



BATCH : B 44-45-46-47  
LESSON : SQL  
DATE : 30.12.2021  
SUBJECT : Introduction

-  techproeducation
-  techproeducation
-  techproeducation
-  techproeducation
-  techproedu

# **SQL**

## **Structured Query Language**

### **Yapılandırılmış Sorgu Dili**



**iPEK BİLGİN**

**Java Full Stack Developer**



## Bugünkü Konumuz

- 1) Database Nedir ?
- 2) Genel Database Kavramları
- 3) SQL'e giriş



# database

**Kaydol**

Hızlı ve kolaydır.

Adın  Soyadın

Cep telefonu numarası veya e-posta

Yeni şifre

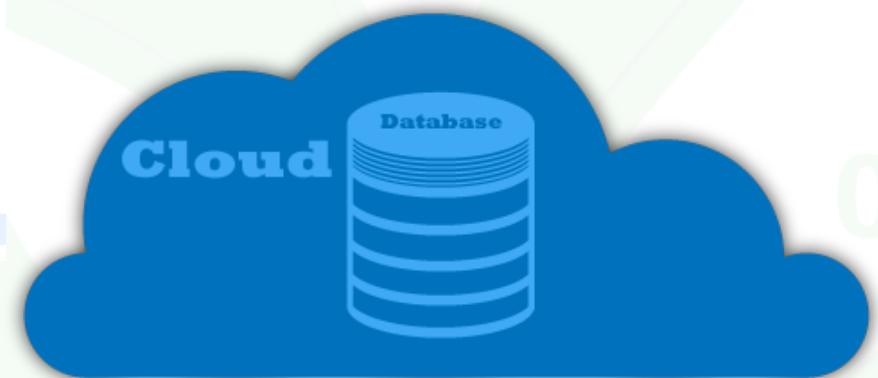
Doğum Tarihi  1 Eki 2020

Cinsiyet  Kadın  Erkek  Özel

Kaydol düğmesine tıklayarak, Koşullarımıza, Veri İlkemizi ve Çerezler İlkemizi kabul etmiş olursun. Bizden SMS Bildirimleri alabilir ve bu bildirimleri istediği zaman durdurabilirsin.

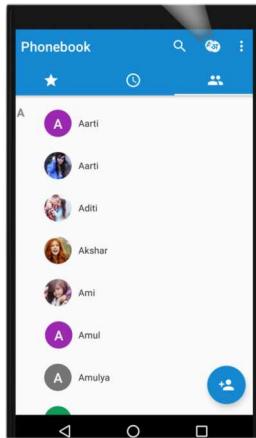
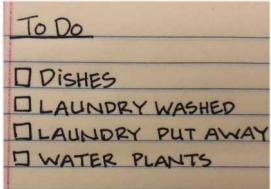
**Kaydol**

```
public class facebook {  
    public static void main(String[] args) {  
        Scanner scan = new Scanner(System.in);  
        System.out.println("Enter your name");  
        String name = scan.nextLine();  
  
        System.out.println("Enter your surname");  
        String surname = scan.nextLine();  
  
        System.out.println("Enter your email");  
        String email = scan.nextLine();  
  
        System.out.println("Enter your password");  
        String password = scan.nextLine();  
  
        scan.close();  
    }  
}
```





# Database (VERİTABANI) nedir?



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

Veritabanı genellikle bir Veritabanı Yönetim Sistemi DBMS (**DataBase Management System**) ile kontrol edilir.

Çoğu veritabanında veri yazma ve sorgulama için yapılandırılmış sorgu dili SQL (**Structured Query Language**) kullanılır.



# Database'in faydalari nelerdir

- 1) Yuksek miktarda bilgi depolanabilir
- 2) Olusturma, Okuma,Degistirme ve Silme kolayligi

**Create,Read,Update,Delete (CRUD)**

- 3) Girisin kolay ve kontrollu olmasi
- 4) Dataya ulasim kolayligi
- 5) Guvenlik

ono	adi	soyadi	dyeri	bid
1	Ali	Turan	İstanbul	1
2	Ahmet	Büyük	Ankara	1
3	Leyla	Şahin	İzmir	1
4	Can	Türkoğlu	Manisa	2
5	Aziz	Keskin	İstanbul	2
6	Talat	Şanlı	İzmir	3
7	Kamuran	Kece	Adana	3
8	Turgut	Cemal	Bursa	4





# Database Validation(dogrulama) Testi

## Kaydol

Hızlı ve kolaydır.

Adın  Soyadın

Cep telefonu numarası veya e-posta

Yeni şifre

Doğum Tarihi  Eki  2020

Cinsiyet  Kadın  Erkek  Özel

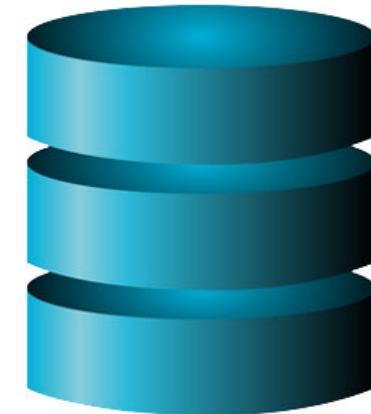
Kaydol düğmesine tıklayarak, Koşullarımıza, Veri İlkemizi ve Çerezler İlkemizi kabul etmiş olursun. Bizden SMS Bildirimleri alabilir ve bu bildirimleri istediği zaman durdurabilirsin.

**Kaydol**

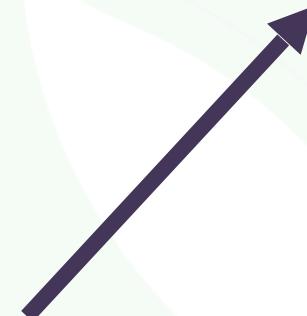
User Interface



**API**



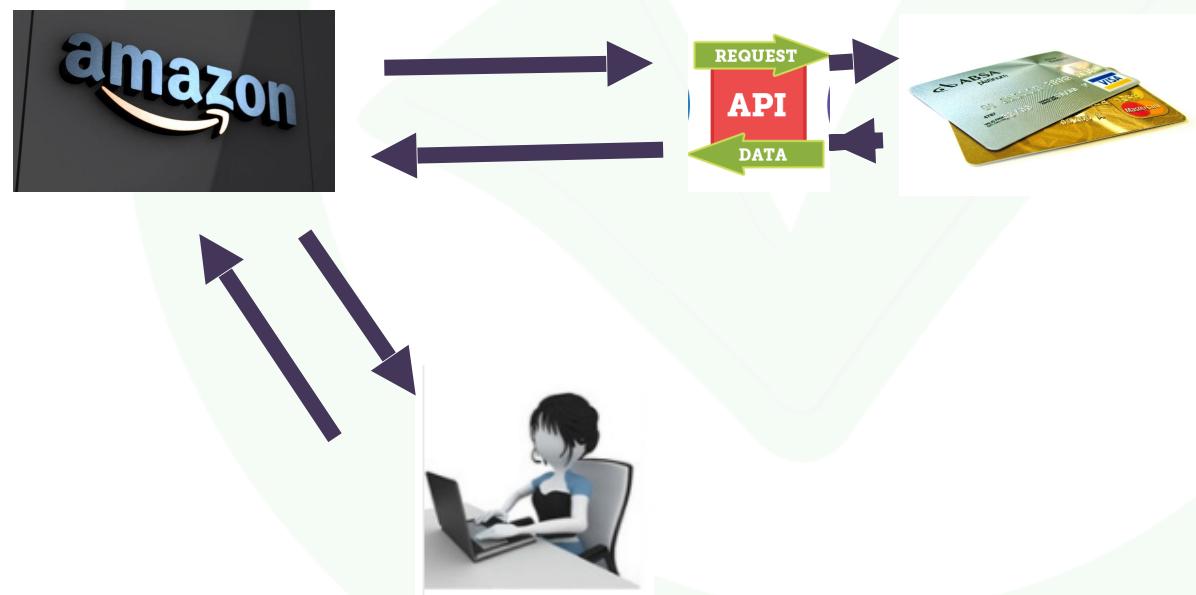
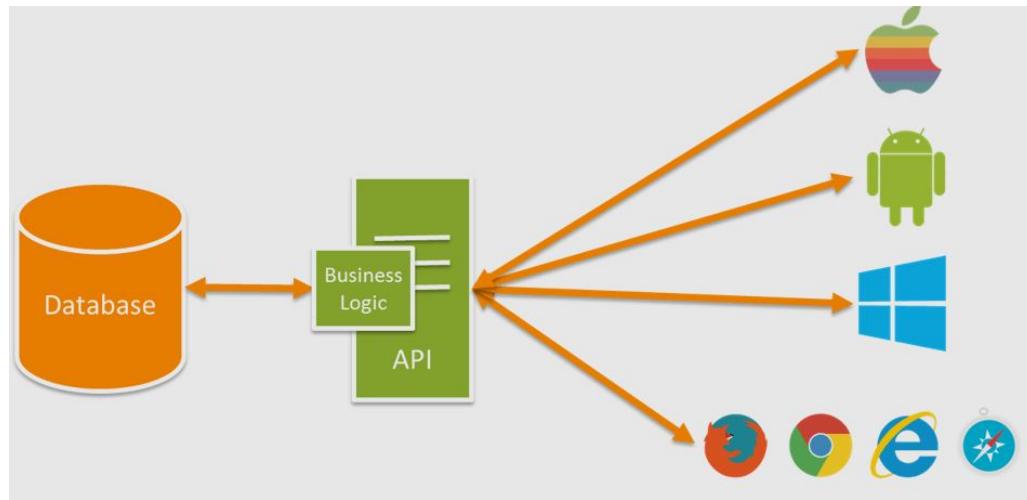
**Database**





# API

Application Programming Interface, bir uygulamaya ait yeteneklerin, başka bir uygulamada da kullanılabilmesi için, yeteneklerini paylaşan uygulamanın sağladığı arayüzdür.





## END To END (E2E) Testing

- 1 Eger datayi User Interface (UI) kullanarak yolladiysaniz
  - A) Datayi UI dan arama fonksiyonunu kullanarak doğrula (Selenium)
  - B) Datayi SQL kodlarini kullanarak doğrula (SQL + Selenium)
  - C) Datayi API kodlarini kullanarak doğrula (API + Selenium)
- 2) Eger datayi SQL kodlarini kullanarak yolladiysaniz
  - A) Datayi UI dan arama fonksiyonunu kullanarak doğrula (Selenium)
  - B) Datayi SQL kodlarini kullanarak doğrula (SQL + Selenium)
  - C) Datayi API kodlarini kullanarak doğrula (API + Selenium)



# DATA SCIENCE İÇİN MYSQL

Her datascience adayının edinmesi gereken en kolay ve temel beceri SQL'dir. Günümüzde çoğu şirket veri odaklı olmaya doğru gidiyor. Bu veriler bir veritabanında saklanır ve bir Veritabanı Yönetim sistemi aracılığıyla yönetilir ve işlenir. DBMS, işimizi çok kolay ve düzenli hale getiriyor. Bu nedenle, en popüler programlama dilini inanılmaz DBMS aracıyla entegre etmek çok önemlidir.

SQL, veritabanlarıyla çalışırken en yaygın kullanılan programlama dilidir ve MySQL, SQL Server ve Oracle gibi çeşitli ilişkisel veritabanı sistemleri tarafından desteklenir. Ancak, SQL standardının farklı veritabanı sistemlerinde farklı şekilde uygulanan bazı özellikleri vardır. Böylece SQL, bu Veri Bilimi alanında öğrenilmesi gereken en önemli kavramlardan biri haline gelir.

## Data Science da SQL İhtiyacı

**SQL (Structured Query Language), veritabanlarında depolanan veriler üzerinde kayıt güncelleme, kayıt silme, tablo, görünüm oluşturma ve değiştirme gibi çeşitli işlemleri gerçekleştirmek için kullanılır. SQL aynı zamanda SQL kullanan mevcut büyük veri platformları için de standarttır. ilişkisel veritabanları için anahtar API'leri.**

Veri Bilimi, verilerin kapsamlı bir şekilde incelenmesidir. Verilerle çalışmak için onu veritabanından çıkarmamız gereklidir. SQL'in resme girdiği yer burasıdır. İlişkisel Veritabanı Yönetimi, Veri Biliminin çok önemli bir parçasıdır. Bir Veri Bilimcisi, SQL komutlarını kullanarak veritabanını kontrol edebilir, tanımlayabilir, değiştirebilir, oluşturabilir ve sorgulayabilir.



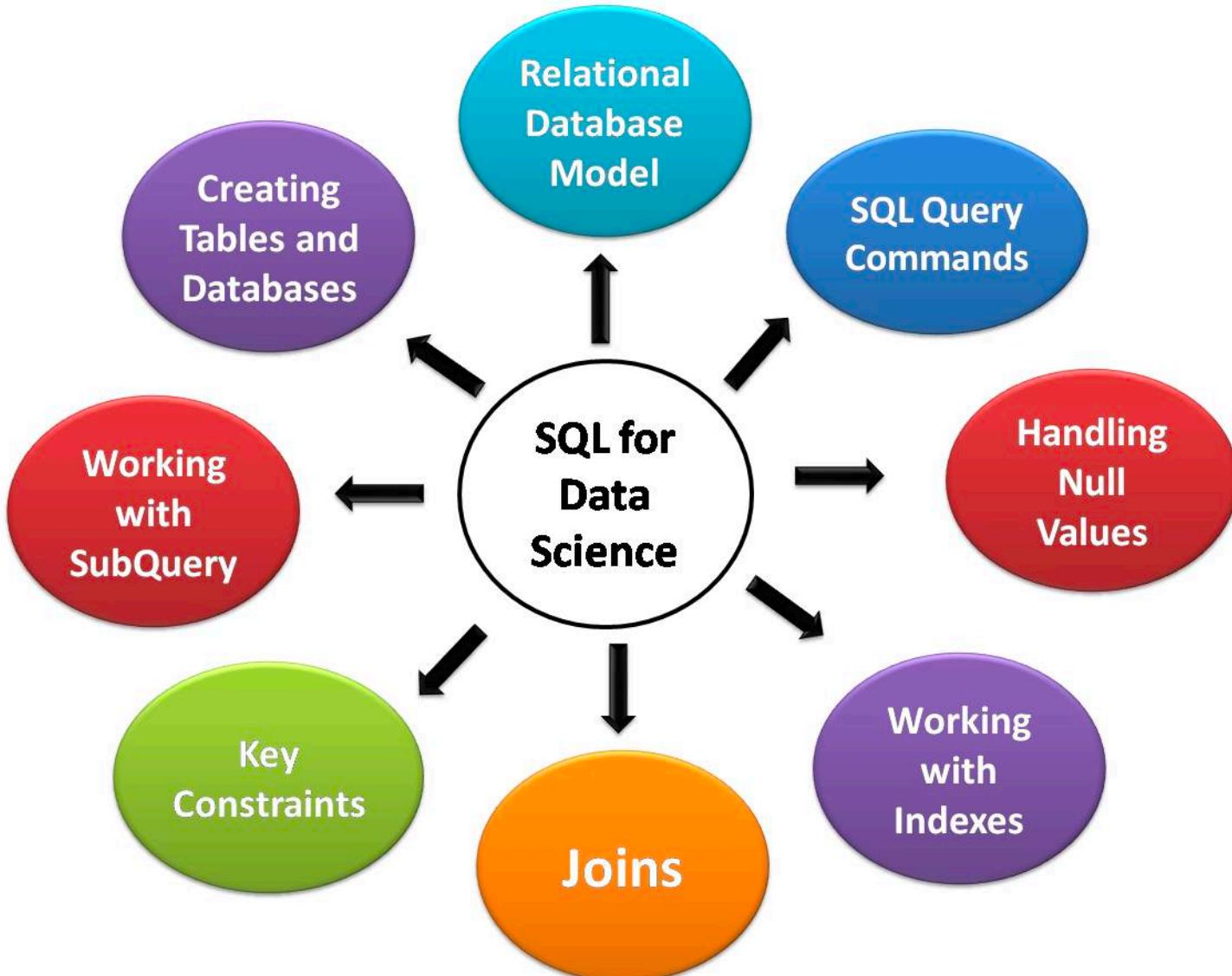
# DATA SCIENCE İÇİN MYSQL

Birçok modern endüstri, ürün veri yönetimini NoSQL teknolojisi ile donatmıştır ancak SQL, birçok iş zekası aracı ve ofis içi operasyonlar için ideal seçim olmaya devam etmektedir.

