

---

# BUGUNKU KONUMUZ

- 1) SQL Nedir ?
- 2) Database Nedir ?
- 3) Genel Database Kavramlari

# DATABASE

```
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 surname");  
        String email = scan.nextLine();  
  
        System.out.println("Enter your password");  
        String password = scan.nextLine();  
  
        scan.close();  
    }  
}
```

## Kaydol

Hızlı ve kolaydır.

Doğum Tarihi ?

1

Eki

2020

Cinsiyet ?

Kadın

Erkek

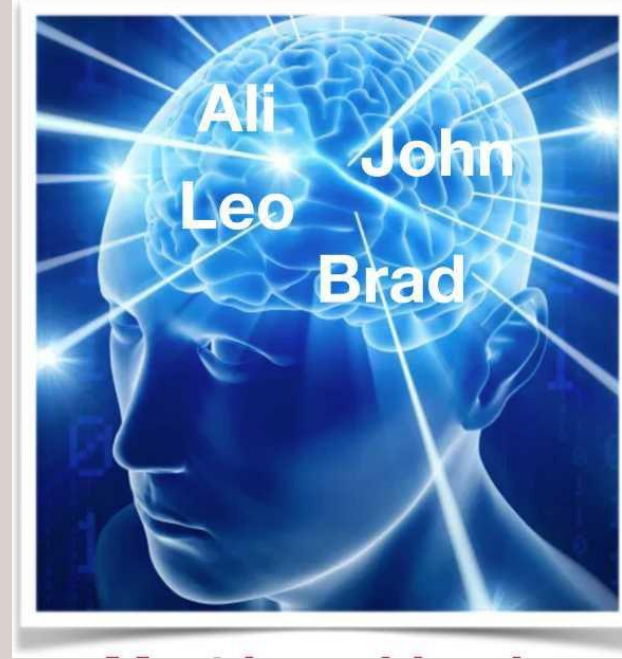
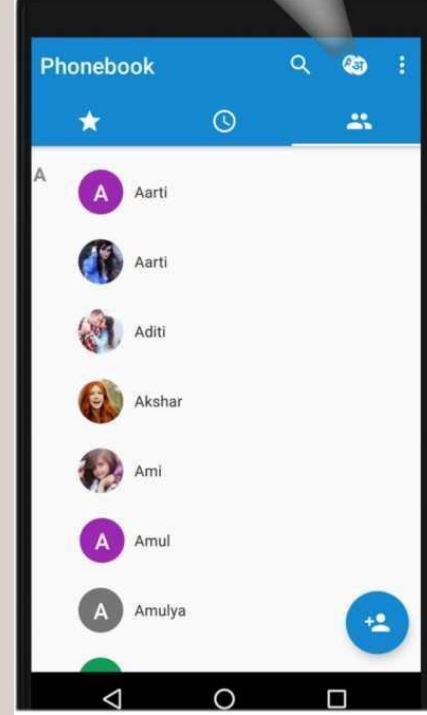
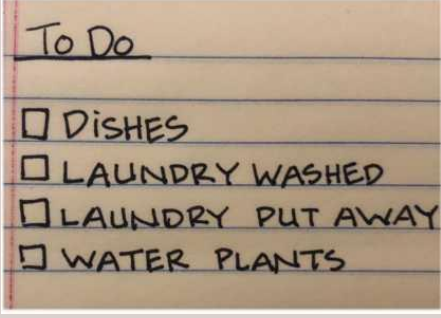
Özel

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

Kaydol



# DATABASE (VERİTABANI) NEDİR?



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

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

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

# DATABASE'IN FAYDALARI NELERDİR

- 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.

Doğum Tarihi ?

▼  ▼  ▼

Cinsiyet ?

☐ Kadın ☐ Erkek ☐ Özel

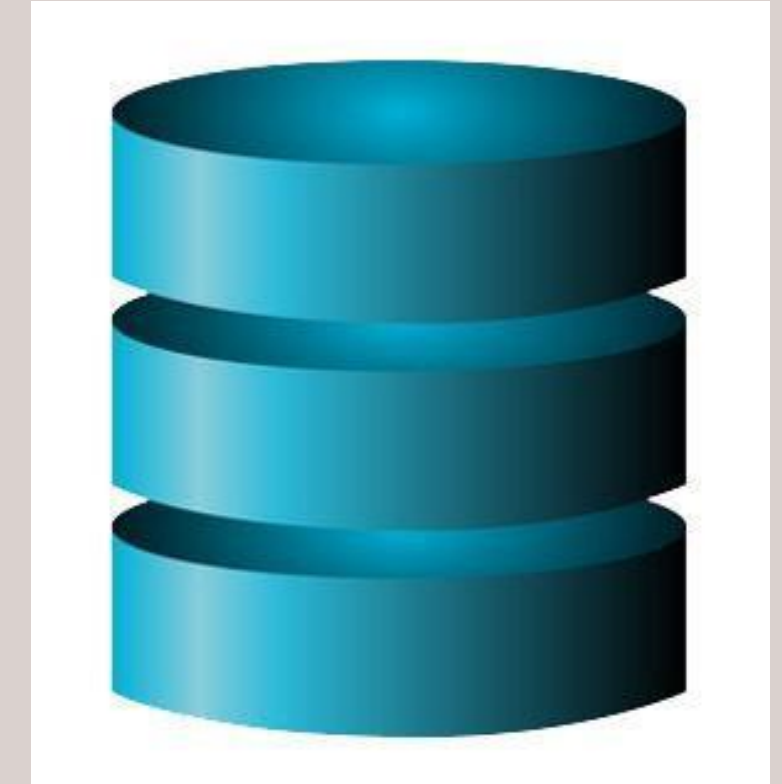
Kaydol düğmesine tıklayarak, Koşullarımızı, Veri İlkemizi ve Çerezler İlkemizi kabul etmiş olursun. Bizden SMS Bildirimleri alabilir ve bu bildirimleri istediğin zaman durdurabilirsin.

**Kaydol**

User Interface



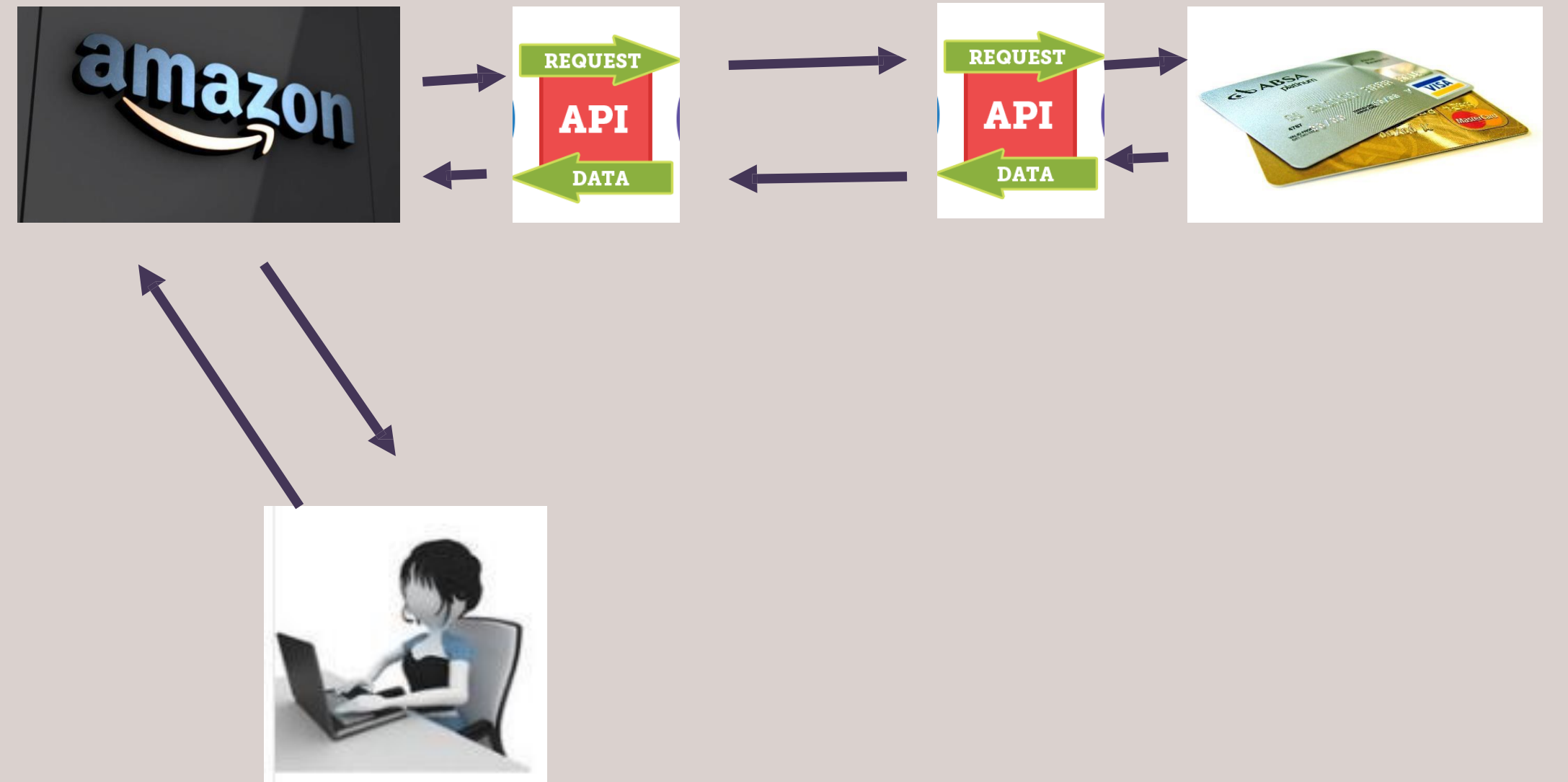
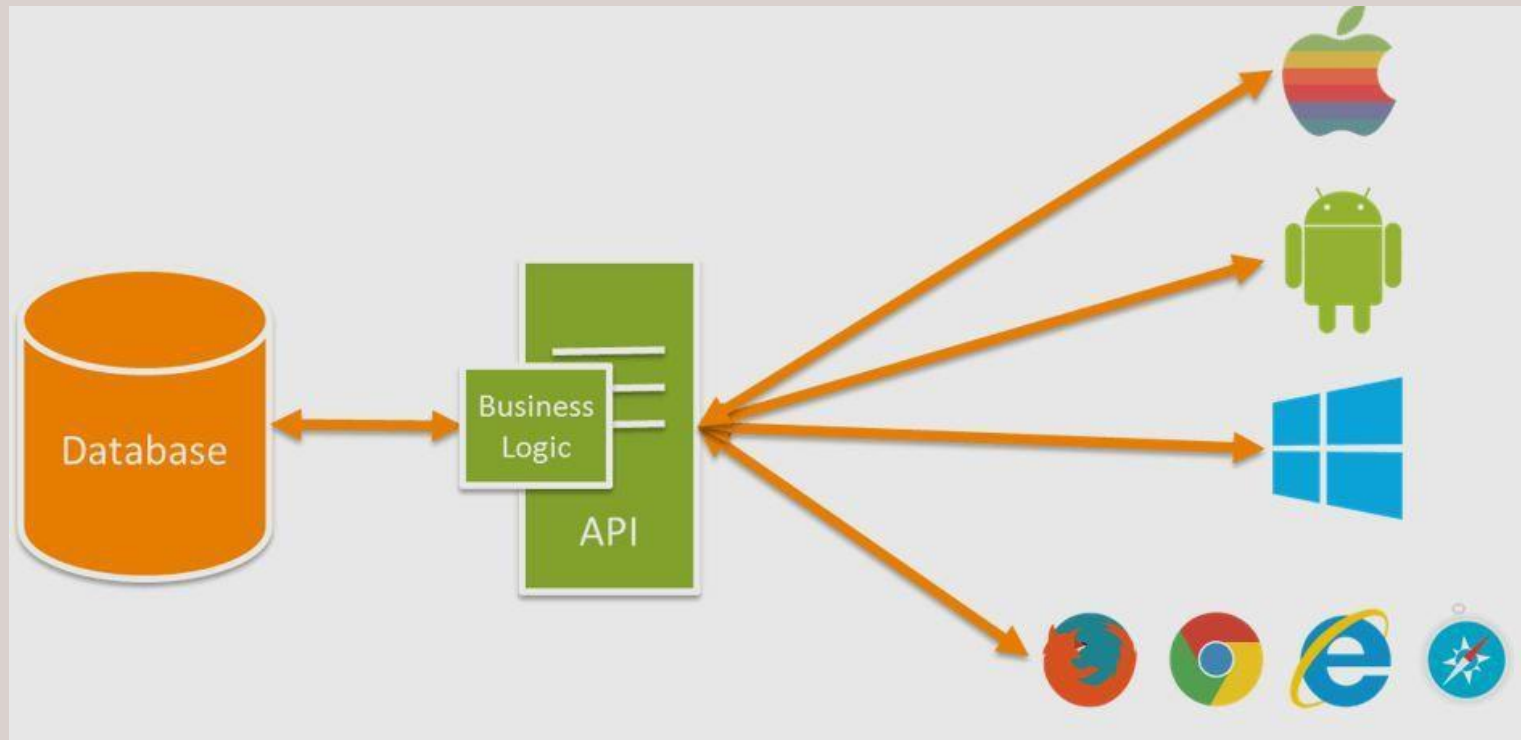
API



Database

# API

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



---

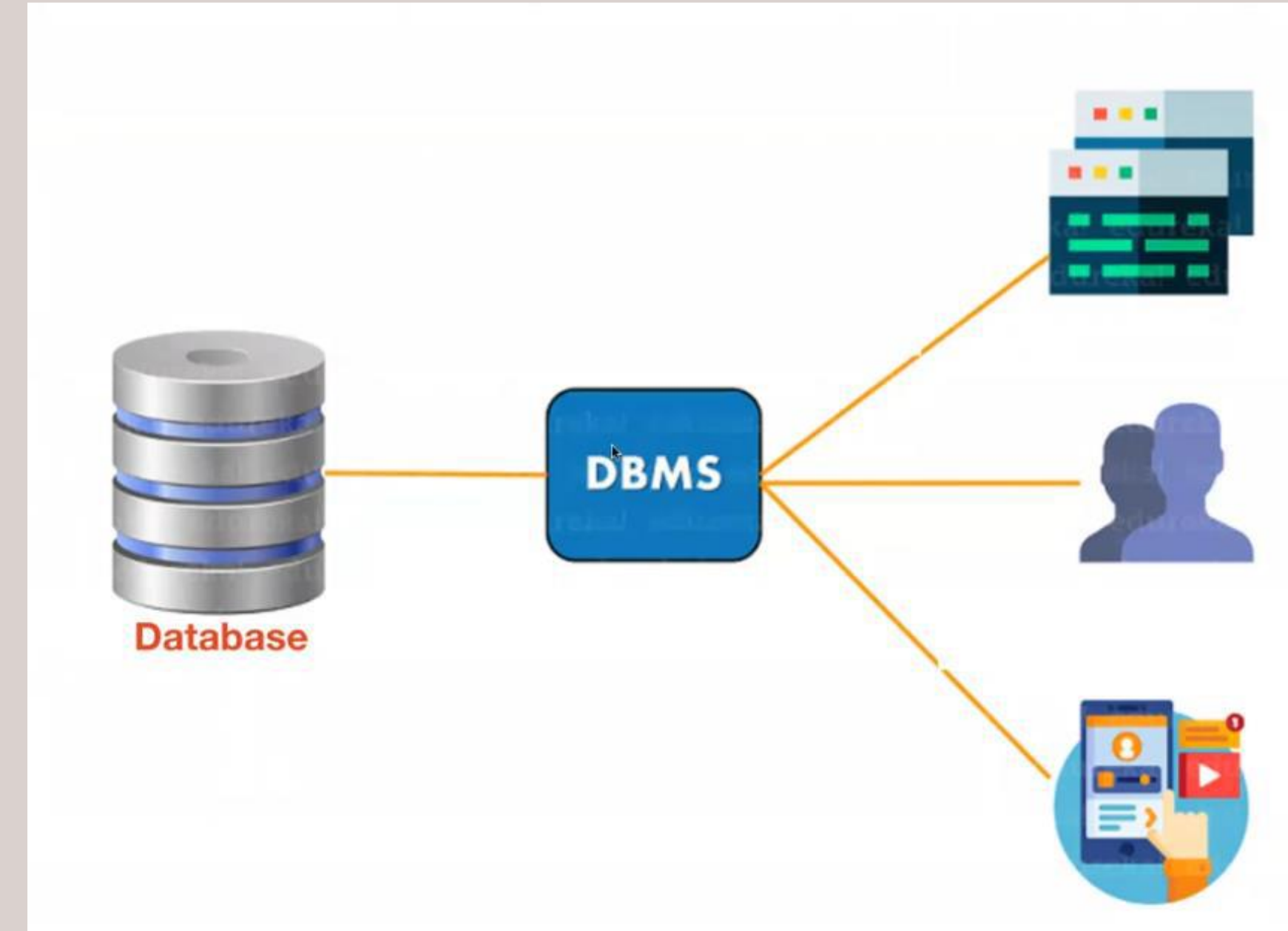
## END To END (E2E) Testing

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

## Database Management System (DBMS)

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

- Database'e erişimi düzenler
- Create, Read, Update, Delete işlemlerini düzenler
- Data güvenliğini sağlar
- Sorgular oluşturur ve sorgular iletilir,
- Raporlar oluşturur ve raporları işletir,
- Uygulamayı kontrol eder
- Diğer uygulamalarla (Application) iletişimi sağlar.





# Cok Kullanilan VTYS(Veri Tabani Yonetim Sistemleri)



**SQL Server** : Microsoft tarafından geliştirilmiştir

**Negatif:** Pahalı – Kurumsal Kullanıcılar için binlerce dolar ödenmesi gereklidir

**Pozitif** : Zengin bir **user interface**'e sahip ve çok büyük verilerin kullanılmasında sorunsuz çalışır.



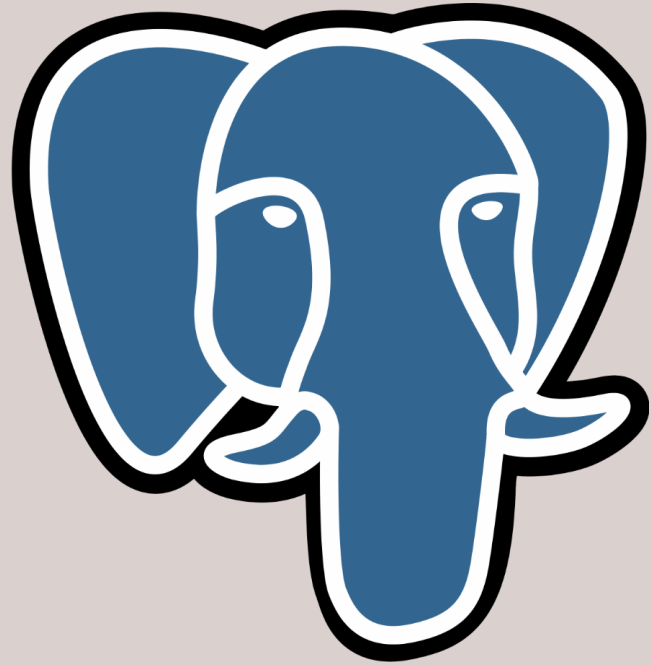
**MySQL Server** : İsveçli MySQL firması tarafından geliştirildi.  
2010'da Oracle satın aldı

**Negatif:** Eszamanlı çok fazla işlem girildiğinde çalışmayı durdurabilir.

**Pozitif:** Açık kaynak. Online destek ve ücretsiz çok fazla doküman var

---

# Cok Kullanilan VTYS(Veri Tabani Yonetim Sistemleri)



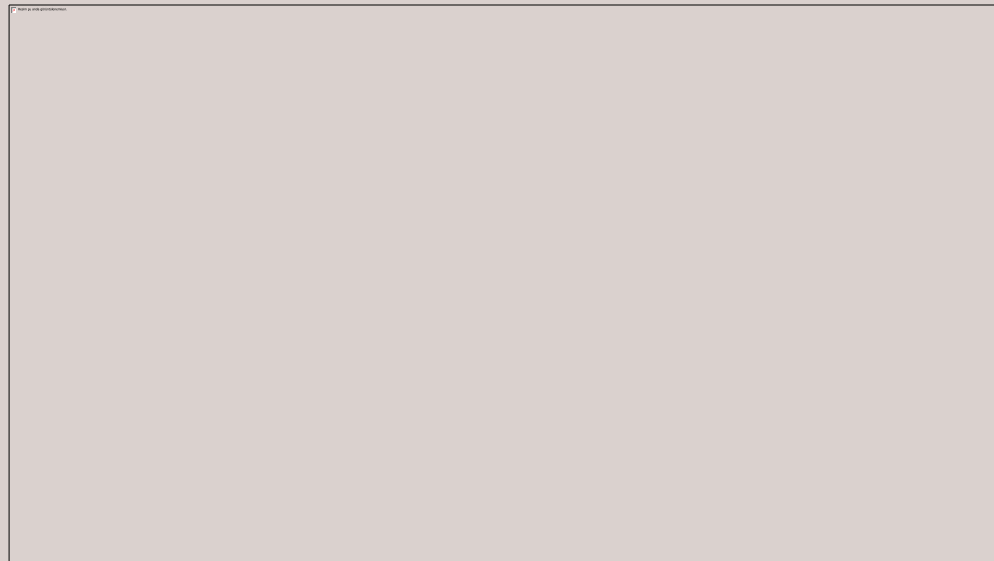
**PostgreSQL Server** : Created by a computer science professor Michael Stonebraker.

Yeni nesil olarak ortaya cikti. Kisisellestirme mumkundur, zor gorevler icin ideal olabilir.

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

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

**Pros:** PL/SQL yuksek guvenlik seviyesi saglar ve Object-Oriented Programing'e uyumludur



# TABLULAR (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

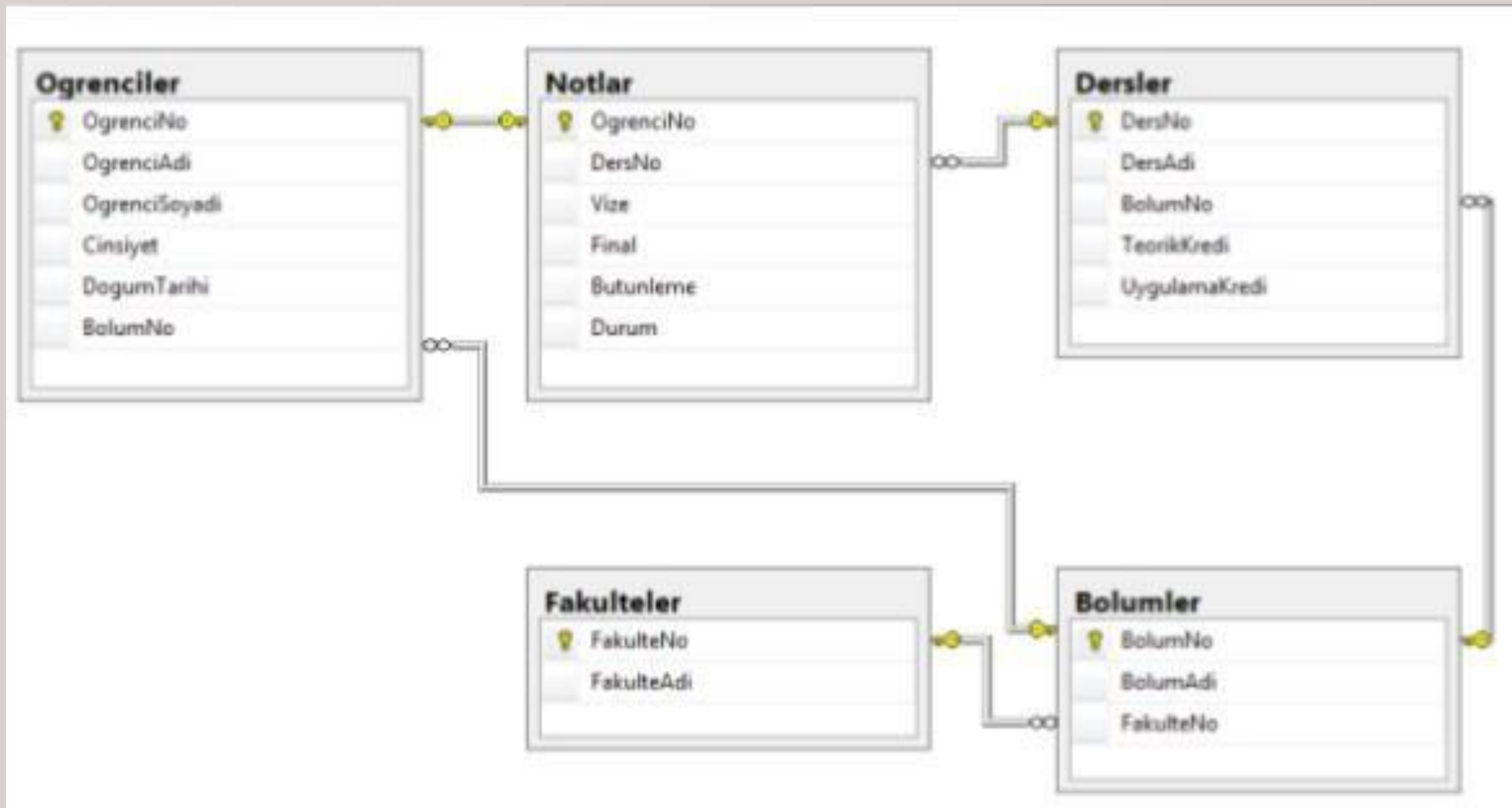
Column (Field) ^====

Column (Field) ^====

Column (Field) ^====

Column (Field) ^====

# RELATIOANAL DATABASES (ILISKILI TABLOLAR)





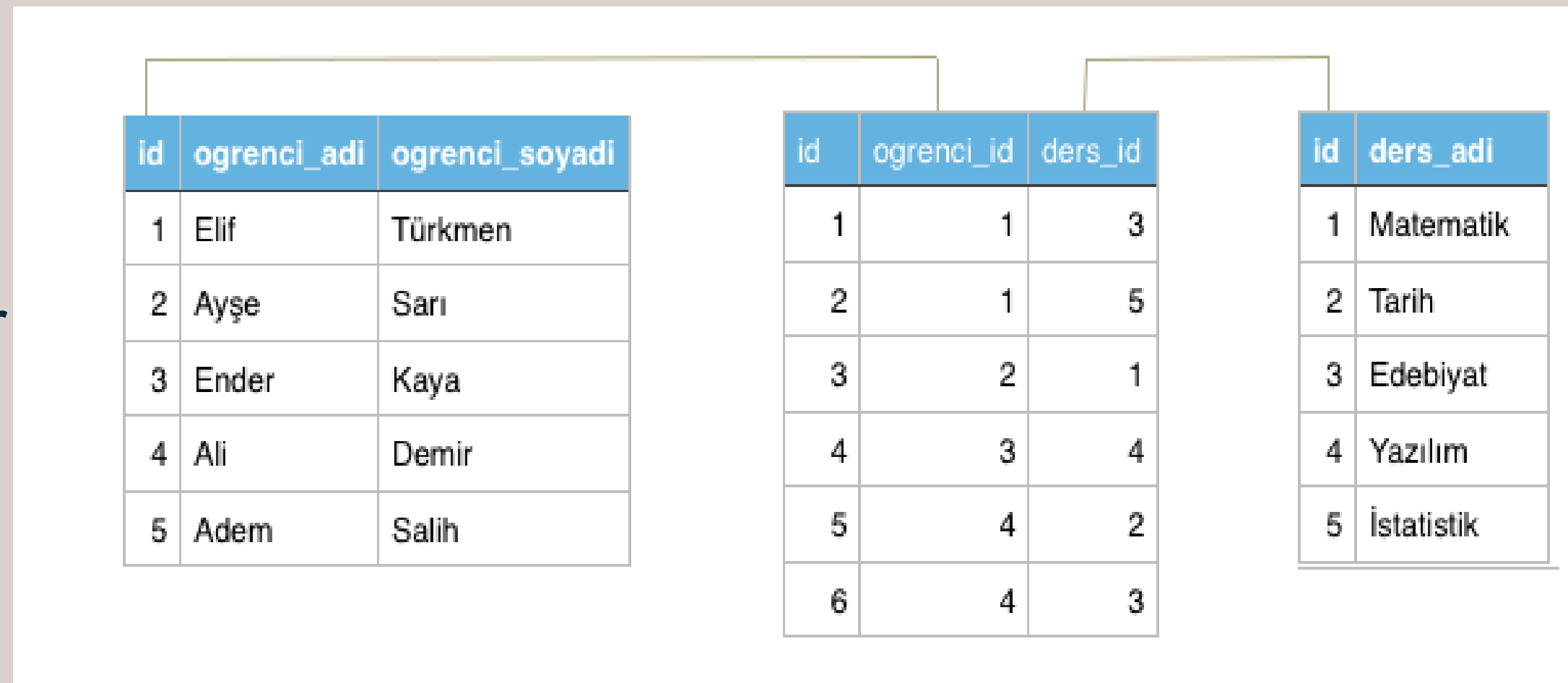
# RELATIONAL DATABASES (ILISKILI TABLOLAR)

➤ **SQL tablolar** datalari iliskili tablolarda depolar.

➤ Tablolar arasi iliskiler net olmalidir.

➤ Tablolar arasi gecis kolay olmalidir

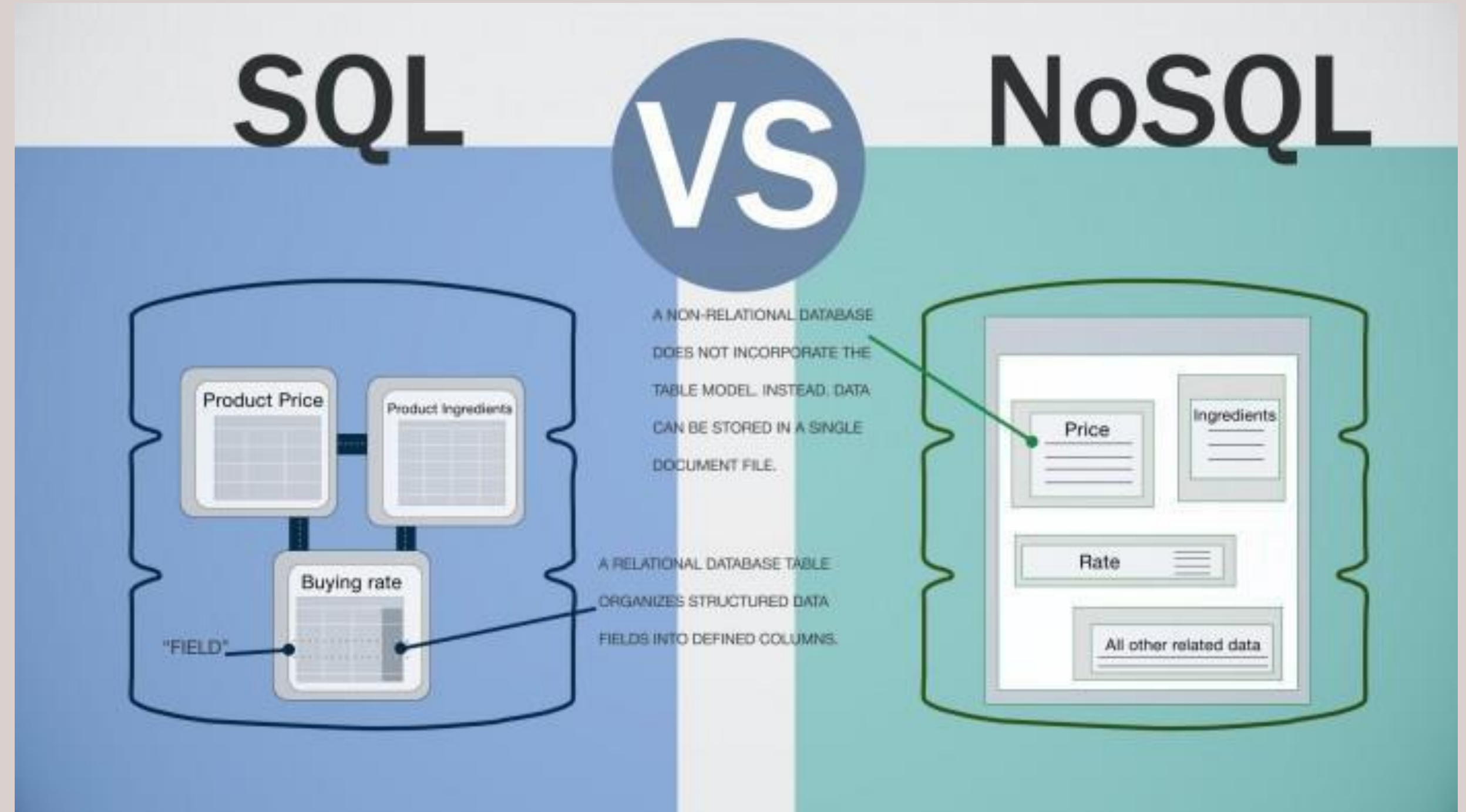
➤ Tablolarin ve iliskilerin butunune SCHEMA denir



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

# Non Relational Databases(non-SQL Database)

**SQL** veritabanı verilerle ilgilenirken Yapısal Sorgu Dili kullanır. Veri yapısını belirlemek için önceden tanımlanmış şemalar gerektirir.



**NoSQL** veritabanı verilerle çalışırken Yapılandırılmamış Sorgu Dili kullanır.

---

# SQL Komutlari

## SQL komutlari 4 ana gruba ayrilir:

### 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 komutlari 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 Komutlari

## 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 kullanıcılara 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.

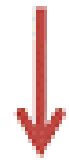
---



# Primary Key

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

Primary Keys

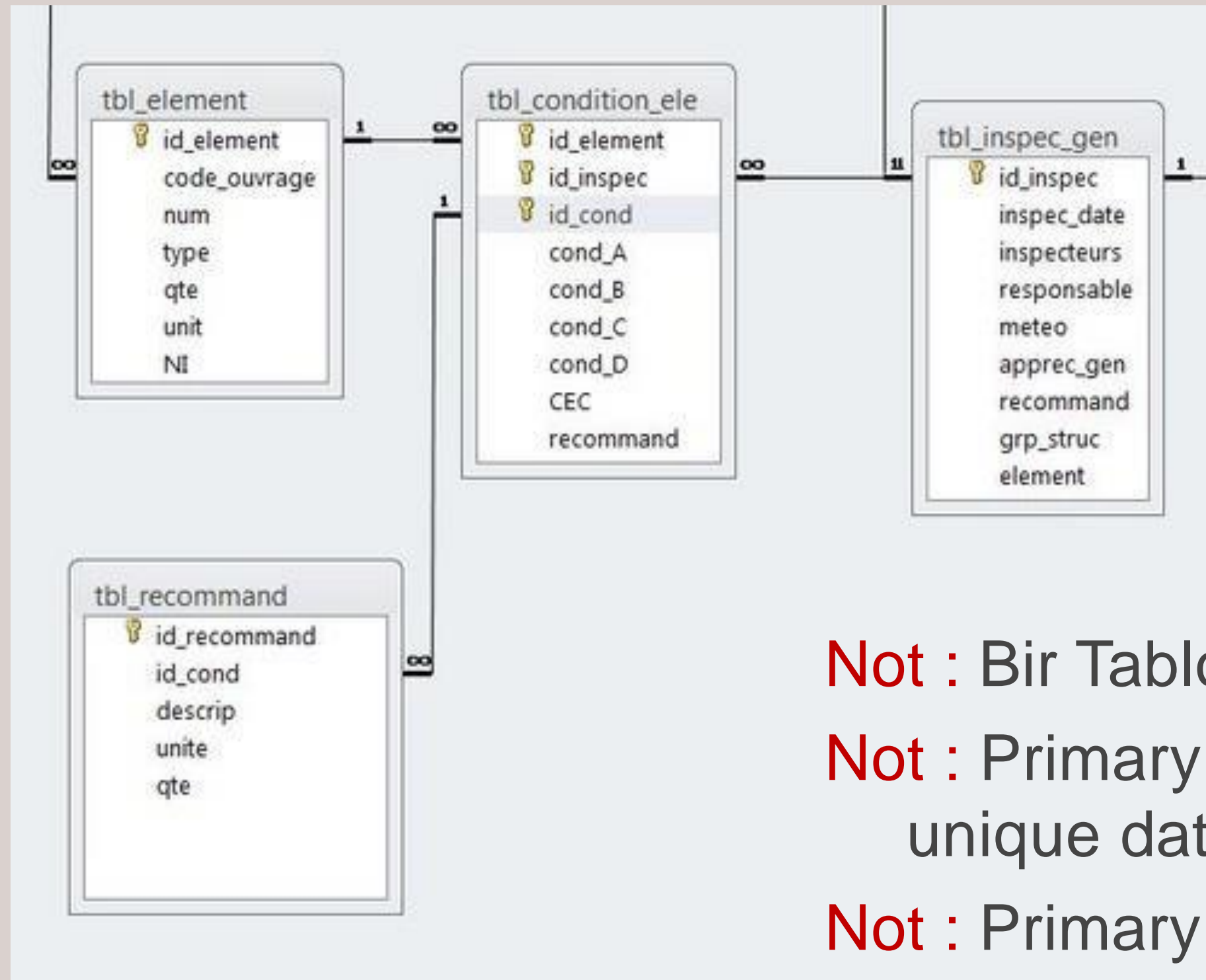


<u>StudentId</u>	firstName	lastName	courseId
L0002345	Jim	Black	C002
L0001254	James	Harradine	A004
L0002349	Amanda	Holland	C002
L0001198	Simon	McCloud	S042
L0023487	Peter	Murray	P301
L0018453	Anne	Norris	S042

Satırlara ait değerlerin karışmaması adına bu alana ait bilginin tekrarlanmaması gerekir.

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 1 tane primary Key olur.

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

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

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

# Primary Key

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

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 sayisal degerlere **Surrogate 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 arasinda relation olusturmak icin kullanilir  
**Foreign Key** baska bir tablodaki Primary Key ile iliskilendirilmis olmalidir



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	NULL

**Child Table**

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 tanimlanan field'da **tekrarlar** olabilir

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



# Foreign and Primary Key

**Note:** Foreign key Tablonun kendi icinde bir relation olusturabilir.

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 manager'i kimdir?
- 2) Angela Martin'in Job\_Name'i nedir ?
- 3) Manual Tester'lerin ortalama Salary'si ne kadardir ?
- 4) En yuksek Salary'yi alan kisinin Job\_Name'i nedir?

# SQL Composite Key

Job_ID	Job_Name
2	SDET
3	Manuel Tester
1	QA Led
Job Table	

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

Job_Id	Name	Company
2	Mark Eye	RCG
3	John Ted	RCG
1	Mark Eye	Signature
1	John Ted	Info Log
1	Cory Al	Info Log
2	Angela Star	Signature
Company Table		

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

Tek basına bir kolon **Primary Key** olma özelliklerini taşıyamıyorsa, bu özellikleri elde etmek için birden fazla kolon birleştirilerek Primary oluşturulur

---

# “UNIQUE KEY” & “PRIMARY KEY”

“UNIQUE KEY” ve “PRIMARY KEY” arasındaki farklar

## Primary Key

Bir Tabloda 1 tane olur  
NULL değer Kabul etmez

## Unique Key

Bir tabloda birden fazla olabilir  
NULL değeri Kabul eder

“UNIQUE KEY” ve “PRIMARY KEY” ortak özellikleri

Dublication(Cift Kullanım)’a izin vermez

---

# Ornek Okul Tablosunun Bir Parçasi

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 sahsi 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	Pennsylvania
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) Kevin Star'in eyaleti nedir?
- 3) ID'si 17 olan kisinin sehri nedir?

# Related Tablolarla Calisma

## 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

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

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

# Related Tablolarla Calisma

## 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

StudentID	InstructorID
12	1
11	2
12	2
13	1
15	1
17	3
15	4

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) Ogretmeni Mark Adam olan ogrencilerin isimleri nedir?
- 2) Kevin Star'in ogretmenlerinin isimleri nedir?

# SQL Data Types

## String Data Types

<i>Data Type</i>	<i>Aciklama</i>
<b>char(size)</b>	Maximum boyutu <b>2000 byte</b> olur. 1 karakter 1 byte kullanir. “ <b>size</b> ” database’e eklenecek karakter sayisidir. “char” data tipinden <b>uzunlugu sabit</b> datalari depolar. (Strings) “char” SSN, zip kodu gibi uzunlugu sabit datalari depolamak icin idealdir.
<b>nchar(size)</b>	Maximum boyutu <b>2000 byte</b> olur. 1 karakter 2 byte kullanir “ <b>size</b> ” depolanacak <b>karakter sayisi</b> ’dir. “ <b>nchar</b> ” <b>Unicode</b> datalari depolamak icin kullanilir. Genellikle <b>farkli dillerdeki karakterler</b> icin kullanilir <b>Uzunlugu belli</b> Stringler icin kullanilir.
<b>varchar(size)</b>	Maximum boyutu <b>4000 byte</b> olur. 1 karakter 1 byte kullanir. “ <b>size</b> ” database’e eklenecek <b>max.</b> karakter sayisidir. <b>Degisken uzunluktaki</b> stringler icin kullanilir.
<b>nvarchar(size)</b>	Maximum boyutu <b>8000 byte</b> olur. 1 karakter 2 byte kullanir “ <b>size</b> ” depolanacak <b>karakter sayisi</b> ’dir. <b>Degisken uzunluktaki</b> stringlerin Unicode degerleri icin kullanilir.

# SQL Data Types

## Numeric Data Types

Data Type	Aciklama
<b>numeric(p, s)</b>	<p>“<b>Precision</b>” (p) sayidaki rakam sayisidir “<b>Scale</b>” (s) virgulden sonar kac rakam oldugunu belirler Ornegin: <b>1234,56</b> ==&gt; <b>Precision : 6, Scale : 2.</b></p> <p>Precision ( <b>p</b> ) can range from <b>1 to 38</b> Scale ( <b>s</b> ) can range from <b>-84 to 127</b></p> <p>1) “<b>numeric(5, 2)</b>” virgulden once 3,virgulden sonra 2 rakam olan sayi ==&gt; <b>123,45</b></p> <p>2) “<b>numeric(4, 2)</b>” ==&gt; <b>123,45</b> ==&gt; <b>error verir</b></p> <p>3) “<b>numeric(7)</b>” ondalik kısmi olmayan 7 basamakli sayi demektir ==&gt; <b>12345,67’i kabul eder ama 12345 olarak depolar</b></p> <p><b>Note:</b> “<b>numeric(7)</b>” ve “<b>numeric(7, 0)</b>” ayni seydir</p>



# SQL Data Types

## ***Numeric DataTypes***

**.TINYINT(*boyut*)** : Alabileceği değerler –128 ile 127 arasındadır.

“*Boyut*” ile alabileceği sınırı belirtebiliriz. kapladığı alan ise 1 byte

**.SMALLINT(*boyut*)** : -32.768 ile 32.767 arasında değer alır.

“*Boyut*” ile alabileceği sınırı belirtebiliriz. Hafızada kapladığı alan: 2 byte

**.MEDIUMINT(*boyut*)** : -8.388.608 ile 8.388.607 arasında değer alır.

“*Boyut*” ile alabileceği sınırı belirtebiliriz. Hafızada kapladığı alan: 3 byte.

**.INT** : Alabileceği değerler –2147483648 ile 2147483647 arasındadır.

“*Boyut*” ile alabileceği 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 alabileceği sınırı belirtebiliriz. Hafızada kapladığı alan: 8 byte.

---

# SQL Data Types

## Numeric Data Types

### DBMS Numeric Types:

<i>DBMS and version</i>	<i>Types</i>
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

<i>Data Type</i>	<i>Aciklama</i>
<b>DATE</b>	<p>“<b>DATE</b>” data tipi tarih ve zamani depolamak icin kullanilir. Saniyenin virgullu kismini da alir.</p> <p>“<b>DATE</b>” <b>yil</b>, ay, gun, saat, dakika, ve saniye icerir.</p> <p>Standart “<b>Date Format</b>” , “<b>YYYY - MM - dd</b>”. <b>Ornegin</b> 1982 - 07- 01</p>

---

# SQL Data Types

## BLOB Data Types

<i>Data Type</i>	<i>Aciklama</i>
<b>BYTEA</b>	“ <b>BYTEA</b> ” resim,video,ses gibi datalari binary formatina cevirerek depolar.

# SQL Komutlari

4. Veri Kontrol Dili (Data Control Language - DCL)  
veritabanı ve tablolar için yetki verilir  
veya geri alınır

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

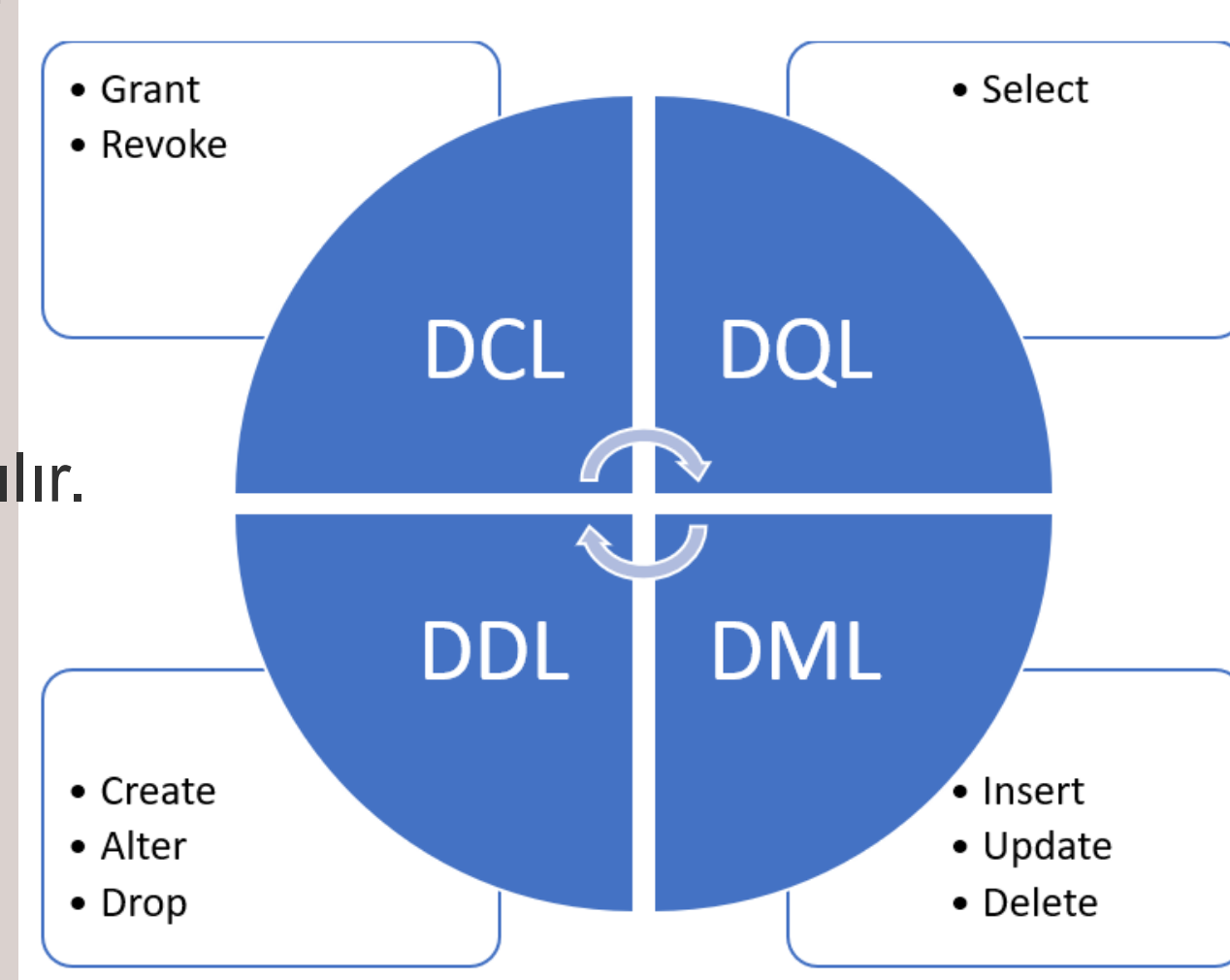
**REVOKE** : Bir kullanıcıya verilen  
yetkiyi geri almak için kullanılır.

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.



---

# DDL-Data Definition Language

## Create-Alter-Drop

### 1) Create – Tablo Oluşturma

```
CREATE TABLE ogrenciler  
(  
id char(4),  
Isim_soyisim varchar(50),  
not_ort real  
kayit_tarihi date  
);
```

### 2) Var olan tablodan yeni tablo olusturmak

```
CREATE TABLE ogrenci_ortalamalar  
AS SELECT isim_soyisim, not_ort  
FROM ogrenciler;
```

---

# DML – Data Manipulation Language

## Insert-Update-Delete

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

Olusturdugumuz ogrenciler tablosuna kayıt ekleyelim

- 1) Tum Field'lere data eklemek için

```
INSERT INTO ogrenciler VALUES (1234, 'