urun;
```

| URUN_ISIM |
|-----------|
| Elma      |
| Portakal  |
| Kaysi     |
| Armut     |

```
SELECT DISTINCT muster_i_isim
FROM muster_i_urun;
```

| MUSTERI_ISIM |
|--------------|
| Veli         |
| Ayşe         |
| Elif         |
| Adem         |
| Ali          |

```
-- Kac farkli meyve satildigini veren QUERY yaziniz
SELECT COUNT (DISTINCT urun_isim) AS farkli_meyve_sayisi
FROM muster_i_urun;
```

| FARKLI_MEYVE_SAYISI |
|---------------------|
| 4                   |

```
SELECT COUNT (DISTINCT muster_i_isim) AS farkli_muster_i_sayisi
FROM muster_i_urun;
```

| FARKLI_MÜSTERI_SAYISI |
|-----------------------|
| 5                     |

## FETCH NEXT (SAYI) ROW ONLY - OFFSET

```
-- tabloyu urun_id ye göre siraliyiniz
SELECT *
FROM muster_i_urun
ORDER BY urun_id;
```

| URUN_ID | MUSTERI_ISIM | URUN_ISIM |
|---------|--------------|-----------|
| 10      | Ali          | Portakal  |
| 10      | Ali          | Portakal  |
| 10      | Adem         | Portakal  |
| 20      | Veli         | Elma      |
| 20      | Elif         | Elma      |
| 20      | Ali          | Elma      |
| 30      | Ayşe         | Armut     |
| 40      | Veli         | Kaysi     |

```
-- urun_id ye göre siraladigimizda ilk bes ürünü veren QUERY yaziniz
SELECT *
FROM muster_i_urun
ORDER BY urun_id
FETCH NEXT 5 ROW ONLY;
```

| URUN_ID | MUSTERI_ISIM | URUN_ISIM |
|---------|--------------|-----------|
| 10      | Ali          | Portakal  |
| 10      | Adem         | Portakal  |
| 10      | Ali          | Portakal  |
| 20      | Veli         | Elma      |
| 20      | Ali          | Elma      |

```
-- urun_id si en yüksek olan 2 ürünün tüm bilgilerini veren QUERY yaziniz
SELECT DISTINCT *
FROM muster_i_urun
ORDER BY urun_id DESC
FETCH NEXT 2 ROW ONLY;
```

| URUN_ID | MUSTERI_ISIM | URUN_ISIM |
|---------|--------------|-----------|
| 40      | Veli         | Kaysi     |
| 30      | Ayşe         | Armut     |

```
-- en büyük urun_id yi bulunuz
SELECT DISTINCT urun_id
FROM muster_i_urun
ORDER BY urun_id DESC
FETCH NEXT 1 ROW ONLY;
```

| URUN_ID |
|---------|
| 40      |

```
-- en büyük 2. urun_id yi bulunuz
SELECT DISTINCT urun_id
FROM muster_i_urun
ORDER BY urun_id DESC
OFFSET 1 ROW
FETCH NEXT 1 ROW ONLY;
```

| URUN_ID |
|---------|
| 30      |

```
-- tüm müşterileri isme göre siralayip 5. müşterinin veren QUERY
SELECT *
FROM muster_i_urun
ORDER BY muster_i_isim
OFFSET 4 ROW
FETCH NEXT 1 ROW ONLY;
```

| URUN_ID | MUSTERI_ISIM | URUN_ISIM |
|---------|--------------|-----------|
| 30      | Ayşe         | Armut     |

```
-- Sirali tablodan 4. kayittan 7.kayida kadar olan kayitlari listeleyin
SELECT *
FROM muster_i_urun
ORDER BY urun_id
OFFSET 3 ROW
FETCH NEXT 4 ROW ONLY;
```

| URUN_ID | MUSTERI_ISIM | URUN_ISIM |
|---------|--------------|-----------|
| 20      | Veli         | Elma      |
| 20      | Elif         | Elma      |
| 20      | Ali          | Elma      |
| 30      | Ayşe         | Armut     |

## PIVOT CLAUSES

```
CREATE TABLE musteri_urun
(
urun_id number(10),
musteri_isim varchar2(50),
urun_isim varchar2(50)
);
INSERT INTO musteri_urun VALUES (10, 'Ali', 'Portakal');
INSERT INTO musteri_urun VALUES (10, 'Ali', 'Portakal');
INSERT INTO musteri_urun VALUES (20, 'Veli', 'Elma');
INSERT INTO musteri_urun VALUES (30, 'Ayse', 'Armut');
INSERT INTO musteri_urun VALUES (20, 'Ali', 'Elma');
INSERT INTO musteri_urun VALUES (10, 'Adem', 'Portakal');
INSERT INTO musteri_urun VALUES (40, 'Veli', 'Kaysi');
INSERT INTO musteri_urun VALUES (20, 'Elif', 'Elma');
```

```
SELECT * FROM musteri_urun;
```

| URUN_ID | MUSTERI_ISIM | URUN_ISIM |
|---------|--------------|-----------|
| 10      | Ali          | Portakal  |
| 10      | Ali          | Portakal  |
| 20      | Veli         | Elma      |
| 30      | Ayşe         | Armut     |
| 20      | Ali          | Elma      |
| 10      | Adem         | Portakal  |
| 40      | Veli         | Kaysi     |
| 20      | Elif         | Elma      |

```
SELECT *
FROM (SELECT musteri_isim, urun_isim FROM musteri_urun)
PIVOT (COUNT (urun_isim) FOR urun_isim IN ('Portakal', 'Elma', 'Armut', 'Kaysi'));
```

| MUSTERI_ISIM | 'Portakal' | 'Elma' | 'Armut' | 'Kaysi' |
|--------------|------------|--------|---------|---------|
| Veli         | 0          | 1      | 0       | 1       |
| Ayşe         | 0          | 0      | 1       | 0       |
| Elif         | 0          | 1      | 0       | 0       |
| Adem         | 1          | 0      | 0       | 0       |
| Ali          | 2          | 1      | 0       | 0       |

```
SELECT *
FROM (SELECT musteri_isim, urun_isim FROM musteri_urun)
PIVOT (COUNT (musteri_isim) FOR musteri_isim IN ('Ayse', 'Veli', 'Ali', 'Adem', 'Elif'));
```

| URUN_ISIM | 'Ayse' | 'Veli' | 'Ali' | 'Adem' | 'Elif' |
|-----------|--------|--------|-------|--------|--------|
| Elma      | 0      | 1      | 1     | 0      | 1      |
| Portakal  | 0      | 0      | 2     | 1      | 0      |
| Kaysi     | 0      | 1      | 0     | 0      | 0      |
| Armut     | 1      | 0      | 0     | 0      | 0      |

```
SELECT *
FROM (SELECT musteri_isim, urun_id FROM musteri_urun)
PIVOT (COUNT (urun_id) FOR urun_id IN (10, 20, 30, 40));
```

| MUSTERI_ISIM | 10 | 20 | 30 | 40 |
|--------------|----|----|----|----|
| Veli         | 0  | 1  | 0  | 1  |
| Ayşe         | 0  | 0  | 1  | 0  |
| Elif         | 0  | 1  | 0  | 0  |
| Adem         | 1  | 0  | 0  | 0  |
| Ali          | 2  | 1  | 0  | 0  |



## ALTER TABLE STATEMENT

```
CREATE TABLE personel
(
id number(9),
isim varchar2(50),
sehir varchar2(50),
maas number(20),
sirket varchar2(20),
CONSTRAINT personel_pk PRIMARY KEY (id)
);
INSERT INTO personel VALUES(123456789, 'Ali Yilmaz', 'Istanbul', 5500, 'Honda');
INSERT INTO personel VALUES(234567890, 'Veli Sahin', 'Istanbul', 4500, 'Toyota');
INSERT INTO personel VALUES(345678901, 'Mehmet Ozturk', 'Ankara', 3500, 'Honda');
INSERT INTO personel VALUES(456789012, 'Mehmet Ozturk', 'Izmir', 6000, 'Ford');
INSERT INTO personel VALUES(567890123, 'Mehmet Ozturk', 'Ankara', 7000, 'Tofas');
INSERT INTO personel VALUES(456715012, 'Veli Sahin', 'Ankara', 4500, 'Ford');
INSERT INTO personel VALUES(123456710, 'Hatice Sahin', 'Bursa', 4500, 'Honda');
SELECT * FROM personel;
```

| ID        | ISIM          | SEHIR    | MAAS | SIRKET |
|-----------|---------------|----------|------|--------|
| 123456789 | Ali Yilmaz    | Istanbul | 5500 | Honda  |
| 234567890 | Veli Sahin    | Istanbul | 4500 | Toyota |
| 345678901 | Mehmet Ozturk | Ankara   | 3500 | Honda  |
| 456789012 | Mehmet Ozturk | Izmir    | 6000 | Ford   |
| 567890123 | Mehmet Ozturk | Ankara   | 7000 | Tofas  |
| 456715012 | Veli Sahin    | Ankara   | 4500 | Ford   |
| 123456710 | Hatice Sahin  | Bursa    | 4500 | Honda  |

```
SELECT *
FROM (SELECT isim, maas FROM personel)
PIVOT (COUNT (maas) FOR maas IN (4500, 3500, 5500, 6000, 7000));
```

| ISIM          | 4500 | 3500 | 5500 | 6000 | 7000 |
|---------------|------|------|------|------|------|
| Hatice Sahin  | 1    | 0    | 0    | 0    | 0    |
| Veli Sahin    | 2    | 0    | 0    | 0    | 0    |
| Ali Yilmaz    | 0    | 0    | 1    | 0    | 0    |
| Mehmet Ozturk | 0    | 1    | 0    | 1    | 1    |

```
SELECT *
FROM (SELECT isim, maas FROM personel)
PIVOT (SUM (maas) FOR isim IN ('Ali Yilmaz', 'Mehmet Ozturk'));
```

| 'Ali Yilmaz' | 'Mehmet Ozturk' |
|--------------|-----------------|
| 5500         | 16500           |

```
SELECT *
FROM (SELECT isim, maas FROM personel)
PIVOT(COUNT(isim) FOR isim IN ('Ali Yilmaz', 'Mehmet Ozturk', 'Hatice Sahin', 'Veli Sahin'));
```

| MAAS | 'Ali Yilmaz' | 'Mehmet Ozturk' | 'Hatice Sahin' | 'Veli Sahin' |
|------|--------------|-----------------|----------------|--------------|
| 4500 | 0            | 0               | 1              | 2            |
| 7000 | 0            | 1               | 0              | 0            |
| 6000 | 0            | 1               | 0              | 0            |
| 5500 | 1            | 0               | 0              | 0            |
| 3500 | 0            | 1               | 0              | 0            |

```
SELECT *
FROM (SELECT isim, sirket FROM personel)
PIVOT (COUNT (sirket) FOR sirket IN ('Honda', 'Ford'));
```

| ISIM          | 'Honda' | 'Ford' |
|---------------|---------|--------|
| Hatice Sahin  | 1       | 0      |
| Veli Sahin    | 0       | 1      |
| Ali Yilmaz    | 1       | 0      |
| Mehmet Ozturk | 1       | 1      |

```
SELECT *
FROM (SELECT isim, sirket FROM personel)
PIVOT (COUNT (isim) FOR isim IN ('Ali Yilmaz', 'Veli Sahin', 'Mehmet Ozturk'));
```

| SIRKET | 'Ali Yilmaz' | 'Veli Sahin' | 'Mehmet Ozturk' |
|--------|--------------|--------------|-----------------|
| Honda  | 1            | 0            | 1               |
| Ford   | 0            | 1            | 1               |
| Toyota | 0            | 1            | 0               |
| Tofas  | 0            | 0            | 1               |

- **ALTER TABLE** statement tabloda **add**, **modify** veya **drop/delete columns** işlemleri için kullanılır.
- **ALTER TABLE** statement **tabloları yeniden isimlendirmek** için de kullanılır.
- **ALTER TABLE** kısaca tabloları değiştirir.

```
-- Tabloya 1 sütun ulke_isim ekleyelim ve default olarak Türkiye yazsin
ALTER TABLE personel
```

```
ADD ulke_isim varchar2(30) DEFAULT 'Türkiye';
```

- Tabloya ADD ile bir sütun eklediğimizde DEFAULT yazarsak yazdığımız değer, tüm kayıtlara eklenir.

| ID        | ISIM          | SEHIR    | MAAS | SIRKET | ULKE_ISIM |
|-----------|---------------|----------|------|--------|-----------|
| 123456789 | Ali Yilmaz    | Istanbul | 5500 | Honda  | Türkiye   |
| 234567890 | Veli Sahin    | Istanbul | 4500 | Toyota | Türkiye   |
| 345678901 | Mehmet Ozturk | Ankara   | 3500 | Honda  | Türkiye   |
| 456789012 | Mehmet Ozturk | Izmir    | 6000 | Ford   | Türkiye   |
| 567890123 | Mehmet Ozturk | Ankara   | 7000 | Tofas  | Türkiye   |
| 456715012 | Veli Sahin    | Ankara   | 4500 | Ford   | Türkiye   |
| 123456710 | Hatice Sahin  | Bursa    | 4500 | Honda  | Türkiye   |

- Tabloya birden fazla sütun ekleme

```
-- yas ve tel ekleyelim
```

```
ALTER TABLE personel
```

```
ADD (yas NUMBER (2), tel_no char(11));
```

| ID        | ISIM          | SEHIR    | MAAS | SIRKET | ULKE_ISIM | YAS | TEL_NO |
|-----------|---------------|----------|------|--------|-----------|-----|--------|
| 123456789 | Ali Yilmaz    | Istanbul | 5500 | Honda  | Türkiye   | -   | -      |
| 234567890 | Veli Sahin    | Istanbul | 4500 | Toyota | Türkiye   | -   | -      |
| 345678901 | Mehmet Ozturk | Ankara   | 3500 | Honda  | Türkiye   | -   | -      |
| 456789012 | Mehmet Ozturk | Izmir    | 6000 | Ford   | Türkiye   | -   | -      |
| 567890123 | Mehmet Ozturk | Ankara   | 7000 | Tofas  | Türkiye   | -   | -      |
| 456715012 | Veli Sahin    | Ankara   | 4500 | Ford   | Türkiye   | -   | -      |
| 123456710 | Hatice Sahin  | Bursa    | 4500 | Honda  | Türkiye   | -   | -      |

```
-- cinsiyet ekleyelim, bos kalmasin ve tüm satirlara eklensin
```

```
ALTER TABLE personel
```

```
ADD cinsiyet varchar2(20) DEFAULT 'Seciminizi giriniz' NOT NULL;
```

| ID        | ISIM          | SEHIR    | MAAS | SIRKET | ULKE_ISIM | YAS | TEL_NO | CINSIYET           |
|-----------|---------------|----------|------|--------|-----------|-----|--------|--------------------|
| 123456789 | Ali Yilmaz    | Istanbul | 5500 | Honda  | Türkiye   | -   | -      | Seciminizi giriniz |
| 234567890 | Veli Sahin    | Istanbul | 4500 | Toyota | Türkiye   | -   | -      | Seciminizi giriniz |
| 345678901 | Mehmet Ozturk | Ankara   | 3500 | Honda  | Türkiye   | -   | -      | Seciminizi giriniz |
| 456789012 | Mehmet Ozturk | Izmir    | 6000 | Ford   | Türkiye   | -   | -      | Seciminizi giriniz |
| 567890123 | Mehmet Ozturk | Ankara   | 7000 | Tofas  | Türkiye   | -   | -      | Seciminizi giriniz |
| 456715012 | Veli Sahin    | Ankara   | 4500 | Ford   | Türkiye   | -   | -      | Seciminizi giriniz |
| 123456710 | Hatice Sahin  | Bursa    | 4500 | Honda  | Türkiye   | -   | -      | Seciminizi giriniz |

- **DROP COLUMN** ile tablodan sütun silme

```
-- tel_no sütununu silin
```

```
ALTER TABLE personel
```

```
DROP COLUMN tel_no;
```

| ID        | ISIM          | SEHIR    | MAAS | SIRKET | ULKE_ISIM | YAS | CINSIYET           |
|-----------|---------------|----------|------|--------|-----------|-----|--------------------|
| 123456789 | Ali Yilmaz    | Istanbul | 5500 | Honda  | Türkiye   | -   | Seciminizi giriniz |
| 234567890 | Veli Sahin    | Istanbul | 4500 | Toyota | Türkiye   | -   | Seciminizi giriniz |
| 345678901 | Mehmet Ozturk | Ankara   | 3500 | Honda  | Türkiye   | -   | Seciminizi giriniz |
| 456789012 | Mehmet Ozturk | Izmir    | 6000 | Ford   | Türkiye   | -   | Seciminizi giriniz |
| 567890123 | Mehmet Ozturk | Ankara   | 7000 | Tofas  | Türkiye   | -   | Seciminizi giriniz |
| 456715012 | Veli Sahin    | Ankara   | 4500 | Ford   | Türkiye   | -   | Seciminizi giriniz |
| 123456710 | Hatice Sahin  | Bursa    | 4500 | Honda  | Türkiye   | -   | Seciminizi giriniz |

- RENAME COLUMN TO ile yeniden adlandırma

```
-- ulke_isim sütununu ulke_adi yapalım
ALTER TABLE personel
RENAME COLUMN ulke_isim TO ulke_adi;
```

| ID        | ISIM          | SEHIR    | MAAS | SIRKET | ULKE_AD | YAS | CINSIYET           |
|-----------|---------------|----------|------|--------|---------|-----|--------------------|
| 123456789 | Ali Yilmaz    | Istanbul | 5500 | Honda  | Türkiye | -   | Seciminizi giriniz |
| 234567890 | Veli Sahin    | Istanbul | 4500 | Toyota | Türkiye | -   | Seciminizi giriniz |
| 345678901 | Mehmet Ozturk | Ankara   | 3500 | Honda  | Türkiye | -   | Seciminizi giriniz |
| 456789012 | Mehmet Ozturk | Izmir    | 6000 | Ford   | Türkiye | -   | Seciminizi giriniz |
| 567890123 | Mehmet Ozturk | Ankara   | 7000 | Tofas  | Türkiye | -   | Seciminizi giriniz |
| 456715012 | Veli Sahin    | Ankara   | 4500 | Ford   | Türkiye | -   | Seciminizi giriniz |
| 123456710 | Hatice Sahin  | Bursa    | 4500 | Honda  | Türkiye | -   | Seciminizi giriniz |

- RENAME TO ile tablo adını personel yerine calisanlar yapma

```
ALTER TABLE personel
RENAME TO calisanlar;
```

SELECT \* FROM personel; <= artık çalışmaz! (ORA-00942: table or view does not exist) Hatası alırsız, bunun yerine =>

SELECT \* FROM calisanlar; yazılmalı.

Table Name

PERSONEL

Table Name

CALISANLAR

- MODIFY ile sütun özelliklerini değiştirme

```
-- isim fieldini NOT NULL yapın
ALTER TABLE calisanlar
MODIFY isim varchar2(30) NOT NULL;
```

| Constraint      | Type        | Condition              |
|-----------------|-------------|------------------------|
| SYS_C0041407899 | Check       | "CINSIYET" IS NOT NULL |
| SYS_C0041408871 | Check       | "ISIM" IS NOT NULL     |
| PERSONEL_PK     | Primary Key | -                      |

Burada SQL ders konuları bitmiştir. Örnek çözmek isteyenler;

[www.sqlteaching.com](http://www.sqlteaching.com) veya

<https://sqlbolt.com/> gibi

Web sitelerinden yararlanabilir. Ayrıca sonraki sayfalarda da önemli interview soruları ve çözümleri yer almaktadır.

## Örnek Interview Soru ve Çözümleri

```
CREATE TABLE personel
(
id number(9),
isim varchar2(50),
sehir varchar2(50),
maas number(20),
sirket varchar2(20)
);
INSERT INTO personel VALUES(123456789, 'Johnny Walk', 'New Hampshire', 2500, 'IBM');
INSERT INTO personel VALUES(234567891, 'Brian Pitt', 'Florida', 1500, 'LINUX');
INSERT INTO personel VALUES(245678901, 'Eddie Murphy', 'Texas', 3000, 'WELLS FARGO');
INSERT INTO personel VALUES(456789012, 'Teddy Murphy', 'Virginia', 1000, 'GOOGLE');
INSERT INTO personel VALUES(567890124, 'Eddie Murphy', 'Massachuset', 7000, 'MICROSOFT');
INSERT INTO personel VALUES(456789012, 'Brad Pitt', 'Texas', 1500, 'TD BANK');
INSERT INTO personel VALUES(123456719, 'Adem Stone', 'New Jersey', 2500, 'IBM');
SELECT * FROM personel;
```

| ID        | ISIM         | SEHIR         | MAAS | SIRKET      |
|-----------|--------------|---------------|------|-------------|
| 123456789 | Johnny Walk  | New Hampshire | 2500 | IBM         |
| 234567891 | Brian Pitt   | Florida       | 1500 | LINUX       |
| 245678901 | Eddie Murphy | Texas         | 3000 | WELLS FARGO |
| 456789012 | Teddy Murphy | Virginia      | 1000 | GOOGLE      |
| 567890124 | Eddie Murphy | Massachuset   | 7000 | MICROSOFT   |
| 456789012 | Brad Pitt    | Texas         | 1500 | TD BANK     |
| 123456719 | Adem Stone   | New Jersey    | 2500 | IBM         |

```
CREATE TABLE isciler
(
id number(9),
isim varchar2(50),
sehir varchar2(50),
maas number(20),
sirket varchar2(20)
);
INSERT INTO isciler VALUES(123456789, 'John Walker', 'Florida', 2500, 'IBM');
INSERT INTO isciler VALUES(234567890, 'Brad Pitt', 'Florida', 1500, 'APPLE');
INSERT INTO isciler VALUES(345678901, 'Eddie Murphy', 'Texas', 3000, 'IBM');
INSERT INTO isciler VALUES(456789012, 'Eddie Murphy', 'Virginia', 1000, 'GOOGLE');
INSERT INTO isciler VALUES(567890123, 'Eddie Murphy', 'Texas', 7000, 'MICROSOFT');
INSERT INTO isciler VALUES(456789012, 'Brad Pitt', 'Texas', 1500, 'GOOGLE');
INSERT INTO isciler VALUES(123456710, 'Mark Stone', 'Pennsylvania', 2500, 'IBM');
SELECT * FROM isciler;
```

| ID        | ISIM         | SEHIR        | MAAS | SIRKET    |
|-----------|--------------|--------------|------|-----------|
| 123456789 | John Walker  | Florida      | 2500 | IBM       |
| 234567890 | Brad Pitt    | Florida      | 1500 | APPLE     |
| 345678901 | Eddie Murphy | Texas        | 3000 | IBM       |
| 456789012 | Eddie Murphy | Virginia     | 1000 | GOOGLE    |
| 567890123 | Eddie Murphy | Texas        | 7000 | MICROSOFT |
| 456789012 | Brad Pitt    | Texas        | 1500 | GOOGLE    |
| 123456710 | Mark Stone   | Pennsylvania | 2500 | IBM       |

-- 1) Her iki tablodaki ortak id ve isim e sahip kayıtları listeleyen Query yazınız.

```
SELECT id, isim
FROM personel
INTERSECT
SELECT id, isim
FROM isciler;
```

| ID        | ISIM      |
|-----------|-----------|
| 456789012 | Brad Pitt |

-- 2) Her iki tablodaki ortak id 'leri ve personel tablosunda bu id 'ye sahip isimleri listeleyen Query yazınız.

```
SELECT id, isim
FROM personel
WHERE id IN (SELECT id
 FROM isciler
 WHERE personel.id = isciler.id);
```

| ID        | ISIM         |
|-----------|--------------|
| 123456789 | Johnny Walk  |
| 456789012 | Teddy Murphy |
| 456789012 | Brad Pitt    |

-- 3) Personel tablosunda kaç farklı şehirden personel var?

```
SELECT COUNT (DISTINCT sehir) AS farkli_sehir_sayisi
FROM personel;
```

| FARKLI_SEHIR_SAYISI |
|---------------------|
| 6                   |

-- 4) Personel tablosunda id 'si çift sayı olan personel 'in tüm bilgilerini listeleyen Query yazınız.

```
SELECT *
FROM personel
WHERE MOD (id,2) = 0;
```

| ID        | ISIM         | SEHIR       | MAAS | SIRKET    |
|-----------|--------------|-------------|------|-----------|
| 456789012 | Teddy Murphy | Virginia    | 1000 | GOOGLE    |
| 567890124 | Eddie Murphy | Massachuset | 7000 | MICROSOFT |
| 456789012 | Brad Pitt    | Texas       | 1500 | TD BANK   |

-- 5) Personel tablosunda kaç tane kayıt olduğunu gösteren Query yazın.

```
SELECT COUNT (id) AS kayit_sayisi
FROM personel;
yada
SELECT COUNT (*) AS kayit_sayisi
FROM personel;
```

| KAYIT_SAYISI |
|--------------|
| 7            |

-- 6) İşçiler tablosunda en yüksek maaşı alan kişinin tüm bilgilerini gösteren Query yazın.

- İlk önce en yüksek maaşı bulalım. Sonra bulduğumuz Query 'i Subquery olarak kullanırız.

```
SELECT MAX (maas) AS max_maas
FROM isciler;
```

| MAX_MAAS |
|----------|
| 7000     |

```
SELECT *
FROM isciler
WHERE maas IN (SELECT MAX (maas)
 FROM isciler);
```

| ID        | ISIM         | SEHIR | MAAS | SIRKET    |
|-----------|--------------|-------|------|-----------|
| 567890123 | Eddie Murphy | Texas | 7000 | MICROSOFT |

-- 2.yol büyükten küçüğe sıralayıp en üstteki kaydı almak

```
SELECT *
FROM isciler
ORDER BY maas DESC
FETCH NEXT 1 ROW ONLY;
```

| ID        | ISIM         | SEHIR | MAAS | SIRKET    |
|-----------|--------------|-------|------|-----------|
| 567890123 | Eddie Murphy | Texas | 7000 | MICROSOFT |

-- 7) Personel tablosunda en düşük maaşı alan kişinin tüm bilgilerini gösteren Query yazın.

```
SELECT MIN (maas)
FROM personel;
```

```
SELECT *
FROM personel
WHERE maas IN (SELECT MIN (maas)
 FROM personel);
```

| ID        | ISIM         | SEHIR    | MAAS | SIRKET |
|-----------|--------------|----------|------|--------|
| 456789012 | Teddy Murphy | Virginia | 1000 | GOOGLE |

-- 2.yöntemle en az maaş alan 3 kişinin tüm bilgilerini yazdıran Query.

```
SELECT *
FROM personel
ORDER BY maas ASC
FETCH NEXT 3 ROW ONLY;
```

| ID        | ISIM         | SEHIR    | MAAS | SIRKET  |
|-----------|--------------|----------|------|---------|
| 456789012 | Teddy Murphy | Virginia | 1000 | GOOGLE  |
| 234567891 | Brian Pitt   | Florida  | 1500 | LINUX   |
| 456789012 | Brad Pitt    | Texas    | 1500 | TD BANK |

-- 8) İşçiler tablosunda ikinci en yüksek maaşı alan personelin tüm bilgisini gösteren Query yazın.

```
SELECT *
FROM isciler
ORDER BY maas DESC
OFFSET 1 ROW
FETCH NEXT 1 ROW ONLY;
```

| ID        | ISIM         | SEHIR | MAAS | SIRKET |
|-----------|--------------|-------|------|--------|
| 345678901 | Eddie Murphy | Texas | 3000 | IBM    |

- 2.yöntemle sıralamadaki 2.işçiyi görüntüleyebiliriz ama sorun olabilir çünkü en yüksek maaşa sahip 2 kişi varsa bu sorgu en yüksek maaşlı da olsa listeden 2.yi getirir.

```
SELECT MAX (maas)
FROM isciler
WHERE maas <> 7000;
```

| MAX(MAAS) |
|-----------|
| 3000      |

```
-- SUBQUERY ile çözme
SELECT *
FROM isciler
WHERE maas IN (SELECT MAX (maas)
 FROM isciler
 WHERE maas <> 7000);
```

| ID        | ISIM         | SEHIR | MAAS | SIRKET |
|-----------|--------------|-------|------|--------|
| 345678901 | Eddie Murphy | Texas | 3000 | IBM    |

-- ama soruyu çözerken 7000'i kullanmamalıyız.

```
-- A) En yüksek maasi veren sorgu
SELECT MAX (maas)
FROM isciler;
```

| MAX(MAAS) |
|-----------|
| 7000      |

```
-- B) En yüksek maaşa eşit olmayanlar içindeki en yüksek maaşı (2.en yüksek maaş)
-- veren sorgu.
SELECT MAX (maas)
FROM isciler
WHERE maas <> (SELECT MAX (maas)
 FROM isciler);
```

| MAX(MAAS) |
|-----------|
| 3000      |

```
-- İkinci en yüksek maaşı alan kişinin tüm bilgilerini veren Query yazın.
SELECT *
FROM isciler
WHERE maas = (SELECT MAX (maas)
 FROM isciler
 WHERE maas <> (SELECT MAX (maas)
 FROM isciler));
```

| ID        | ISIM         | SEHIR | MAAS | SIRKET |
|-----------|--------------|-------|------|--------|
| 345678901 | Eddie Murphy | Texas | 3000 | IBM    |

```
-- 9) İşçiler tablosunda ikinci en düşük maaşı alan işçinin tüm bilgilerini gösteren Query yazın.
SELECT *
FROM isciler
ORDER BY maas
OFFSET 1 ROW
FETCH NEXT 1 ROW ONLY;
```

| ID        | ISIM      | SEHIR   | MAAS | SIRKET |
|-----------|-----------|---------|------|--------|
| 234567890 | Brad Pitt | Florida | 1500 | APPLE  |

```
-- 10) İşçiler tablosunda en yüksek maaşı alan işçinin dışındaki tüm işçilerin, tüm bilgilerini gösteren
Query yazın.
```

```
SELECT *
FROM isciler
WHERE maas <> (SELECT MAX (maas)
 FROM isciler)
ORDER BY maas DESC;
```

| ID        | ISIM         | SEHIR        | MAAS | SIRKET |
|-----------|--------------|--------------|------|--------|
| 345678901 | Eddie Murphy | Texas        | 3000 | IBM    |
| 123456710 | Mark Stone   | Pennsylvania | 2500 | IBM    |
| 123456789 | John Walker  | Florida      | 2500 | IBM    |
| 234567890 | Brad Pitt    | Florida      | 1500 | APPLE  |
| 456789012 | Brad Pitt    | Texas        | 1500 | GOOGLE |
| 456789012 | Eddie Murphy | Virginia     | 1000 | GOOGLE |