Veritabanı platformlarının çoğu SQL'den sonra modellenmiştir. Bu nedenle birçok veritabanı sistemi için bir standart haline gelmiştir. Hadoop, Spark gibi modern büyük veri sistemleri de SQL'i yalnızca ilişkisel veritabanı sistemlerini korumak ve yapılandırılmış verileri işlemek için kullanır.

## ÖZETLE:

1. Bir Veri Bilimcisi, yapılandırılmış verileri işlemek için SQL'e ihtiyaç duyar. Yapılandırılmış veriler ilişkisel veritabanlarında depolandığından. Bu nedenle, bu veritabanlarını sorgulamak için bir veri bilimcisi, SQL komutları hakkında iyi bir bilgiye sahip olmalıdır.
2. Hadoop ve Spark gibi Büyük Veri Platformları, işlemek için SQL komutlarını kullanarak sorgulama için bir uzanti sağlar.
3. SQL, test ortamlarının oluşturulması yoluyla verilerle deneme yapmak için standart araçtır.
4. Oracle, Microsoft SQL, MySQL gibi ilişkisel veritabanlarında saklanan verilerle analitik işlemleri gerçekleştirmek için SQL'e ihtiyacımız var.
5. SQL ayrıca veri karıştırma ve hazırlama için önemli bir araçtır. Bu nedenle, çeşitli Büyük Veri araçlarıyla uğraşırken SQL'den yararlanıyoruz.

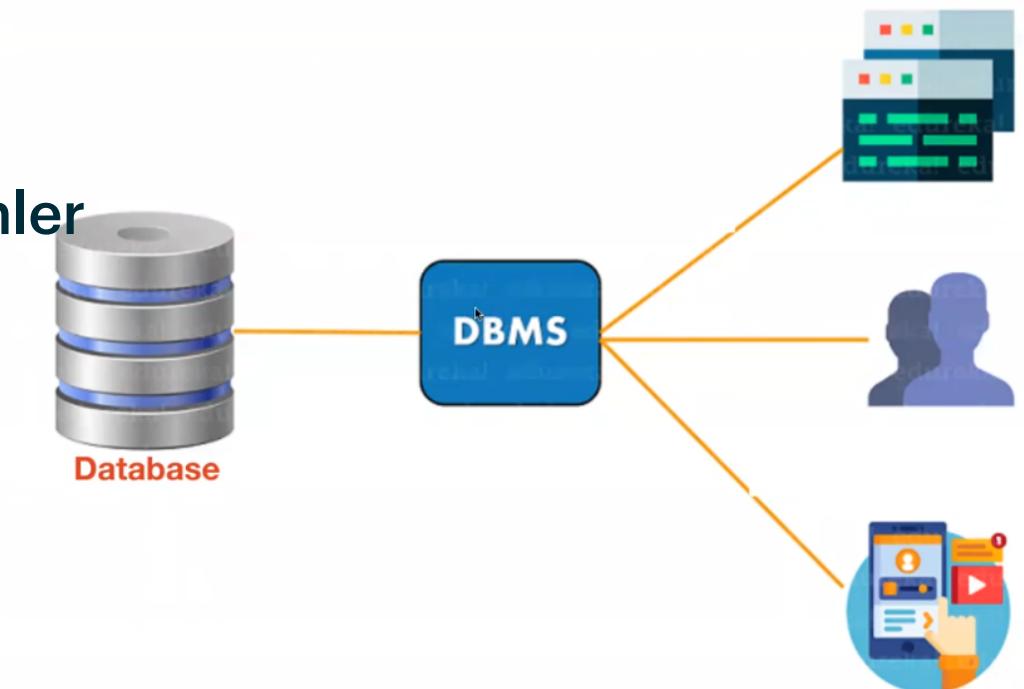




# Data Base Management System (DBMS)

Veritabanlarını yönetmek, kullanmak, geliştirmek ve bakımını yapmak için kullanılan yazılımlara denir.

- Database'e erisimi düzenler
- **Create, Read, Update, Delete** işlemlerini düzenler
- Data güvenliğini sağlar
- **Formlar oluşturur ve formları işler,**
- **Sorgular oluşturur ve sorguları iletilir,**
- **Raporlar oluşturur ve raporları işletir,**
- Uygulamayı kontrol eder
- Diger uygulamalarla (Application) iletisimi saglar.





# Tablolar (tables)

Headers====>

Row (Record) ===>

Row (Record) ===>

Row (Record) ===>

Row (Record) ===>

contactID	name	company	email
1	Bill Gates	Microsoft	bill@XBoxOneRocks.com
2	Steve Jobs	Apple	steve@rememberNewton.com
3	Linus Torvalds	Linux Foundation	linus@gnuWho.org
4	Andy Harris	Wiley Press	andy@aharrisBooks.net

Column (Field) ==>

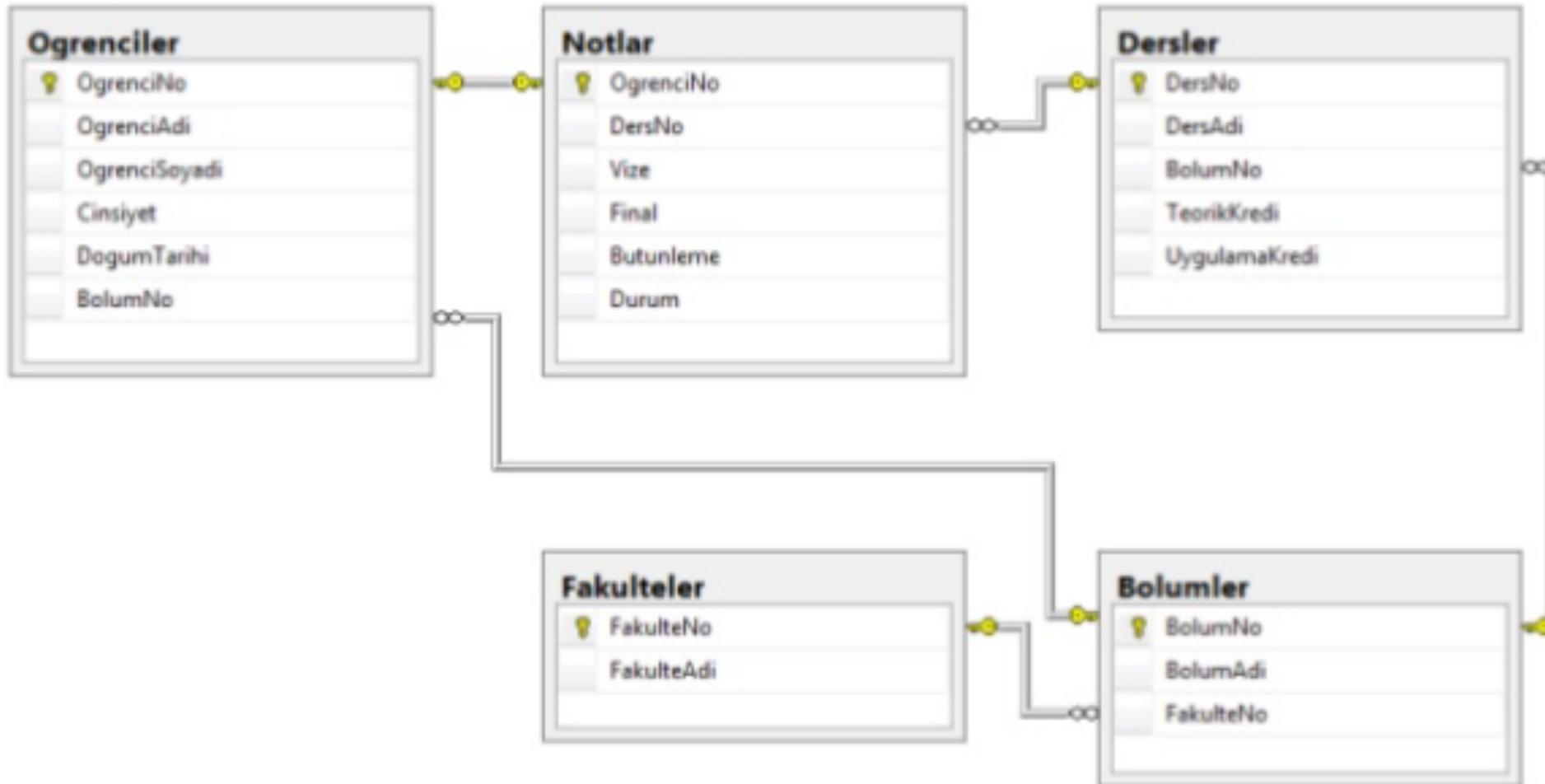
Column (Field) ==>

Column (Field) ==>

Column (Field) ==>



# Relatioanal databases (iliskili tablolar)





# Relational databases (iliskili tablolar)

➤ **SQL tablolar** dataları ilişkili tablolarda depolar.

➤ Tablolar arası ilişkiler net olmalıdır.

➤ Tablolar arası geçiş kolay olmalıdır

➤ Tabloların ve ilişkilerin bütününe SCHEMA denir

<b>id</b>	<b>ogrenci_adi</b>	<b>ogrenci_soyadi</b>
1	Elif	Türkmen
2	Ayşe	Sarı
3	Ender	Kaya
4	Ali	Demir
5	Adem	Salih

<b>id</b>	<b>ogrenci_id</b>	<b>ders_id</b>
1	1	3
2	1	5
3	2	1
4	3	4
5	4	2
6	4	3

<b>id</b>	<b>ders_adi</b>
1	Matematik
2	Tarih
3	Edebiyat
4	Yazılım
5	İstatistik

➤ **Relational Databases, SQL Databases (Structured Query Language) olarak da adlandırılır**



# Çok Kullanılan Relational Databases(SQL Database)



**SQL Server** : Microsoft tarafından gelistirilmistir

**Negatif:** Pahali – Kurumsal Kullanicilar icin binlerce dolar odenmesi gereklidir

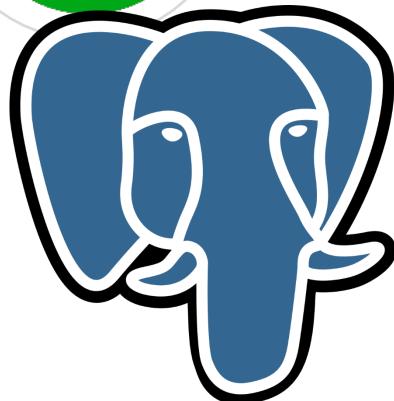
**Pozitif:** Zengin bir **user interface**'e sahip ve çok büyük dataların kullanilmasında sorunsuz calisir.

**Negatif:** Eszmalanma fazla istemigi gibi calismayi engelabilir.

**Pozitif:** Acik kaynak. Online destek ve ucretsiz çok fazla dokuman var



# Çok Kullanılan Relational Databases(SQL Database)



**PostgreSQL Server** : Created by a computer science professor Michael Stonebraker.

**Negatif:** Kurulum ve ayarlar zor. Yeni baslayanlar icin kullanimi zor

**Pozitif:** Yeni nesil olarak ortaya ciktig. Kisisellestirme mumkundur, zor gorevler icin ideal olabilir.

**ORACLE®**  
DATABASE



**PL/SQL** Oracle database sunuculari icinde depolanir

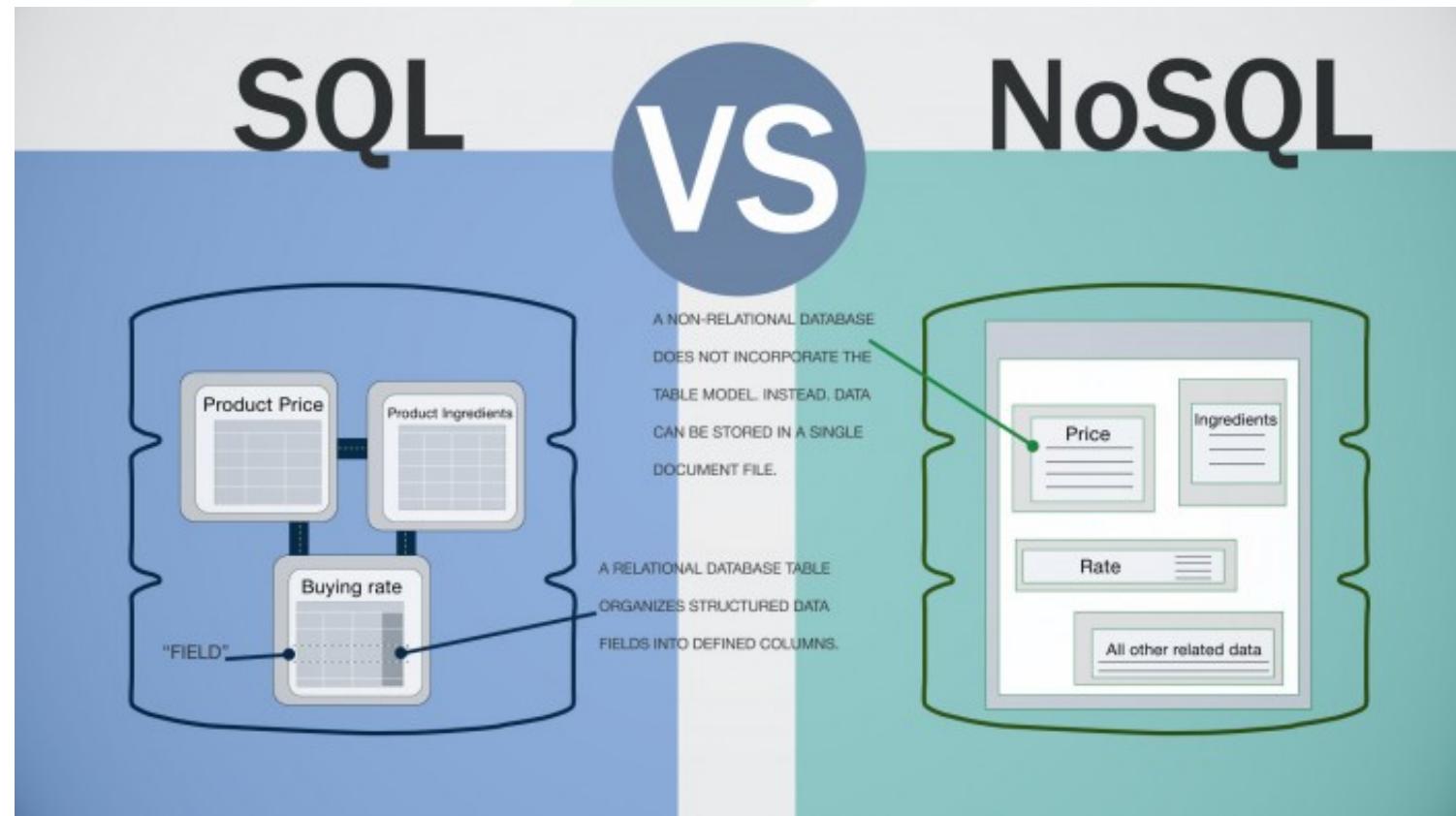
**PL/SQL** SQL komutlarini ozellikle karsilamak uzere dizayn edilmistir.

**Pros:** PL/SQL yüksek güvenlik seviyesi sağlar ve Object-Oriented Programming'e uyumludur



# Non Relational Databases(non-SQL Database)

SQL veritabanı verilerle ilgilenirken Yapısal Sorğu Dili kullanır. Veri yapısını belirlemek için önceden tanımlanmış şemalar gerektir.



NoSQL veritabanı verilerle çalışırken Yapılandırılmamış Sorğu Dili kullanır.



# SQL Komutları

**SQL komutları 4 ana gruba ayrılır:**

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

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

**SELECT : Veritabanındaki verileri alır.**

2. 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.

**INSERT : Veritabanına yeni veri ekler.**

**UPDATE : Veritabanındaki verileri günceller.**

**DELETE : Veritabanındaki verileri siler.**



# SQL Komutları

## 3. Veri Tanımlama Dili (Data Definition Language - DDL)

DDL komutları ile veritabanı ve tabloları oluşturma, değiştirme ve silme işlemleri yapılır:

**CREATE :** Bir veritabanı veya veritabanı içinde tablo oluşturur.

**ALTER :** Bir veritabanı veya veritabanı içindeki tabloyu günceller.

**DROP :** Bir veritabanını veya veritabanı içindeki tabloyu siler.

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

DCL komutları ile kullanıcılaraya veritabanı ve tablolar için yetki verilir veya geri alınır:

**GRANT :** Bir kullanıcıya yetki vermek için kullanılır.

**REVOKE :** Bir kullanıcıya verilen yetkiyi geri almak için kullanılır.



# Primary Key

Primary Keys



StudentId	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

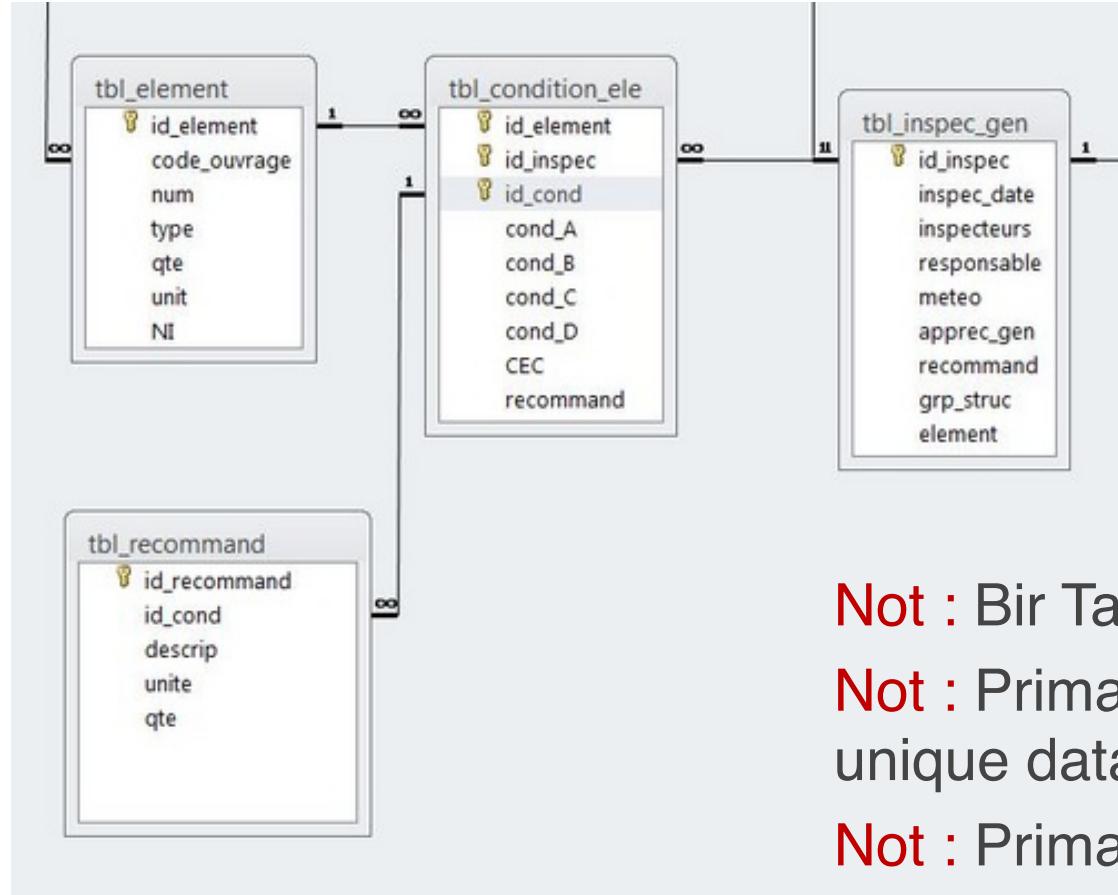
**Primary Key (birincil anahtar)**, bir veri tablosunda yer alan her satır için bir vekil / tanımlayıcı (identify) görevi görür, kısıtlamadır (constraint) ve eşsizdir (Unique).

Satırlara ait değerlerin karışmaması adına bu alana ait bilginin tekrarlanması gereklidir.

Çoğunlukla tek bir alan (`id`, `user_id`, `e_mail`, `username`, `national_identification_number` vb.) olarak kullanılsa da birden fazla alanın birleşimiyle de oluşturulabilir



# Primary Key



Primary Key değeri boş geçilemez ve NULL değer alamaz.

**Relational veri tabanlarında (relational database management system) mutlaka birincil anahtar olmalıdır.**

**Not :** Bir Tabloda yalnızca 1 tane primary Key olabilir.

**Not :** Primary Key benzersiz (Unique) olmalıdır ama her unique data Primary Key degildir

**Not :** Primary key her türlü datayı içerebilir. Sayı, String..

**Not :** Her tabloda Primary Key olması zorunlu degildir



# Primary Key

StudentID	FirstName	LastName
10 ←	John	Walker
11	Tom	Hanks
12	Kevin	Star
13 ←	Carl	Wall
14	Andrei	Apazniak
15	Mark	High
16	Clara	Star
17	John	Ocean
18 ←	John	Walker
19	Pamela	Star
20 ←	Carl	Wall

Genel olarak kayıt eklenmeden önce üretilen sıra numarası gibi sayısal değerlere **Surrogate Key** denir

Primary Key, dış dünyadaki gerçek verileri temsil ediyorsa, orneğin; TC kimlik numarası, bir kitabın ISBN numarası, bir ürünün ismi, email hesabı gibi buna **Natural key** denir

Email	FirstName	LastName
JWalker@gmail.com	John	Walker
THanks@gmail.com	Tom	Hanks
KStar@gmail.com	Kevin	Star
CWall@gmail.com	Carl	Wall
AApazniak@gmail.com	Andrei	Apazniak
MHigh@gmail.com	Mark	High
CStar@gmail.com	Clara	Star
JOcean@gmail.com	John	Ocean
JWalker01@gmail.com	John	Walker
PStar@gmail.com	Pamela	Star
CWall01@gmail.com	Carl	Wall



# Foreign Key

Foreign Key iki tablo arasında relation olusturmak icin kullanılır

Foreign Key baska bir tablodaki Primary Key ile ilişkilendirilmiş olmalıdır

StudentID	FirstName	LastNam	CourseID
10	John	Walker	200
11	Tom	Hanks	400
12	Kevin	Star	400
13	Carl	Wall	200
14	Andrei	Apazniak	300
15	Mark	High	400
16	Clara	Star	100
17	John	Ocean	100
18	John	Walker	200
19	Pamela	Star	300
20	Carl	Wall	NULL

Child Table

Foreign Key

Primary Key

CourseID	CourseName	CourseCredit	CourseFee
100	Biology	3	1200
200	Math	3	1200
300	English	2	600
400	Selective	1	200

Parent Table

Bir Tabloda birden fazla Foreign Key olabilir  
Forein Key **NULL** degeri Kabul eder  
Foreign Key olarak tanımlanan field'da **tekrarlar** olabilir

Foreign Key, değerleri farklı bir tablodaki Primary Key ile eşleşen bir sütun veya sütunların birleşimidir.



**Note:** Foreign key Tablonun kendi içinde bir relation oluşturabilir.

Emp_ID	first_name	last_name	birth_date	Gender	salary	Job_ID	Manager_ID
100	Jan	Levinson	1961-05-11	F	110,000	1	NULL
101	Michael	Scott	1964-03-15	M	75,000	2	100
102	Josh	Porter	1969-09-05	M	78,000	3	100
103	Angela	Martin	1971-06-25	F	63,000	2	101
104	Andy	Bernard	1973-07-22	M	65,000	3	101

Job_ID	Job_Name
2	SDET
3	Manual Tester
1	QE Lead

- 1) Michael Scott'un yoneticisi kimdir?
  - 2) Angela Martin'in Job\_Name'i nedir ?
  - 3) Manual Tester'larin ortalama Salary'si ne kadardir ?
  - 4) En yuksek Salary'yi alan kisinin Job\_Name'i nedir?



# **SQL**

## **Structured Query Language**

## **Yapılandırılmış Sorgu Dili**

### **Data Tipleri**

### **Tablo Olusturma**

**iPEK BİLGİN**

**Java Full Stack Developer**



# SQL Composite Key

Job_ID	Job_Name
2	SDET
3	Manual Tester
1	QE Lead

Job Table

Recruiter	NumberOfClient
Mark Eye	121
John Ted	283
Cory AI	67
Angela Star	301

Recruiter Table

Composite Key birden fazla field(kolon)'in kombinasyonu ile olusturulur.

Tek basina bir kolon Primary Key olma ozelliklerini tasimiyorsa, bu ozellikleri elde etmek icin birden fazla kolon birlestirilerek Primary olusturulur

JobJD	Recruiter 1	Recruiter 2	Company Table
2	Mark Eye	John Ted	RCG
1	John Ted	Mark Eye	RCG
1	Mark Eye	John Ted	Signature Inf
1	John Ted	Mark Eye	Long Inf
2	Cory AI	Angela Star	Signature

Company Table



# “UNIQUE KEY” & “PRIMARY KEY”

## “UNIQUE KEY” ve “PRIMARY KEY” arasındaki farklar

### Primary Key

Bir Tabloda sadece 1 tane olur  
NULL deger Kabul etmez

### Unique Key

Bir tabloda birden fazla olabilir  
Sadece 1 tane NULL degeri Kabul eder

## “UNIQUE KEY” ve “PRIMARY KEY” ortak ozellikleri

Dublication(Cift Kullanim)'a izin vermez



# Örnek Okul Tablosunun Bir Parçası

sinif tablosu		
sinif id	sinif	sube adi
9a	9 a	
9b	9 b	
9c	9 c	
9d	9 d	
10a	10 a	
10b	10 b	
10c	10 c	

ders tablosu	
ders id	ders adi
k10	10.sinif kimya
k11	11.sinif kimya
k12	12.sinif kimya
b10	10.sinif biyoloji
k9	9.sinif kimya
b9	9.sinif biyoloji

ogrenci tablosu				
ogrenci no	adi	soyadi	giris yili	sinif id
111	ali	velioglu	2020	9a
112	ayse	atakul	2018	9a
113	hasan	delioglan	2019	9a
114	hulya	kar	2019	9b
115	ali	yasa	2019	9b
116	ayse	atakul	2020	9b
117	kemal	velioglu	2018	10a
118	hatice	gulsen	2019	10b
119	hasan	delioglan	2019	10c
120	kemal	kar	2018	10c

ogretmen tablosu			
adi	soyadi	ders	ogr id
ahmet	baba	kimya	k101
mehmet	kilim	fizik	f102
ayse	gulcu	tarih	t101
ayse	gulmez	biyoloji	b102
kemal	yasa	biyoloji	b105
fatma	yasa	kimya	k103

ogrenci sahisi bilgileri					
ogrenci no	tel	boyu	kilosu	saglik raporu	fotografi
111	12124435	160	50	var	var

veli bilgileri						
ogrenci no	veli adi	veli soy	veli yak.	veli tel	veli tel 2	adres
111	hasan	velioglu	babasi	64654613	31646	

yazili tablosu				
ogrenci no	ders	ogretmen	not	
111	k9	k101	85	
112	b9	b102	80	
116	b9	b105	65	
118	k10	k103	90	



# Related Tablolarla Calisma

## One to One Relation

StudentID	FirstName	LastName
10	John	Walker
11	Tom	Hanks
12	Kevin	Star
13	Carl	Wall
14	Andrei	Apazniak
15	Mark	High
16	Clara	Star
17	John	Ocean
18	John	Walker
19	Pamela	Star
20	Carl	Wall

StudentID	Street	ZipCode	City	State
10	1234 W 23th Street	33018	Hialeah	Florida
11	1235 N 3th Street	22145	Austwell	Texas
12	1236 SE 12th Street	54234	Orange	California
13	1237 N 5th Street	33018	Hialeah	Florida
14	1238 SW 53th Street	33026	Miami	Florida
15	1239 S 123th Street	22314	Avery	Texas
16	1240 N 1 st Street	12345	Arlington	Virginia
17	1241 NW 2nd Street	65432	Pittsburgh	Penssylvania
18	1242 W 5th Street	22133	Baytown	Texas
19	1243 SE 55th Street	74352	Beachwood	Ohio
20	1244 SW 17th Street	22314	Avery	Texas

- 1) Tom Hanks'in adresi nedir?
- 2) John Walker'in eyaleti nedir?
- 3) ID'si 17 olan kisinin sehri nedir?



# Related Tablolarla Çalışma

# One to Many Relation

CourseID	CourseName	CourseCredit	CourseFee	InstructorID
100	Biology	3	1200	1
200	Math	3	1200	2
300	English	2	600	3
400	Selective	1	200	1

- 1) Biology dersi alan ogrenciler kimler?
  - 2) Selective ders alan ogrencilerin isimleri ?
  - 3) CourseFee 600 olan ogrencilerin isimleri ?

StudentID	FirstName	LastName	CourseID
10	John	Walker	200
11	Tom	Hanks	400
12	Kevin	Star	400
13	Carl	Wall	200
14	Andrei	Apazniak	300
15	Mark	High	400
16	Clara	Star	100
17	John	Ocean	100
18	John	Walker	200
19	Pamela	Star	300
20	Carl	Wall	400



# Related Tablolarla Çalışma

# Many to Many Relation

StudentID	FirstName	LastName
10	John	Walker
11	Tom	Hanks
12	Kevin	Star
13	Carl	Wall
14	Andrei	Apazniak
15	Mark	High
16	Clara	Star
17	John	Ocean
18	John	Walker
19	Pamela	Star
20	Carl	Wall

InstructorID	FirstName	LastName	Phone	Department
1	Mark	Adam	1234567891	Science
2	Eve	Sky	1239876543	Engineering
3	Leo	Ocean	1237845691	Language
4	Andy	Mark	1232134567	Health

- 1) Öğretmeni Mark Adam olan öğrencilerin isimleri nedir?
  - 2) Kevin Star'in öğretmenlerinin isimleri nedir?
  - 3) Pamela Star'in öğretmenlerinin isimleri nedir?



# SQL Data Types

## String Data Types

### Data Type

#### **char(size)**

### Açıklama

Sabit sayıdaki karakterleri (harf, numara veya özel karakter) tutar. Parantez içindeki boyut uzunluğu belirtir. Char(5) tanımlı bir alana 2 karakterlik bir veri girerseniz 5 byte alan ayırr, yani tanımladığınız boyut kadar. Maksimum 255 karakter barındırabilir. Tekrar eden boşluklar değer alındığı zaman silinir. En fazla 8.000 karaktere kadar depolama yapar

#### **varchar(size)**

Değişken sayıdaki karakterleri tutar. En fazla 8.000 karaktere kadar depolama yapar. Varchar(5) tanımlı bir alana 2 karakterlik bir veri girerseniz 2 byte alan ayırr. Buradan varchar ile char arasındaki farkı; char tanımlanan boyut kadar alan ayırrken, varchar girilen karakter boyutu kadar alan ayırr şeklinde ifade edebiliriz.

#### **text**

En fazla 65.535 karaktere kadar veri girilebilir. (Boşluklar dahildir.)

#### **longtext**

4.294.967.295 karaktere kadar metinsel ifadeleri depolayabilir.



# SQL Data Types

**DataTypes    ACIKLAMA**

**MYSQL**

- **TINYINT(boyut)** : Alabilecegi değerler -128 ile 127 arasındadır. Unsigned (Sadece pozitif değerler girilecek) olarak tanımlı ise 0 ile 255 arasındadır. "Boyut" ile alabilecegi sınırı belirtebiliriz. Hafızada kapladığı alan ise 1 byte
- **SMALLINT(boyut)** : -32.768 ile 32.767 arasında değer alır. Unsigned tanımlı aralık 0 ile 65535 arasında değer alır. "Boyut" ile alabilecegi sınırı belirtebiliriz. Hafızada kapladığı alan: 2 byte
- **MEDIUMINT(boyut)** : -8.388.608 ile 8.388.607 arasında değer alır. Unsigned tanımlı aralık 0 ile 16777215 arasındadır. "Boyut" ile alabilecegi sınırı belirtebiliriz. Hafızada kapladığı alan: 3 byte.
- **INT** : Alabilecegi değerler -2147483648 ile 2147483647 arasındadır. Unsigned tanımlı aralık 0 ile 4294967295 arasındadır. "Boyut" ile alabilecegi sınırı belirtebiliriz. Hafızada kapladığı alan: 4 byte.
- **BIGINT(boyut)** : -9.223.372.036.854.775.808 ile 9.223.372.036.854.775.807 arasında değer alır. "Boyut" ile alabilecegi sınırı belirtebiliriz. Hafızada kapladığı alan: 8 byte.
- **FLOAT(boyut,d)** : Değer aralıkları; -3.402823466E+38 ile -1.175494351E-38, 0 arası ve 1.175494351E-38 ile 3.402823466E+38 arasındadır. Küçük rakamlı virgülü ifadeler için kullanılır. "Boyut" ile sayının (virgülü kismi dahil) alabilecegi en fazla miktar belirtilirken, "d" ile virgülden sonra kaç basamak olacağı belirtilir. Örneğin, 32.658 sayısını saklayacağımız float türü bir sütun tanımlarken, FLOAT(5,3) olarak tanımlarız. Buradaki 5 rakamı, sayının tamamının (noktasız olarak) basamak uzunluğu, 3 rakam ise noktadan sonraki hane sayısını ifade eder. Float veri türünde eğer ondalıklı hane daha uzun ise sayının yuvarlanma durumu oluşabilir. FLOAT(5,2) olarak tanımladığımız bir sütunda 275.199 sayısını saklamak istersek, MySQL otomatik olarak bu sayıyı 275.20 olarak saklayacaktır. Boyut değeri en fazla 23 olabilir. Unsigned olarak çalışmazlar. Hafızada kapladığı alan: 4 byte.
- **DOUBLE(boyut,d)** : Büyük noktalı sayı. Büyük rakamlı virgülü ifadeler için kullanılır. "Boyut" ile sayının virgülü kismi dahil alabilecegi en fazla miktar belirtilirken, "d" ile virgülden sonra kaç basamak olacağı belirtilir. Boyut değeri en fazla 53 olabilir. Unsigned olarak çalışmazlar. Hafızada kapıldığı alan: 8 byte.



# SQL Data Types

## Numeric Data Types

### DBMS Numeric Types:

DBMS and version	Types
MySQL 5.7	INTEGER(TINYINT, SMALLINT, MEDIUMINT, INT BIGINT, INTEGER) FIXED-POINT(DECIMAL, NUMERIC) FLOATING-POINT(FLOAT, DOUBLE) BIT-VALUE(BIT),
PostgreSQL 9.5.3	SMALLINT, INTEGER, BIGINT, DECIMAL, NUMERIC, REAL, DOUBLE PRECISION, SMALLSERIAL, SERIAL, BIGSERIAL
SQL Server 2014	EXACT NUMERICS(BIGINT, BIT, DECIMAL, INT, MONEY, NUMERIC, SMALLINT, SMALLMONEY, TINYINT) APPROXIMATE NUMERICS(FLOAT, REAL )
Oracle 11g	NUMBER FLOATING-POINT(BINARY_FLOAT, BINARY_DOUBLE)



# SQL Data Types

## Date Data Types

- **DATE** : Desteklenen aralık ‘1000-01-01’ ile ‘9999-12-31’ arasıdır. MySQL tarihleri YYYY-MM-DD biçiminde gösterir.
- **DATETIME** : Desteklenen aralık ‘1000-01-01 00:00:00’ ile ‘9999-12-31 23:59:59’ arasıdır. MySQL DATETIME değerlerini ‘YYYY-MM-DD HH:MM:SS’ biçiminde gösterir.
- **TIME** : Sadece saat verisi saklamak için kullanılır. Desteklenen aralık ‘-838:59:59’ ile ‘838:59:59’ arasındadır. MySQL TIME değerlerini ‘HH:MM:SS’ biçiminde gösterir.
- **YEAR**: 2 veya 4 basamaklı yıl bilgisini saklamak için kullanılır. Dört basamaklı verilerde 1901 ile 2155 arası değer saklanır. İki basamaklı verilerde ise 70 ile 69 (1970 ile 2069) arası değerler saklanır.

Standart “Date Format” , “YYYY- MM - DD”. Ornegin 2020-01-23

Tarih formatını “**ALTER SESSION SET NLS\_DATE\_FORMAT = “YYYY-MMM-DD”** kodu kullanılarak degistirilebilir. Koddan sonra tarih 2020 - Jan – 13 olur.



# SQL Data Types

## BLOB Data Types

### *Data Type*

### *Aciklama*

**BLOB**

“**BLOB**” , “Binary Large OBjects” demektir  
“**BLOB**” resim,video,ses gibi datalari binary formatina cevirerek depolar.



# SQL Komutları

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

veritabanı ve tablolar için yetki verilir veya geri alınır

**GRANT** : Bir kullanıcıya yetki vermek için kullanılır.

**REVOKE** : Bir kullanıcıya verilen yetkiyi geri almak için kullanılır.

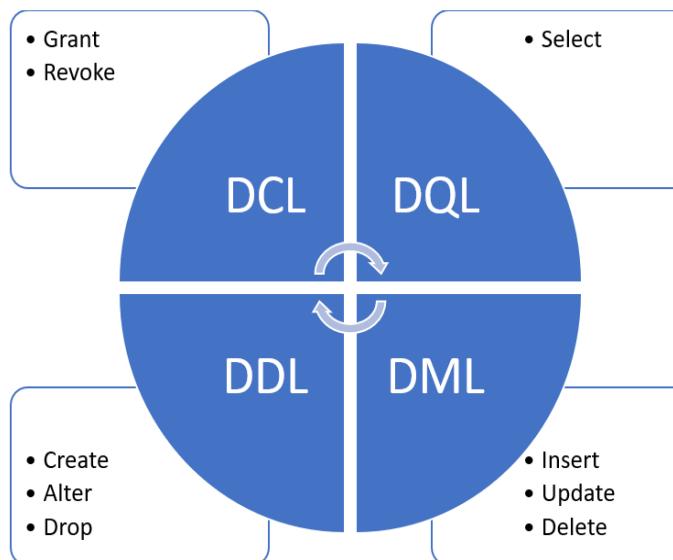
3. Veri Tanımlama Dili (Data Definition Language - DDL)

veritabanı ve tabloları oluşturma, değiştirme ve silme işlemleri yapılır

**CREATE** : Bir veritabanı veya tablo oluşturur.

**ALTER** : Bir veritabanı veya tabloyu günceller.

**DROP** : Bir veritabanını veya tabloyu siler.



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

**mevcut** kayıtların bir kısmını veya tamamını tanımlanan koşullara bağlı olarak alır.

**SELECT** : Veritabanındaki verileri alır.

2. Veri Degistirme Dili (Data Manipulation Language - DML)

veritabanına yeni kayıt ekleme, mevcut kayıtları güncelleme ve silme işlemleri yapılır.

**INSERT** : Veritabanına yeni veri ekler.

**UPDATE** : Veritabanındaki verileri günceller.

**DELETE** : Veritabanındaki verileri siler.



# Table Nasıl Olusturulur?

## 1) Create from Scratch

```
CREATE TABLE student_table  
(  
id char(11),  
name varchar(50),  
grade int,  
adres varchar(100),  
last_update date  
);
```

## 2) Var olan tablodan yeni tablo olusturmak

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



# Table Nasıl Olusturulur?

## 1) Create from Scratch

### Practice1:

“tedarikciler” isminde bir tablo olusturun ve “tedarikci\_id”, “tedarikci\_ismi”, “tedarikci\_adres” ve “ulasim\_tarihi” field’lari olsun.

```
CREATE TABLE tedarikciler  
(  
    tedarikci_id char(10),  
    tedarikci_ismi varchar(50),  
    tedarikci_adres varchar(100),  
    ulasim_tarihi date  
) ;
```

## 2) Var olan tablodan yeni tablo olusturmak

“tedarikciler\_id\_name” isminde bir tabloyu  
“tedarikciler” tablosundan olusturun.  
Icinde “tedarikci\_id”, “tedarikci\_ismi” field’lari olsun.

```
CREATE TABLE tedarikci_ziyaret  
AS  
SELECT tedarikci_ismi,ulasim_tarihi  
FROM tedarikciler;
```



# Bir field'in “tekrarsız” deger almasi nasil saglanir?

“id” field'ini “tekrarsız” yapmak icin , id field'inda Data Type'dan sonra “**UNIQUE**” yazmak gereklidir

```
CREATE TABLE students_table  
(  
    id char(11) UNIQUE,  
    name varchar(50),  
    grade int,  
    adres varchar(100),  
    last_update date  
);
```



# Bir field'in “NULL” deger almamasi nasil saglanir?

“name” field’inin “NULL” deger kabul etmemesi icin , name field’inda Data Type’dan sonra “NOT NULL” yazmak gereklidir

```
CREATE TABLE students_table
(
    id char(11) UNIQUE,
    name varchar(50) NOT NULL,
    grade int,
    adres varchar(100),
    last_update date
);
```



## Bir Tabloya “Primary Key” Nasıl Eklenir

- 1) Primary Key bir record'u tanımlayan bir field veya birden fazla field'in kombinasyonudur.
- 2) Primary Key Unique'dir
- 3) Bir tabloda en fazla bir Primary Key Olabilir
- 4) Primary Key field'ında hiç bir data NULL olamaz

“id” field'ini “primary key” yapmak için 2 yol var

- 1) Data Type'dan sonra “PRIMARY KEY” yazarak.

```
CREATE TABLE students_table
(
    id int PRIMARY KEY,
    name varchar(50) NOT NULL,
    grade int,
    adres varchar(100),
    last_update date
);
```



## Bir Tabloya “Primary Key” Nasıl Eklenir

### 2) CONSTRAINT Keyword Kullanılarak Primary Key Tanımlanabilir

“CONSTRAINT constraintName PRIMARY KEY(column1, column2, ... column\_n)”

```
CREATE TABLE students
(
    id int,
    name varchar(50) NOT NULL,
    grade int,
    address varchar(100), last_modification date,
    CONSTRAINT id_pk PRIMARY KEY(id)
);
```



# SQL

Structured Query Language  
Yapılandırılmış Soru Dili

PRIMARY KEY-FOREIGN KEY TANIMLAMA  
TABLOYA DATA EKLEME

İPEK BİLGİN  
Java Full Stack Developer



## Bir Tabloya “Primary Key” Nasıl Eklenir

### Practice 3:

“sehirler” isimli bir Table olusturun. Tabloda “alan\_kodu”, “isim”, “nufus” field’lari olsun. Isim field’i bos bırakılamasıın.  
1.Yontemi kullanarak “alan\_kodu” field’ini “Primary Key” yapın

`CREATE TABLE sehirler`

```
(  
alan_kodu int PRIMARY KEY,  
name varchar(50) NOT NULL,  
isim varchar(20),  
nufus int  
);
```

### Practice 4:

“ogretmenler” isimli bir Table olusturun. Tabloda “id”, “isim”, “brans”, “cinsiyet” field’lari olsun. Id field’i tekrarlı değer Kabul etmesin.  
2.Yontemi kullanarak “id ve isim” field’lerinin birleşimini “primary key” yapın

`CREATE TABLE ogretmenler`

```
(  
id char(10) UNIQUE,  
isim varchar(20),  
brans varchar(20),  
CONSTRAINT ogretmenler_pk PRIMARY KEY (id,isim)  
);
```



# Tabloya “Foreign Key” Nasil Eklenir ?

- Foreign Key iki tablo arasında Relation olusturmak icin kullanilir.
- Foreign Key baska bir tablonun Primary Key'ine baglidir.
- Referenced table (baglanilan tablo, Primary Key'in oldugu Tablo) *parent table* olarak adlandirilir. Foreign Key'in oldugu tablo ise *child table* olarak adlandirilir.
- Bir Tabloda birden fazla Foreign Key olabilir
- Foreign Key NULL degeri alabilir

Note 1: “Parent Table” olmayan bir id'ye sahip datayi “Child Table”a ekleyemezsiniz

Note 2: Child Table'i silmeden Parent Table'i silemezsiniz. Once “Child Table” silinir, sonra “Parent Table” silinir.



# Tabloya “Foreign Key” Nasıl Eklenir ?

SIRKETLER Tablosu

Primary Key

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

Parent Table

PERSONEL Tablosu

Foreign Key

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

Child Table



# Tabloya “Foreign Key” Nasıl Eklenir ?

## Practice 5:

“tedarikciler” isimli bir tablo olusturun. Tabloda “tedarikci\_id”, “tedarikci\_ismi”, “iletisim\_isim” field’lari olsun ve “tedarikci\_id” yi Primary Key yapin.

“urunler” isminde baska bir tablo olusturun “tedarikci\_id” ve “urun\_id” field’lari olsun ve “tedarikci\_id” yi Foreign Key yapin.

### CREATE TABLE urunler

```
(  
    tedarikci_id char(10),  
    product_id char(10),  
    CONSTRAINT urunler_fk FOREIGN  
    KEY (tedarikci_id) REFERENCES  
    tedarikciler (tedarikci_id) );
```

### CREATE TABLE tedarikciler

```
(  
    tedarikci_id char(10),  
    tedarikci_ismi varchar(50),  
    iletisim_isim varchar(50),  
    CONSTRAINT tedarikci_pk PRIMARY KEY  
    (tedarikci_id) );
```



# Tabloya “Foreign Key” Nasıl Eklenir ?

## Practice 6:

“*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 tedarikciler
```

```
(  
    tedarikci_id int,  
    tedarikci_isim varchar(50),  
    iletisim_isim varchar(50),  
    CONSTRAINT tedarikci_pk PRIMARY KEY  
  
(tedarikci_id,tedarikci_isim)  
);
```

```
CREATE TABLE urunler
```

```
(  
    tedarikci_id int,  
    urun_id varchar(10),  
    CONSTRAINT fk_tedarikci FOREIGN KEY  
        (tedarikci_id,urun_id) REFERENCES  
        tedarikciler(tedarikci_id,tedarikci_isim) );
```



# SQL

Structured Query Language  
Yapılandırılmış Soru Dili

TABLOYA DATA EKLEME  
TABLODAKI DATALARI UPDATE ETME

İPEK BİLGİN

Java Full Stack Developer



# Tabloya Nasil Data Eklenir (**INSERT INTO**)?

**INSERT INTO** komutu, Oracle SQL'de tabloya bir veya birden fazla kayıt eklemek için kullanılır.

Ogrenciler isminde bir tablo oluşturun, içinde  
id, isim, not\_ortalaması, adres ve son\_degistirme\_tarihi fieldleri olsun

```
CREATE TABLE ogrenciler
(
    id int,
    isim varchar(50),
    not_ortalaması int,
    adres varchar(100),
    last_modification date,
    CONSTRAINT id_pk PRIMARY KEY(id)
);
```

1) Tüm Field'lere data eklemek için

```
INSERT INTO students VALUES (123456789, 'Ali Can', 11, 'İstanbul,bakirkoy', '14-Oct-2020');
```

2) Bazi Field'lere data eklemek için

```
INSERT INTO students(id,name) VALUES (123456789, 'Ali Can');
```



# Tabloya Nasil Data Eklenir (**INSERT INTO**)?

**Note:** INSERT INTO kodunu kullanarak bir tabloya data eklemek istediginizde, **CONSTRAINT**'lere uymak zorundayiz. Ornegin; **NOT NULL** yazan kisma bir deger atamak zorundayiz

Personel isminde bir tablo olusturun, icinde  
id, isim, soyisim, email, ise\_baslama\_tarihi ve maas fieldlari olsun, isim field'i bos bırakılamasın

```
CREATE TABLE personel
(
    id char(10),
    isim varchar(50) NOT NULL,
    soyisim varchar(50),
    email varchar(50),
    ise_baslama_tar date,
    maas int
);
```

```
INSERT INTO personel (id, is_unvani) VALUES(123456789, 'isci');
```

```
ORA-01400: cannot insert NULL into
("SQL_LFGUHVRSSOWWDACLEMHRMQGCJQ"."STUDENTS"."NAME") ORA-06512: at "SYS.DBMS_SQL",
Line 1721
```



# Tabloya Nasil Data Eklenir (**INSERT INTO**)?

## Practice 7:

“*ogretmenler*” isminde bir SQL tablosu olusturun. Icinde “kimlik\_no”, “isim”, “brans” ve “cinsiyet” field’lari olsun.

“*ogretmenler*” tablosuna bilgileri asagidaki gibi olan bir kisi ekleyin.

*Kimlik\_no: 234431223, isim: Ayse Guler, brans : Matematik, cinsiyet: kadin.*

```
CREATE TABLE ogretmenler
(
kimlik_no char(11),
isim varchar(50),
brans varchar(50),
cinsiyet varchar(50)
);
```

## Practice 8:

“*ogretmenler*” tablosuna bilgileri asagidaki gibi olan bir kisi ekleyin.

*Kimlik\_no: 567597624, isim: Kemal Yasa*

```
INSERT INTO ogretmenler VALUES ('234431223','ayse guler', 'matematik','kadin');
```



# Tablodaki Data Nasil Update Edilir (UPDATE SET)?

1) Bir tedarikciler tablosu olusturun icinde id, isim ve iletisim\_isim field'lari olsun. Id ve isim'i beraber Primary Key yapin

`CREATE TABLE tedarikciler`

```
(  
    id int,  
    isim varchar(50),  
    iletisim_isim varchar(50),  
    CONSTRAINT tedarikci_pk PRIMARY KEY (id, isim) );
```

3) id'si 1 olan tedarikcinin ismini 'KRM' ve iletisim\_isim'ini 'Kemal Kan' yapin

`UPDATE tedarikciler`

```
SET isim = 'KRM',  
    iletisim_isim = 'Hasan Kan'  
WHERE id =1;
```

2) Icine 3 kayit 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');`

4) Ismi RDB olan tedarikcinin iletisim isim'ini Kemal Yasa yapin

`UPDATE tedarikciler`

```
SET iletisim_isim='Kemal Yasa'  
WHERE isim='RDB';
```



# Tablodaki Data Nasil Update Edilir (UPDATE SET)?

Practice 11: verilen tablolara gore asagidaki islemleri yapin.

- a) Urnler tablosundan Ali Can'in aldigı ürünün ismini, tedarikci tablosunda irtibat\_isim Merve Temiz olan şirketin ismi ile degistirin
- b) TV satın alan musterinin ismini, Apple'in irtibat\_isim'ı ile degistirin

**CREATE TABLE tedarikci**

```
(  
    id int PRIMARY KEY,  
    isim varchar(50),  
    irtibat_isim varchar(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 int,  
    urun_isim varchar(50),  
    musteri_isim varchar(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');
```



# Tablodaki Data Nasıl Update Edilir (UPDATE SET)?

ID	ISIM	IRTIBAT_ISIM
100	IBM	Ali Can
101	APPLE	Merve Temiz
102	SAMSUNG	Kemal Can
103	LG	Ali Can

**tedarikci**

a) Urunler tablosundan Ali Can'in aldigı ürünün ismini, tedarikci tablosunda iribat\_isim Merve Temiz olan şirketin ismi ile degistirin

## UPDATE ürünler

```
SET urun_isim= (SELECT isim  
                  FROM tedarikci  
                  WHERE irtibat_isim='Merve Temiz')  
WHERE musteri_isim='Ali Can';
```

TEDARIKCI_ID	URUN_ID	URUN_ISIM	MUSTERIR_ISIM
103	1004	Phone	Ali Can
100	1001	Laptop	Suleyman
101	1002	iPad	Fatma
102	1003	TV	Ramazan

## urunler

b) TV satin alan musterinin ismini, Apple'in irtibat isim'i ile degistirin

UPDATE ürünler

**SET musteri\_isim=(SELECT irtibat\_isim**

FROM tedarikci

```
WHERE isim='APPLE')  
WHERE urun isim='TV';
```



# Tablodaki Data Nasil Update Edilir (UPDATE SET)?

- 1) Ogrenciler tablosu olusturun. Icinde id,isim,veli\_isim ve grade field'lari olsun. Id ve isim fieldlari birlikte Primary Key olsun.

```
CREATE TABLE ogrenciler
(
    id char(3),
    isim varchar(50),
    veli_isim varchar(50),
    yazili_notu int,
    CONSTRAINT ogrenciler_pk PRIMARY KEY (id)
);
```

- 3) notlar tablosu olusturun. ogrenci\_id,ders\_adi,yazili\_notu field'lari olsun, ogrenci\_id field'i Foreign Key olsun

```
CREATE TABLE notlar
(
    ogrenci_id char(3),
    ders_adi varchar(30),
    yazili_notu int,
    CONSTRAINT notlar_fk FOREIGN KEY (ogrenci_id) REFERENCES ogrenciler (id )
);
```

- 2) 3 kisiyi tabloya ekleyin. (123, 'Ali Can', 'Hasan',75), (124, 'Merve Gul', 'Ayse',85), (125, 'Kemal Yasa', 'Hasan',85).

```
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);  
4) notlar tablosuna 3 kayit ekleyin ('123','kimya',75), ('124','fizik',65),('125','tarih',90)

```
INSERT INTO notlar VALUES ('123', 'kimya', 75);
INSERT INTO notlar VALUES ('124', 'fizik', 65);
INSERT INTO notlar VALUES ('125', 'tarih', 90);
```

- 5) Tum ogrencilerin yazili notlarini notlar tablosundaki ile update edin

```
UPDATE ogrenciler
SET yazili_notu= (SELECT yazili_notu
                   FROM notlar
                  WHERE ogrenciler.id=notlar.ogrenci_id)
WHERE id>100;
```



# Tablodaki Data Nasil Update Edilir (UPDATE SET)?

## Practice 9:

- a) “*ogrenciler*” isminde bir tablo olusturun. icinde “*id*”, “*statu*”, “*name*”, “*ortalama\_not*”, “*okul\_adi*” field’lari olsun
- b) Tabloya 5 ogrenci ekleyin ortalamalari 2.6, 1.9, 3.2, 3.8, 3.5 olsun.
- c) Ortalamasi 3.0 ve ustu olan ogrencilerin statu field’ina “*odullu ogrenci*” yazdirin.

## Practice 10:

- a) “*Ogrenciler*” isminde bir tablo olusturun,icinde “*id*”, “*isim*”, “*ortalama\_not*”, “*okul\_ismi*” field’lari olsun.
- b) Tabloya 5 tane ogrenci ekleyin, ortalamalari 2.6, 1.9, 3.2, 3.8, 3.5 olsun,okul isimleri birbirinden farkli olsun.
- c) “*veliler*” isminde bir tablo daha olusturun, icinde “*ogrenci\_id*”, “*veli\_isim*”, “*okul\_isim*” field’leri olsun
- d) 5 veli ekleyin, okul isimleri ogrenci tablosundaki 5 okul adi olsun, her velinin okulu farkli olsun.
- e) Okul adi ayni olan ogrenci ismi yerine veli ismini yazin



# SQL

Structured **Query Language**  
Yapılandırılmış Sorğu Dili

TABLODAKİ DATALARI UPDATE ETME  
TABLODAN DATA SİLME

**İPEK BİLGİN**  
Java Full Stack Developer



# Tablodaki Data Nasil Update Edilir (UPDATE SET)?

Mart\_satislar isimli bir tablo olusturun, icinde urun\_id, musteri\_isim, urun\_isim ve urun\_fiyat field'lari olsun

```
CREATE TABLE mart_satislar
(
    urun_id int,
    musteri_isim varchar(50),
    urun_isim varchar(50),
    urun_fiyat int
);
```

```
INSERT INTO mart_satislar VALUES (10, 'Ali', 'Honda',75000);
INSERT INTO mart_satislar VALUES (10, 'Ayse', 'Honda',95200);
INSERT INTO mart_satislar VALUES (20, 'Hasan', 'Toyota',107500);
INSERT INTO mart_satislar VALUES (30, 'Mehmet', 'Ford', 112500);
INSERT INTO mart_satislar VALUES (20, 'Ali', 'Toyota',88000);
INSERT INTO mart_satislar VALUES (10, 'Hasan', 'Honda',150000);
INSERT INTO mart_satislar VALUES (40, 'Ayse', 'Hyundai',140000);
INSERT INTO mart_satislar VALUES (20, 'Hatice', 'Toyota',60000);
```

- 1) Ismi hatice olan musterinin urun\_id'sini 30,urun\_isim'ini Ford yapin
- 2) Toyota marka araclara %10 indirim yapin
- 3) Ismi Ali olanların urunfiyatlarını %15 artırın
- 4) Honda araclarının urun kodu'nu 50 yapın



# Tablodan Data Nasil Silinir (Delete)?

```
CREATE TABLE ogrenciler
(
    id char(3),
    isim varchar(50),
    veli_isim varchar(50),
    yazili_notu int,
    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);
```

- 1) “DELETE FROM ogrenciler” tablodaki tum datalari siler, fakat tabloyu silmez.  
“DELETE FROM ogrenciler”, kodunu kullaninca bos bir tablo kalir.

no data found



# Tablodan Data Nasil Silinir (Delete)?

2) CREATE TABLE ogrenciler  
(  
id char(3),  
isim varchar(50),  
veli\_isim varchar(50),  
yazili\_notu int,  
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 );

ID	ISIM	VELI_ISIM	YAZILI_NOTU
123	Ali Can	Hasan	75
124	Merve Gul	Ayse	85
125	Kemal Yasa	Hasan	85

- “ DELETE FROM ogrenciler WHERE isim = ‘Ali Can’ ” kodu isim olarak **Ali Can** girilen kaydi (record) siler.
- “ DELETE FROM ogrenciler WHERE yazili\_notu = 85 kodu not olarak 85 girilen kaydi (record) siler.

ID	ISIM	VELI_ISIM	YAZILI_NOTU
124	Merve Gul	Ayse	85
125	Kemal Yasa	Hasan	85

no data found



# Tablodan Data Nasil Silinir (Delete)?

3) CREATE TABLE ogrenciler  
(  
id char(3),  
isim varchar(50),  
veli\_isim varchar(50),  
yazili\_notu int,  
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);

ID	ISIM	VELI_ISIM	YAZILI_NOTU
123	Ali Can	Hasan	75
124	Merve Gul	Ayse	85
125	Kemal Yasa	Hasan	85

“ DELETE FROM ogrenciler WHERE isim = ‘Ali Can’ OR veli\_isim=‘Ayse’ kodu isim olarak **Ali Can** veya veli\_isim olarak **Ayse** girilen kaydi (record) siler.

ID	ISIM	VELI_ISIM	YAZILI_NOTU
124	Merve Gul	Ayse	85
125	Kemal Yasa	Hasan	85

no data found



### Practice 12:

ID	ISIM	SOYISIM	EMAIL	ISE_BASLAMA_TAR	IS_UNVANI	MAAS
123456789	Ali	Can	alican@gmail.com	10-APR-10	isci	5000
123456788	Veli	Cem	velicem@gmail.com	10-JAN-12	isci	5500
123456787	Ayse	Gul	aysegul@gmail.com	01-MAY-14	muhasebeci	4500
123456789	Fatma	Yasa	fatmayasa@gmail.com	10-APR-09	muhendis	7500

- 1) Verilen bilgilerin oldugu bir tablo olusturun
- 2) Tumisci'lerin maasina %20 zam yapin
- 3) Muhendis'lerin maasini 7000 yapin
- 4) Muhasebecinin adini 'Sena' soyadini 'Yilmaz' yapin
- 5) Maasi 5000 den buyuk olanlari silin



# Tablodan Data Nasil Silinir (**TRUNCATE**)?

- “**Truncate**” kodu kullanilarak bir tablo silinirse datalarin geri getirilme ihtimali olmaz
- “**Truncate**” kodu geri getirilmesini (**rolling back**) istemeyeceginiz tablolari silmek icin kullanilir.

**TRUNCATE TABLE** customers

**DELETE FROM** customers;

**INTERVIEW QUESTION :** TRUNCATE, DELETE FROM VE DROP arasindaki fark nedir ?

DELETE FROM ile sildigimiz kayitlari geri getirebiliriz ama TRUNCATE ile silinen kayitlar geri getirilemez.



# Tablo Nasil Silinir (Drop)? (Tum Tablo Icerigi ve Tablo Yapisi )

Ogrenciler isminde bir Tablo olusturun icinde id, isim, yazili\_not, adres ve son degistirme field'lari olsun. 3 kisiyi tabloya ekleyin

```
CREATE TABLE ogrenciler  
(  
id number(9),  
isim varchar2(50),  
adres varchar2(100),  
yazili_not number(3)  
);
```

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

DROP TABLE ogrenciler; ogrenciler Tablosunu siler (**MYSQL İÇİN GEÇERLİ**)  
**ALTTAKİLER ORACLE SQL İÇİN GEÇERLİ**

DROP TABLE ogrenciler; ogrenciler Tablosunu siler ve RecycleBin'e yollar  
FLASHBACK TABLE ogrenciler TO BEFORE DROP; dosyayı geri alır.  
PURGE TABLE ogrenciler; RecycleBin'de olan dosyayı geri getiremeyecek şekilde siler  
DROP TABLE ogrenciler PURGE; Kullandığımız bir dosyayı getiremeyecek şekilde siler

**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 KOMUTU

## 1) Tablodaki Tum Field'lari Cagirma

```
CREATE TABLE ogrenciler  
(  
id int,  
isim varchar(50),  
adres varchar(100),  
yazili_not int  
);
```

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

```
SELECT *  
FROM students; Tablodaki tum datalari getirir
```

```
SELECT *  
FROM students;  
WHERE yazili_not >80; Tablodaki yazili_notu 80'den buyuk olan kayitlari getirir
```



## WHERE ile Kullanılan Mantıksal Operatorler

=	==> 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



## SELECT KOMUTU

### 2) Tablodaki Belli Bir Field'i Çağırma

**SELECT** adres

**FROM** students;

Tablodan sadece adres field'indaki tümdataları getirir

**SELECT** adres

**FROM** students

**WHERE** yazılı\_not=85;

Tablodan sadece yazılı notu 85 olanların adres field'indaki dataları getirir

ADRES
Ankara
Ankara
Istanbul

ADRES
Ankara
Istanbul



## SELECT KOMUTU

### 3) Tablodan Birden Fazla Field'i Çağırma

```
SELECT adres,isim  
FROM ogrenciler;
```

Tablodan adres ve isim field'indaki tümdataları getirir

```
SELECT adres,isim  
FROM ogrenciler  
WHERE yazili_not>80;
```

Tablodan yazılı notu 80'den büyük olan kayıtların adres ve isim field'indaki dataları getirir

ADRES	ISIM
Ankara	Ali Can
Ankara	Merve Gul
Istanbul	Kemal Yasa

ADRES	ISIM
Ankara	Merve Gul
Istanbul	Kemal Yasa



## SELECT KOMUTU

- 4) Tablodan Bir Kayida Ait Birden Fazla Field'i Cagirma

```
SELECT adres,isim  
FROM ogrenciler  
WHERE id=123 ;
```

Tablodan id'si 123 olan kaydin adres ve isim field'indaki datalari getirir



# IN CONDITION

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

```
CREATE TABLE musteriler  
(  
    urun_id int,  
    musteri_isim varchar(50),  
    urun_isim varchar(50)  
);
```

```
SELECT *  
FROM musteriler  
WHERE urun_isim ='Orange' OR urun_isim ='Apple' OR urun_isim ='Apricot';
```

```
SELECT *  
FROM musteriler  
WHERE urun_isim IN ('Orange', 'Apple', 'Apricot');
```

```
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');
```

URUN_ID	MUSTERI_ISIM	URUN_ISIM
10	Mark	Orange
10	Mark	Orange
20	John	Apple
20	Mark	Apple
10	Adem	Orange
40	John	Apricot
20	Eddie	Apple



# BETWEEN CONDITION

**BETWEEN Condition** iki mantıksal ifade ile tanımlayabileceğimiz durumları tek komutla yazabilme imkani verir. Yazdığımız 2 sınırlı aralığa dahildir (**INCLUSIVE**)

```
CREATE TABLE musteriler  
(  
    urun_id int,  
    musteri_isim varchar(50),  
    urun_isim varchar(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');
```

1) Urun\_id 20 ile 40 arasında olan ürünlerin tüm bilgilerini listeleyiniz

```
SELECT *  
FROM musteriler  
WHERE urun_id>=20 AND urun_id<=40;
```

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

URUN_ID	MUSTERI_ISIM	URUN_ISIM
20	John	Apple
30	Amy	Palm
20	Mark	Apple
40	John	Apricot
20	Eddie	Apple

2) İsminin ilk harfi E ile J arasında olan kişilerin tüm bilgilerini listeleyin



# NOT BETWEEN CONDITION

**NOT BETWEEN Condition** iki mantiksal ifade ile tanimlayabilecegimiz durumları tek komutla yazabilme imkani verir. Yazdigimiz 2 sinirda araliga harictir (**EXCLUSIVE**)

```
CREATE TABLE musteriler  
(  
    urun_id int,  
    musteri_isim varchar(50),  
    urun_isim varchar(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');
```

1) Urun\_id 20 ile 40 arasında olmayan ürünlerin tüm bilgilerini listeleyiniz

```
SELECT *  
FROM musteriler  
WHERE urun_id<20 OR urun_id>30;
```

```
SELECT *  
FROM musteriler  
WHERE urun_id NOT BETWEEN 20 AND 40 ;
```

URUN_ID	MUSTERI_ISIM	URUN_ISIM
10	Mark	Orange
10	Mark	Orange
10	Adem	Orange
40	John	Apricot

2) Isminin ilk harfi E ile J arasında olmayan kişilerin tüm bilgilerini listeleyin



## Practice 13

ID	ISIM	SOYISIM	EMAIL	ISE_BASLAMA_TAR	IS_UNVANI	MAAS
123456789	Ali	Can	alican@gmail.com	10-APR-10	isci	5000
123456788	Veli	Cem	velicem@gmail.com	10-JAN-12	isci	5500
123456787	Ayse	Gul	aysegul@gmail.com	01-MAY-14	muhasebeci	4500
123456789	Fatma	Yasa	fatmayasa@gmail.com	10-APR-09	muhendis	7500

- a) Yukarda verilen “personel” tablosunu olusturun
- b) Tablodan maasi 5000’den az veya unvaniisci olanların isimlerini listeleyin
- c) Iscilerin tum bilgilerini listeleyin
- d) Soyadi Can,Cem veya Gul olanların unvanlarını ve maaslarını listeleyin
- e) Maasi 5000’den cok olanların emailve is baslama tarihlerini listeleyin
- f) Maasi 5000’den cok veya 7000’den az olanların tum bilgilerini listeleyin



## Tekrar Soruları

- 1) DELETE ile TRUNCATE arasindaki fark nedir ?
- 2) DELETE ile DROP arasindaki fark nedir?
- 3) DROP ile DROP PURGE arasindaki fark nedir?(oracleSql için)
- 4) Asagidaki sorgu ile ayni sonucu veren bir sorgu yaziniz.

```
SELECT *  
FROM ogrenciler  
WHERE yas>=8 AND yas<=17;
```

- 5) Asagidaki sorgu ile ayni sonucu veren bir sorgu yaziniz

```
SELECT *  
FROM ogrenciler  
WHERE yas<8 OR yas>17;
```

- 6) Asagidaki sorgu ile ayni sonucu veren bir sorgu yaziniz

```
SELECT *  
FROM ogrenciler  
WHERE yas = 6 OR yas = 7 OR yas = 8 OR yas = 9;
```



# Tekrar Sorularinin Cevaplari

## 1) DELETE ile TRUNCATE arasindaki fark nedir ?

- A) TRUNCATE tum kayitlari siler, DELETE istersek tum kayitlari, istersek belirli kayitlari siler
- B) DELETE ile sildigimiz datalari ROLLBACK yapabiliriz, TRUNCATE ile silinenler geri getirilemez
- C) DELETE ile WHERE komutunu kullanabiliriz ama TRUNCATE ile kullanamayiz

## 2) DELETE ile DROP arasindaki fark nedir?

DELETE kayitlari siler, DROP ise tab;olari.

## 3) DROP ile DROP PURGE arasindaki fark nedir?(oracleSQL icin)

DROP ile sildigimiz dosyalar RECYCLEBIN'e gider. PURGE RECYCLEBIN'deki dosyalari geri getirilmeyecek sekilde siler. DROP PURGE beraber kullanilirsa geri getirilmeyecek sekilde silinir.

## 4) Asagidaki soru ile ayni sonucu veren bir soru yaziniz.

```
SELECT *  
FROM ogrenciler  
WHERE yas>=8 AND yas<=17;
```

## 5) Asagidaki soru ile ayni sonucu veren bir soru yaziniz

```
SELECT *  
FROM students  
WHERE age<8 OR yas>17;
```

## 6) Asagidaki soru ile ayni sonucu veren bir soru yaziniz

```
SELECT *  
FROM students  
WHERE yas = 6 OR yas= 7 OR yas = 8 OR yas = 9;
```

```
SELECT *  
FROM ogrenciler  
WHERE yas BETWEEN 8 AND 17;
```

```
SELECT *  
FROM ogrenciler  
WHERE yas NOT BETWEEN 8 AND 17;
```

```
SELECT *  
FROM ogrenciler  
WHERE yas IN (6,7,8,9);
```



# SQL

**Structured Query Language**  
**Yapılandırılmış Sorgu Dili**

## SUBQUERIES

**iPEK BİLGİN**  
**Java Full Stack Developer**



# KISA TEKRAR

Personel isminde bir tablo olusturun. Icinde id, isim, sehir, maas ve sirket field'lari olsun. Id'yi 2.yontemle PK yapin

```
CREATE TABLE personel
(
    id int,
    isim varchar(50),
    sehir varchar(50),
    maas int,
    sirket varchar(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');
```

Personel\_bilgi isminde bir tablo olusturun. Icinde id, tel ve cocuk sayisi field'lari olsun. Id'yi FK yapin ve personel tablosu ile relation kurun

```
CREATE TABLE personel_bilgi
(
    id int,
    tel char(10) UNIQUE ,
    cocuk_sayisi int,
    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);
```



# KISA TEKRAR

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
456789152	Veli Sahin	Ankara	4500	Ford
123456710	Hatice Sahin	Bursa	4500	Honda

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

**SORU 1)** Personel\_bilgi tablosundan 5 cocugu olan kisinin cocuk sayisini 2 yapin

```
UPDATE personel_bilgi  
SET cocuk_sayisi=2  
WHERE cocuk_sayisi=5;
```



## Tekrar Soruları

**SORU 2)** Personel tablosundan ücreti 4500 veya 5000 olanların maaşlarını %10 artırın

```
UPDATE personel  
SET maas=maas*1.1  
WHERE maas IN (4500,5000);
```

**SORU 3)** Personel tablosundan maaşı 4950 olanları silin

```
DELETE personel  
WHERE maas =4900;
```

ORA-02292: integrity constraint (SQL\_AHZDXVGZXDBVBVLYOPIBABNDG.PERSONEL\_BILGI\_FK) violated - child record found ORA-06512: at "SYS.DBMS\_SQL", line 1721



## Tekrar Soruları

**SORU 4)** Personel\_bilgi tablosundan 3 veya 4 cocugu olanlari silin

```
DELETE personel_bilgi  
WHERE cocuk_sayisi IN (3,4);
```

**SORU 5)** Persons tablosundan HONDA'da calisip maasi 3500 olanlari silin

```
DELETE personel  
WHERE maas =3500 AND sirket='Honda';
```



## Tekrar Soruları

**SORU 6)** Personel\_bilgi tablosundan dataları geri getirilemeyecek şekilde silin

**TRUNCATE TABLE** personel\_bilgi;

**SORU 7)** Personel tablosundan maasi 4000 ile 5000 arasında olanları silin

**DELETE** personel  
**WHERE** maas **BETWEEN** 4000 **AND** 5000;



## Tekrar Soruları

**SORU 8)** Personel tablosundan maasi 5000 ile 6000 arasında olmayanları silin

```
DELETE personel  
WHERE maas NOT BETWEEN 5000 AND 6000;
```

**SORU 9)** Persone tablosunu geri getirilemeyecek sekilde silin (Oracle sql için)

```
DROP TABLE personel PURGE; HATA VERIR  
Once personel_bilgi tablosunu silin  
DROP TABLE personel_bilgi;  
Persone tablosunu geri getirilemeyecek sekilde silin
```



# SUBQUERIES

**SUBQUERY** baska bir SORGU(query)'nun icinde calisan SORGU'dur.

1) **WHERE**' den sonra kullanilabilir

```
CREATE TABLE personel
```

```
(  
    id int,  
    isim varchar(50),  
    sehir varchar(50),  
    maas int,  
    sirket varchar(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');
```

```
CREATE TABLE sirketler
```

```
(  
    sirket_id int,  
    sirket varchar(20),  
    personel_sayisi int  
);
```

```
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);
```



# SUBQUERIES

1) Personel sayisi 15.000'den cok olan sirketlerin isimlerini ve bu sirkette calisan isimlerini listeleyiniz

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

Subquery sonucu	
ISIM	SIRKET
Ayse Gul	Toyota
Veli Yilmaz	Ford
Ayse Gul	Ford

Query sonucu

2) Sirket\_id'si 101'den buyuk olan sirketlerin maaslarini ve sehirlerini listeleyiniz

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

Subquery sonucu		
SIRKET	SEHIR	MAAS
Hyundai	Ankara	7000
Toyota	Istanbul	1500

Query sonucu

3) Ankara'daki sirketlerin sirket id ve calisan sayilarini listeleyiniz

SIRKET	PERSONEL_SAYISI
Honda	12000
Hyundai	10000
Ford	18000



## SUBQUERIES

- 2) **SELECT'** den sonra kullanılabilir. Ancak SELECT CLAUSE da kullanılan Subquery **SADECE 1 DEGER** donmelidir. Dolayısıyla **SUM**, **COUNT**, **MIN**, **MAX** ve **AVG** gibi fonksiyonlar kullanılır. Bu fonksiyonlara **AGGREGATE FUNCTION** denir.

**SORU 1-** Her sirketin ismini, personel sayısını ve personelin ortalama maasını listeleyen bir QUERY yazın.

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

SIRKET	PERSONEL SAYISI	(SELECT AVG(MAAS) FROM PERSONEL WHERE SIRKETLER.SIRKET=PERSONEL.SIRKET)
Honda	12000	2666.666666666666666666666666666666666666667
Ford	18000	1250
Hyundai	10000	7000
Toyota	21000	1500



# SUBQUERIES

SORU 2- Her sirketin ismini ve personelin aldigı max. maasi listeleyen bir QUERY yazın.

```
SELECT sirket,personel_sayisi,(SELECT MAX(maas)
```

```
FROM personel
```

```
WHERE sirketler.sirket=personel.sirket
```

```
) AS
```

```
sirketteki_Max_Maas,
```

```
FROM sirketler;
```

SIRKET	PERSONEL_SAYISI	SIRKETTEKİ_MAX_MAAS
Honda	12000	3000
Ford	18000	1500
Hyundai	10000	7000
Toyota	21000	1500



# SUBQUERIES

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

```
SELECT sirket_id,sirket,(SELECT COUNT(sehir)
```

```
FROM personel
```

```
WHERE sirketler.sirket=personel.sirket
```

```
bulundugu_sehir_sayisi
```

```
FROM sirketler;
```

SIRKET	PERSONEL SAYISI	BULUNDUGU_SEHIR SAYISI
Honda	12000	3
Ford	18000	2
Hyundai	10000	1
Toyota	21000	1



# SQL

Structured Query Language  
Yapılandırılmış Soru Dili

SUBQUERIES  
EXISTS, IS NULL KOMUTLARI  
ORDER BY, GROUP By



# SUBQUERIES

SORU 4- Her sirketin ismini, personel sayisini ve personelin aldigı max. ve min. maasi 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
```

SIRKET	PERSONEL SAYISI	MAX_MAAST	MIN_MAAST
Honda	12000	3000	2500
Ford	18000	1500	1000
Hyundai	10000	7000	7000
Toyota	21000	1500	1500



# SUBQUERIES

SORU 5- Her sirketin ismini ve personel sayisini ve iscilere odedigi toplam maasi listeleyen bir QUERY yazin.

```
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



# EXISTS 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.

```
CREATE TABLE mart_satislar
(
    urun_id int,
    musteri_isim varchar(50),
    urun_isim varchar(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');
```

```
CREATE TABLE nisan_satislar
(
    urun_id int,
    musteri_isim varchar(50),
    urun_isim varchar(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');
```



# EXISTS CONDITION

Her iki ayda da aynı id ile satılan ürünlerin urun\_id'lerini ve ürünlerin mart ayında alanların isimlerini getiren bir query yazınız..

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

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

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

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



# EXISTS CONDITION

Her iki ayda da satilan urun\_isimleri ayni urunlerin urun\_isim'ini ve urunleri nisan ayinda alanların isimlerini getiren bir query yaziniz..

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

```
SELECT musteri_isim  
FROM nisan_satislar  
WHERE NOT 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	Ayse
Toyota	Mine

URUN_ISIM	MUSTERI_ISIM
Volvo	Yasar



# IS NULL CONDITION

Arama yapılan field'da NULL degeri almis kayitlari getirir.

```
CREATE TABLE insanlar  
(  
    ssn char(9),  
    isim varchar(50),  
    adres varchar(50)  
)
```

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

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

```
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');
```

SSN	NAME	ADDRESS
456789012	-	Bursa
567890123	-	Denizli

SSN	NAME	ADDRESS
456789012	Isim Girilmemis	Bursa
567890123	Isim Girilmemis	Denizli

SSN	NAME	ADDRESS
123456789	Ali Can	Istanbul
234567890	Veli Cem	Ankara
345678901	Mine Bulut	Izmir
456789012	-	Bursa
567890123	-	Denizli

```
SELECT *  
FROM insanlar  
WHERE isim IS NOT NULL;
```

SSN	NAME	ADDRESS
123456789	Ali Can	-
234567890	Veli Cem	Ankara
345678901	Mine Bulut	Izmir



# ORDER BY CLAUSE

ORDER BY komutu belli bir field'a gore NATURAL ORDER olarak siralama yapmak icin kullanilir

ORDER BY komutu sadece SELECT komutu ile kullanilir

CREATE TABLE insanlar

```
(  
    ssn char(9),  
    isim varchar(50),  
    soyisim varchar(50),  
    adres varchar(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 (345678901, 'Veli','Yilmaz', 'Istanbul ');
```

Insanlar tablosundaki datalari adres'e gore siralayin

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

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

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



# ORDER BY CLAUSE

İnsanlar tablosundaki ismi Mine olanları SSN sıralı olarak listeleyin

```
SELECT *
FROM insanlar
WHERE isim='Mine'
ORDER BY ssn;
```

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

**NOT :** Order By komutundan sonra field ismi yerine field numarası da kullanılabilir

İnsanlar tablosundaki soyismi Bulut olanları isim sıralı olarak listeleyin

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

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



# ORDER BY field\_name DESC CLAUSE

Insanlar tablosundaki tum kayitlari SSN numarasi buyukten kucuge olarak siralayin

```
SELECT *
FROM insanlar
ORDER BY ssn DESC;
```

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

Insanlar tablosundaki tum kayitlari isimler Natural sirali, Soyisimler 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
345678901	Veli	Yilmaz	Istanbul
234567890	Veli	Cem	Ankara



# ALIASES

Aliases kodu ile tablo yazdirilirken, field isimleri sadece o ciktigicin degistirilebilir

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

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

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

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

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

ID	ISIM	DOGUM_YERI
123456789	Ali Can	Istanbul
234567890	Veli Cem	Ankara
345678901	Mine Bulut	Izmir

ID	ISIM_VE_DOGUM_YERI
123456789	Ali CanIstanbul
234567890	Veli CemAnkara
345678901	Mine BulutIzmir



# GROUP BY CLAUSE

Group By komutu sonuçları bir veya daha fazla sütuna göre grüplamak için **SELECT** komutuyla birlikte kullanılır

```
CREATE TABLE manav
(
    isim varchar(50),
    Urun_adi varchar(50),
    Urun_miktar int
);
```

```
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);
```

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) Isme gore alınan toplam ürünler bulun

```
SELECT isim, SUM(urun_miktar) AS Alinan_Toplama_Meyve
FROM manav
GROUP BY isim;
```

ISIM	ALINAN_TOPLAM_MEYVE
Veli	10
Ayse	8
Ali	9
Hasan	4



# GROUP BY CLAUSE

2) Urun ismine gore urunu alan toplam kisi sayisi

```
SELECT urun_adi, COUNT(isim) AS UrunuAlanKisiSayisi  
FROM manav  
GROUP BY urun_adi;
```

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

URUN_ADI	URUNU_ALAN_KISI_SAYISI
Elma	4
Uzum	3
Armut	3

URUN_MIKTAR	URUN_MIKTARINI_ALAN_KISI_SAYISI
2	4
5	2
4	1
3	3

3) Alinan kilo miktarina gore musteri sayisi

```
SELECT urun_miktar, COUNT(isim) AS UrunMiktariniAlanKisiSayisi  
FROM manav  
GROUP BY urun_miktar;
```



# SQL

**Structured Query Language**  
**Yapılandırılmış Sorgu Dili**

**GROUP BY , HAVING  
UNION, UNION ALL, INTERSECT, MINUS**

**İPEK BİLGİN**

**Java Full Stack Developer**



# GROUP BY CLAUSE

CREATE TABLE personel

```
(  
    id int,  
    isim varchar(50),  
    sehir varchar(50),  
    maas int,  
    sirket varchar(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');
```

1) Isme gore toplam maaslari bulun

```
SELECT isim, SUM(maas) AS toplam_maas  
FROM personel  
GROUP BY isim;
```

2) sehere gore toplam personel sayisini bulun

```
SELECT sehir, COUNT(isim) AS calisan_sayisi  
FROM personel  
GROUP BY sehir;
```

ISIM	TOPLAM_MAAZ
Hatice Sahin	4500
Veli Sahin	9000
Ali Yilmaz	5500
Mehmet Ozturk	16500

SEHIR	CALISAN_SAYISI
Izmir	1
Bursa	1
Istanbul	2
Ankara	3

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



# GROUP BY CLAUSE

3) Sirketlere gore maasi 5000 liradan fazla olan personel sayisini bulun

```
SELECT sirket, COUNT (*) AS calisan_sayisi  
FROM personel  
WHERE maas>5000  
GROUP BY sirket;
```

SIRKET	CALISAN_SAYISI
Honda	1
Ford	1
Tofas	1

4) Her sirket icin Min ve Max maasi bulun

```
SELECT sirket, MIN (maas) AS en_az_maas, MAX (maas) AS en_fazla_maas  
FROM personel  
GROUP BY sirket;
```

SIRKET	EN_AZ_MAAKS	EN_FAZLA_MAAKS
Honda	3500	5500
Ford	4500	6000
Toyota	4500	4500
Tofas	7000	7000



# HAVING CLAUSE

HAVING, AGGREGATE FUNCTION'lar ile birlikte kullanılan FILTRELEME komutudur.

```
CREATE TABLE personel  
(  
    id int,  
    isim varchar(50),  
    sehir varchar(50),  
    maas int,  
    sirket varchar(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');
```

1) Her sirketin MIN maaslarini eger 2000'den buyukse goster

```
SELECT sirket, MIN (maas) AS en_az_maas  
FROM personel  
GROUP BY sirket  
HAVING MIN (maas) >2000;
```

SIRKET	EN_AZ_MAAS
Honda	3500
Ford	4500
Toyota	4500
Tofas	7000



# HAVING CLAUSE

2) Toplam geliri 10000 liradan fazla olan isimleri gösteren sorgu yazınız

```
SELECT isim, SUM (maas) AS toplam_maas  
FROM personel  
GROUP BY isim  
HAVING SUM (maas) >10000;
```

ISIM	TOPLAM_MAAZ
Mehmet Ozturk	16500

3) Eger bir sehirde calisan personel sayisi 1'den coksa sehir ismini ve personel sayisini veren sorgu yazınız

```
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



# HAVING CLAUSE

- 4) Eger bir sehirde alınan MAX maas 5000'den dusukse sehir ismini ve MAX maasi veren soru yaziniz

```
SELECT sehir, MAX (maas) AS max_maas  
FROM personel  
GROUP BY sehir  
HAVING MAX (maas) <5000;
```

SEHIR	MAX_MAAS
Bursa	4500



# UNION OPERATOR

Iki farkli sorgulamanin sonucunu birlestiren islemidir. Secilen Field SAYISI ve DATA TYPE'i ayni olmalidir.

```
CREATE TABLE personel  
(  
    id int,  
    isim varchar(50),  
    sehir varchar(50),  
    maas int,  
    sirket varchar(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');
```

1) Maasi 3000'den fazla olan sehir veisci isimlerini gosteren sorguyu yaziniz

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

ISCI VEYA SEHIR ISMI	MAAS
Ali Yilmaz	5500
Ankara	4500
Ankara	7000
Bursa	4500
Hatice Sahin	4500
Istanbul	4500
Istanbul	5500
Izmir	6000
Mehmet Ozturk	6000
Mehmet Ozturk	7000
Veli Sahin	4500
Veli Sahin	4500



# UNION OPERATOR

2) Mehmet Ozturk ismindeki personelin aldigı maaşları ve İstanbul'daki personelin maaşlarını bir tabloda gösteren sorgu yazınız

```
SELECT sehir ASisci_veya_sehir_ismi ,maas  
FROM personel  
WHERE sehir='Istanbul'  
UNION  
SELECT isim ASisci_veya_sehir_ismi , maas  
FROM personel  
WHERE isim = 'Mehmet Ozturk';
```

ISCI_VEYA_SEHIR_ISMI	MAAS
Istanbul	4500
Istanbul	5500
Mehmet Ozturk	3500
Mehmet Ozturk	6000
Mehmet Ozturk	7000

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

ORDER BY maas;

ISCI_VEYA_SEHIR_ISMI	MAAS
Mehmet Ozturk	3500
Istanbul	4500
Istanbul	5500
Mehmet Ozturk	6000
Mehmet Ozturk	7000



# UNION OPERATOR

3) Sehirlerde odenen ucreti 3000'den fazla olan ve personelden ucreti 5000'den az olanlari bir tabloda gosteren soruyu yaziniz

```
SELECT sehir ASisci_veya_sehir_ismi , maas  
FROM personel  
WHERE maas>3000  
UNION  
SELECT isim ASisci_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
Veli Sahin	4500



# UNION OPERATOR

## 2 Tablodan Data Birlestirme

Personel isminde bir tablo olusturun. Icinde id, isim, sehir, maas ve sirket field'lari olsun. Id'yi 2.yontemle PK yapin

```
CREATE TABLE personel
```

```
(  
    id int,  
    isim varchar(50),  
    sehir varchar(50),  
    maas int,  
    sirket varchar(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');
```

Personel\_bilgi isminde bir tablo olusturun. Icinde id, tel ve cocuk sayisi field'lari olsun. Id'yi FK yapin ve personel tablosu ile relation kurun

```
CREATE TABLE personel_bilgi
```

```
(  
    id int,  
    tel char(10) UNIQUE ,  
    cocuk_sayisi int,  
    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);
```



# UNION OPERATOR

id'si 12345678 olan personelin Personel tablosundan sehir ve maasini, personel\_bilgi tablosundan da tel ve cocuk sayisini yazdirin

```
SELECT sehir AS Sehir_tel ,maas AS cocuk_sayisi_veya_tel  
FROM personel  
WHERE id='123456789'
```

UNION

```
SELECT tel,cocuk_sayisi  
FROM personel_bilgi  
WHERE id= '123456789';
```

SEHIR_TEL	COCUK_SAYISI_VEYA_TEL
5302345678	5
Istanbul	5500

## NOT : Union islemi yaparken

- 1) Her 2 QUERY'den elde edeceginiz tablolarin sutun sayiları esit olmali
- 2) Alt alta gelecek sutunların data type'ları aynı olmali

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
456789152	Veli Sahin	Ankara	4500	Ford
123456710	Hatice Sahin	Bursa	4500	Honda

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



# UNION ALL OPERATOR

1) Personel tablosundada maasi 5000'den az olan tum isimleri ve maaslari bulunuz

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

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

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



# UNION ALL OPERATOR

2) Ayni sorugu UNION ile iki kere yazarak calistirin

```
SELECT sehir,maas  
FROM personel  
WHERE maas<5000
```

UNION

```
SELECT sehir,maas  
FROM personel  
WHERE maas<5000;
```

SEHIR	MAAS
Ankara	3500
Ankara	4500
Bursa	4500
Istanbul	4500

3) Ayni sorugu UNION ALL ile iki kere yazarak calistirin

```
SELECT sehir,maas  
FROM personel  
WHERE maas<5000;
```

UNION

```
SELECT sehir,maas  
FROM personel  
WHERE maas<5000;
```

SEHIR	MAAS
Istanbul	4500
Ankara	3500
Ankara	4500
Bursa	4500
Istanbul	4500
Ankara	3500
Ankara	4500
Bursa	4500



## UNION ALL OPERATOR

**UNION** islemi 2 veya daha çok **SELECT** isleminin sonuc **KUMELERINI** birlestirmek icin kullanilir, Ayni kayit birden fazla olursa, sadece bir tanesini alir.

**UNION ALL** ise tekrarli elemanlari, tekrar sayisinda yazar.

**NOT :** UNION ALL ile birlestirmelerde de

- 1) Her 2 QUERY'den elde edeceginiz tablolarin sutun sayiları esit olmali
- 2) Alt alta gelecek sutunların data type'ları aynı olmali



# UNION ALL OPERATOR

1) Tabloda personel maasi 4000'den cok olan tum sehirleri ve maaslari yazdirin

```
SELECT sehir,maas  
FROM personel  
WHERE maas>4000;
```

SEHIR	MAAS
Istanbul	5500
Istanbul	4500
Izmir	6000
Ankara	7000
Ankara	4500
Bursa	4500

2) Tabloda personel maasi 5000'den az olan tum isimleri ve maaslari yazdirin

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

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

3) Iki sorguyu UNION ve UNION ALL ile birlestirin

ID	ISIM	SEHIR	MAAS	SIRKET
123456789	Ali Yilmaz	Istanbul	5000	Honda
234567890	Veli Sahin	Istanbul	5000	Toyota
345678901	Mehmet Ozturk	Ankara	4500	Honda
456789012	Mehmet Ozturk	Izmir	6000	Ford
567890123	Mehmet Ozturk	Ankara	6000	Tofas
456715012	Veli Sahin	Ankara	4500	Ford
123456710	Hatice Sahin	Bursa	4500	Honda

SEHIR	MAAS
Ankara	4500
Ankara	7000
Bursa	4500
Hatice Sahin	4500
Istanbul	4500
Istanbul	5500
Izmir	6000
Ankara	7000
Ankara	4500
Bursa	4500
Veli Sahin	4500
Mehmet Ozturk	3500
Veli Sahin	4500
Hatice Sahin	4500



# (oracleSql)->INTERSECT OPERATOR

1) Personel tablosundan Istanbul veya Ankara'da calisanların id'lerini yazdır

```
SELECT id  
FROM personel  
WHERE sehir IN ('Istanbul','Ankara');
```

ID
123456789
234567890
345678901
567890123
456715012

2) Personel\_bilgi tablosundan 2 veya 3 cocugu olanların id lerini yazdırın

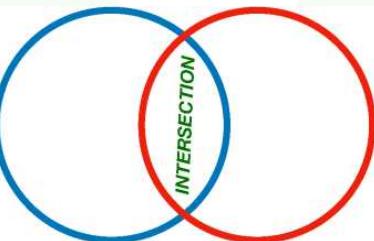
```
SELECT id  
FROM personel_bilgi  
WHERE cocuk_sayisi IN (2,3);
```

ID
345678901
567890123

3) İki soruyu INTERSECT ile birleştirin

SELECT 1

SELECT 2



ID
345678901
456789012
567890123
456789012



# (oracleSql)->INTERSECT OPERATOR

1) Maasi 4800'den az olanlar veya 5000'den cok olanların id'lerini listeleyin

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

2) Personel\_bilgi tablosundan 2 veya 3 cocugu olanların id lerini yazdirin

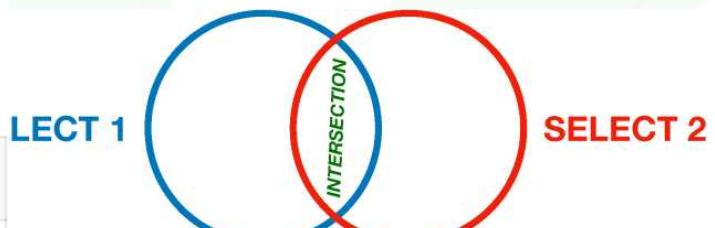
```
SELECT id  
FROM personel_bilgi  
WHERE cocuk_sayisi IN (2,3);
```

3) Iki soruyu INTERSECT ile birlestirin

ID
234567890
345678901
456789012
567890123
456715012
123456710

ID
345678901
456789012
567890123
456789012

ID
345678901
456789012
567890123





# (oracleSql)->INTERSECT OPERATOR

3) Honda,Ford ve Tofas'ta calisan ortak isimde personel varsa listeleyin

```
SELECT isim  
FROM personel  
WHERE sirket='Honda'
```

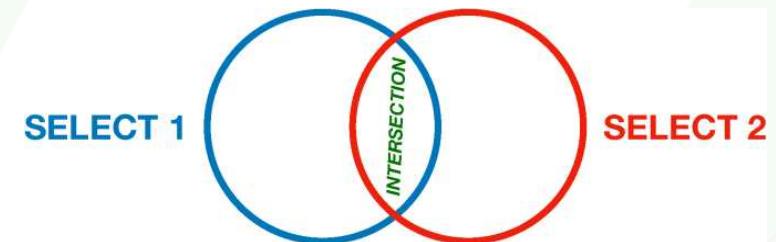
ISIM
Mehmet Ozturk

INTERSECT

```
SELECT isim  
FROM personel  
WHERE sirket='Ford'
```

INTERSECT

```
SELECT isim  
FROM personel  
WHERE sirket='Tofas';
```





# (oracleSql)->MINUS OPERATOR

1) 5000'den az maas alip Honda'da calismayanlari yazdirin

```
SELECT isim,sirket  
FROM personel  
WHERE maas<5000
```

MINUS

```
SELECT isim,sirket  
FROM personel  
WHERE sirket='Honda'
```

2) Ismi Mehmet Ozturk olup Istanbul'da calismayanların isimlerini ve sehirlerini listeleyin

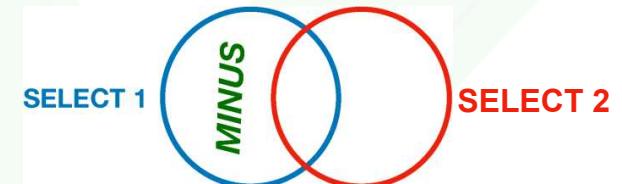
```
SELECT isim,sehir  
FROM personel  
WHERE isim='Mehmet Ozturk'
```

MINUS

```
SELECT isim,sirket  
FROM personel  
WHERE sehir='Istanbul';
```

ISIM	SIRKET
Veli Sahin	Ford
Veli Sahin	Toyota

ISIM	SEHIR
Mehmet Ozturk	Ankara
Mehmet Ozturk	Izmir





# SQL

Structured Query Language  
Yapılandırılmış Sorgu Dili

JOINS  
LIKE, NOT LIKE CONDITIONS

iPEK BİLGİN  
Java Full Stack Developer



# JOINS

2 Tablodaki datalari Birlestirmek icin kullanilir.

Su ana kadar gordugumuz Union,Intersect ve Minus soru sonuclari icin kullanilir  
Tablolar icin ise **JOIN** kullanilir

5 Cesit Join vardir:

- 1) **INNER JOIN** iki Tablodaki ortak datalari gosterir
- 2) **LEFT JOIN** Ilk datada olan tum recordlari gosterir
- 3) **RIGHT JOIN** Ikinci tabloda olan tum recordlari gosterir
- 4) **JOIN** Iki tablodaki tum recordlari gosterir
- 5) **SELF JOIN** Bir tablonun kendi icinde Join edilmesi ile olusur.



# INNER JOINS

```
CREATE TABLE sirketler
(
    sirket_id int,
    sirket_isim varchar(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');
```

SIRKET_ID	SIRKET_ISIM
100	Toyota
101	Honda
102	Ford
103	Hyundai

```
CREATE TABLE
siparisler
(
    siparis_id int,
    sirket_id int,
    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



# INNER JOINS

SORU) İki Tabloda sirket\_id'si aynı olanların sirket\_ismi, siparis\_id ve siparis\_tarihleri ile yeni bir tablo olusturun

TABLE 1

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

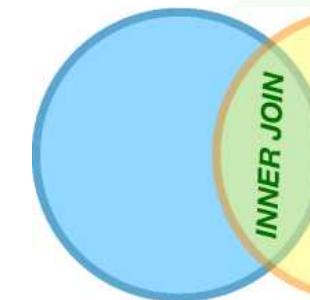


TABLE 2

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

NOT :

- 1) Select'ten sonra tabloda gormek istediginiz sutunlari yazarken **Tablo\_adi.field\_adi** seklinde yazin
- 2) From'dan sonra tablo ismi yazarken **1.Tablo ismi + INNER JOIN + 2.Tablo ismi** yazmaliyiz
- 3) Join'i hangi kurala gore yapacagini belirtmelisiniz. Bunun icin **ON+ kuralimiz** yazilmali



# LEFT JOINS

TABLE 1

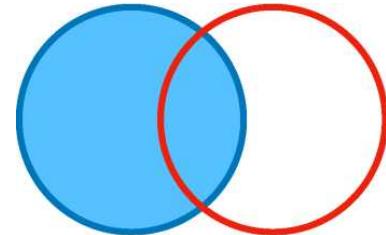


TABLE 2

```
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	-	-

## NOT :

- 1) Left Join'de ilk tablodaki tum record'lar gosterilir.
- 2) Ilk tablodaki datalara 2.tablodan gelen ek datalar varsa bu ek datalar ortak datalar icin gosterilir ancak ortak olmayan datalar icin o kisimlar bos kalir
- 3) Ilk yazdiginiz Tablonun tamamini aligi icin hangi tabloyu istedigimize karar verip once onu yazmaliyiz



# RIGHT JOINS

TABLE 1

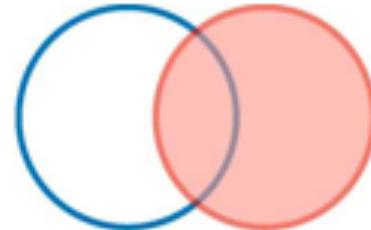


TABLE 2

```
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

## NOT :

- 1) Right Join'de ikinci tablodaki tum record'lar gösterilir.
- 2) Ikinci tablodaki datalara 1.tablodan gelen ek datalar varsa bu ek datalar ortak datalar icin gösterilir ancak ortak olmayan datalar icin o kisimlar bos kalir



# JOINS

```
SELECT sirketler.sirket_isim, siparisler. siparis_id, siparisler. siparis_tarihi  
FROM sirketler 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	-	-

## NOT :

- 1) JOIN de iki tabloda var olan tum record'lar gosterilir.
- 2) Bir tabloda olup otekinde olmayan data'lar bos kalir



# SELF JOINS

CREATE TABLE personel

```
(  
    id int,  
    isim varchar(20),  
    title varchar(60),  
    yonetici_id int  
)
```

```
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);
```

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

Her personelin yanina yonetici ismini yazdiran bir tablo olusturun

```
SELECT p1.isim AS personel_ismi, p2.isim AS yonetici_ismi  
FROM personel p1  
INNER JOIN personel p2  
ON p1.yonetici_id = p2.id;
```

PERSONEL_ISMI	YONETICI_ISMI
Ali Can	Veli Cem
Veli Cem	Ayse Gul
Ayse Gul	Fatma Can



# LIKE Condition

LIKE condition WHERE ile kullanilarak SELECT, INSERT, UPDATE, veya DELETE statement ile calisan wildcards'a izin verir.. Ve bize pattern matching yapma imkani verir.

```
CREATE TABLE musteriler
(
id int UNIQUE,
isim varchar(50) NOT
NULL,
gelir int
);
```

1) % => 0 veya birden fazla karakter belirtir

SORU : 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

ID	ISIM	GELIR
1001	Ali	62000
1002	Ayse	57500
1003	Feride	71000
1004	Fatma	42000
1005	Kasim	44000



## LIKE Condition

SORU : Ismi e harfi ile biten musterilerin isimlerini ve gelir'lerini yazdiran QUERY yazin

```
SELECT isim,gelir  
FROM musteriler  
WHERE isim LIKE '%e';
```

ISIM	GELIR
Ayse	57500
Feride	71000

SORU : Isminin icinde er olan musterilerin isimlerini ve gelir'lerini yazdiran QUERY yazin

```
SELECT isim,gelir  
FROM musteriler  
WHERE isim LIKE '%er%';
```

ISIM	GELIR
Feride	71000



## LIKE Condition

2) \_ => sadece bir karakteri gösterir.

SORU : Ismi 5 harfli olup son 4 harfi atma olan musterilerin tum bilgilerini yazdiran QUERY yazin

```
SELECT *  
FROM musteriler  
WHERE isim LIKE '_atma';
```

SORU : Ikinci harfi a olan musterilerin tum bilgilerini yazdiran QUERY yazin

```
SELECT *  
FROM musteriler  
WHERE isim LIKE '_a%';
```

SORU : Ucuncu harfi s olan musterilerin tum bilgilerini yazdiran QUERY yazin

```
SELECT *  
FROM musteriler  
WHERE isim LIKE '__s%';
```

ID	ISIM	GELIR
1004	Fatma	42000

ID	ISIM	GELIR
1004	Fatma	42000
1005	Kasim	44000

ID	ISIM	GELIR
1002	Ayse	57500
1005	Kasim	44000



## LIKE Condition

SORU : Ucuncu harfi s olan ismi 4 harfli musterilerin tum bilgilerini yazdiran QUERY yazin

```
SELECT *
FROM musteriler
WHERE isim LIKE '__s_';
```

ID	ISIM	GELIR
1002	Ayse	57500

SORU : Ilk harfi F olan en az 4 harfli musterilerin tum bilgilerini yazdiran QUERY yazin

```
SELECT *
FROM musteriler
WHERE isim LIKE 'F_%_%_%';
```

ID	ISIM	GELIR
1003	Feride	71000
1004	Fatma	42000

SORU : 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



# LIKE Condition

3) [ ] REGEXP\_LIKE => sadece bir karakteri gosterir.

CREATE TABLE kelimeler

```
(  
    id int UNIQUE,  
    kelime varchar(50) NOT NULL,  
    Harf_sayisi int  
)
```

```
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);
```

SORU : Ilk harfi h,son harfi t olup 2.harfi a veya i olan 3 harflili kelimelerin tum bilgilerini yazdiran QUERY yazin

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



# LIKE Condition

SORU : İlk harfi h, son harfi t olup 2.harfi a ile k arasında olan 3 harfli kelimelerin tüm bilgilerini yazdırın QUERY yazın

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

SORU : İçinde m veya i olan kelimelerin tüm bilgilerini yazdırın QUERY yazın

```
SELECT *
FROM kelimeler
WHERE REGEXP_LIKE (kelime, '[mi](*) ');
[a|n] de
olur
```

SORU : a veya s ile başlayan kelimelerin tüm bilgilerini yazdırın QUERY yazın

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

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

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

ID	KELIME	HARF_SAYISI
1005	adem	4
1006	selim	5



# LIKE Condition

SORU : m veya f ile biten kelimelerin tum bilgilerini yazdiran QUERY yazin

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

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



## NOT LIKE Condition

SORU 1 : ilk harfi h olmayan kelimelerin 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

SORU 2 : a harfi icermeyen kelimelerin 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



## NOT LIKE Condition

SORU 3 : 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

SORU 4 : 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

Tablolari yazdirirken buyuk harf, kucuk harf veya ilk harfleri buyuk digerleri kucuk harf yazdirmak icin kullaniriz

```
SELECT UPPER(kelime)  
FROM kelimeler;
```

UPPER(KELIME)
HOT
HAT
HIT
HBT
HCT
ADEM
SELIM
YUSUF

```
SELECT  
LOWER(kelime)  
FROM kelimeler;
```

LOWER(KELIME)
hot
hat
hit
hbt
hct
adem
selim
yusuf

```
SELECT  
INITCAP(kelime)  
FROM kelimeler;
```

INITCAP(KELIME)
Hot
Hat
Hit
Hbt
Hct
Adem
Selim
Yusuf



# DISTINCT

```
SELECT DISTINCT urun_isim  
FROM musteri_urun;
```

URUN_ISIM
Elma
Portakal
Kaysi
Armut

```
SELECT DISTINCT musteri_isim  
FROM musteri_urun;
```

MUSTERI_ISIM
Veli
Ayse
Elif
Adem
Ali

```
SELECT COUNT(DISTINCT urun_isim) AS urun_cesit_sayisi  
FROM musteri_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

Tablodan kaç farklı meyve vardır ?

URUN_CESIT_SAYISI
4



# oracleSql->FETCH NEXT (SAYI) ROW ONLY- OFFSET MYSQL->LIMIT

1) Tabloyu urun\_id ye gore siralayiniz

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	Ayse	Armut
40	Veli	Kaysi

2) Sirali tablodan ilk 3 kaydi listeleyin

```
SELECT *
FROM musteri_urun
ORDER BY urun_id
(ORACLE)->FETCH NEXT 3 ROW ONLY;
(MYSQL)->LIMIT 3;
```

URUN_ID	MUSTERI_ISIM	URUN_ISIM
10	Ali	Portakal
10	Adem	Portakal
10	Ali	Portakal

3) Sirali tablodan 4. kayittan 7.kayida kadar olan kayitlari listeleyin

```
SELECT *
FROM musteri_urun
ORDER BY urun_id
(ORACLE)->OFFSET 3 ROW FETCH NEXT 4 ROW ONLY;
(MYSQL)->LIMIT 4,4;
```

URUN_ID	MUSTERI_ISIM	URUN_ISIM
20	Veli	Elma
20	Elif	Elma
20	Ali	Elma
30	Ayse	Armut



# SQL

**Structured Query Language**  
**Yapılandırılmış Sorgu Dili**

**PIVOT, ALTER  
INTERVIEW QUESTIONS**

**iPEK BİLGİN**  
**Java Full Stack Developer**



# PIVOT CLAUSES

```
CREATE TABLE musteri_urun
(
    urun_id int,
    musteri_isim varchar(50),
    urun_isim varchar(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 (SELECT urun_isim, musteri_isim FROM musteri_urun)
PIVOT (COUNT(urun_isim) FOR urun_isim IN ('Portakal', 'Elma', 'Kaysi', 'Armut'));
```

MUSTERI_ISIM	'Portakal'	'Elma'	'Kaysi'	'Armut'
Veli	0	1	1	0
Ayse	0	0	0	1
Elif	0	1	0	0
Adem	1	0	0	0
Ali	2	1	0	0

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

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



# ALTER TABLE STATEMENT

**ALTER TABLE** statement tabloda **add**, **modify**, veya **drop/delete columns** islemleri icin kullanilir.

**ALTER TABLE** statement tablolari yeniden isimlendirmek icin de kullanilir.

**CREATE TABLE** personel

```
(  
    id int,  
    isim varchar(50),  
    sehir varchar(50),  
    maas int,  
    sirket varchar(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');
```

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



# ALTER TABLE STATEMENT

## 1) ADD default deger ile tabloya bir sutun ekleme

```
ALTER TABLE personel  
ADD ulke_isim varchar(20) DEFAULT 'Turkiye';
```

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

## 2) Tabloya birden fazla sutun ekleme

```
ALTER TABLE personel  
ADD (cinsiyet varchar(20) , yas int);
```

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



# ALTER TABLE STATEMENT

## 3) DROP tablodan sutun silme

ALTER TABLE personel  
DROP COLUMN yas;

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

## 4) RENAME COLUMN sutun adi degistirme

ALTER TABLE personel  
RENAME COLUMN ulke\_isim TO ulke\_adi;

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



# ALTER TABLE STATEMENT

## 5) RENAME tablonun ismini degistirme

ALTER TABLE personel  
RENAME TO isciler;

## 6) MODIFY sutunlarin ozelliklerini degistirme

ALTER TABLE isciler  
MODIFY ulke\_adi varchar(30) NOT NULL;

Constraints		
Constraint	Type	Condition
SYS_C0040949308	Check	"ULKE_ADI" IS NOT NULL
PERSONEL_PK	Primary Key	-

ISCILER

Table  
Status: Valid  
Created 17 minutes ago

Columns			
#	Column	Type	Length
1	ID	NUMBER	22
2	ISIM	VARCHAR2	50
3	SEHIR	VARCHAR2	50
4	MAAS	NUMBER	22
5	SIRKET	VARCHAR2	20
6	ULKE_ADI	VARCHAR2	30
7	CINSIYET	VARCHAR2	20



# INTERVIEW QUESTION

**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');
```

**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');
```



# INTERVIEW QUESTION

1) Her iki tablodaki ortak id'leri ve personel tablosunda bu id'ye sahip isimleri listeleyen query yaziniz

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

ISIM	ID
Johnny Walk	123456789
Teddy Murphy	456789012
Brad Pitt	456789012

2) Her iki tablodaki ortak id ve isme sahip kayitlari listeleyen query yaziniz

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

ISIM	ID
Brad Pitt	456789012

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



# INTERVIEW QUESTION

3) Personel tablosunda kac farkli sehirden personel var?

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

SEHIR SAYISI
5

4) Personel tablosunda id'si cift sayi olan personel'in tum bilgilerini listeleyen Query yaziniz

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

ID	ISIM	SEHIR	MAAS	SIRKET
456789012	Teddy Murphy	Virginia	1000	GOOGLE
456789012	Brad Pitt	Texas	1500	TD BANK



# INTERVIEW QUESTION

5) Personel tablosunda kac tane kayit oldugunu gosteren query yazin

```
SELECT COUNT(*)  
FROM personel;
```

COUNT(*)
6

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

KAYIT_SAYISI
6

6) Isciler tablosunda en yuksek maasi alan kisinin tum bilgilerini gosteren query yazin

Max Maas    

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

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

```
FROM isciler);
```

ID	ISIM	SEHIR	MAAS	SIRKET
567890123	Eddie Murphy	Texas	7000	MICROSOFT



# INTERVIEW QUESTION

7) Personel tablosunda en dusuk maasi alan kisinin tum bilgilerini gosteren query yazin

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

ID	ISIM	SEHIR	MAAS	SIRKET
456789012	Teddy Murphy	Virginia	1000	GOOGLE

8) Isciler tablosunda ikinci en yuksek maasi maasi gosteren query yazin

```
SELECT MAX(maas)
FROM personel
WHERE maas <> (SELECT MAX(maas))
```

```
FRC MAX(MAAS) |;
      2500
```



# INTERVIEW QUESTION

9) Isciler tablosunda ikinci en dusuk maasi alan iscinin tum bilgilerini gosteren query yazin

```
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

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

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



# INTERVIEW QUESTION

10) Isciler tablosunda en yüksek maasi alan iscinin disindaki tum iscilerin, tum bilgilerini gosteren query yazin

```
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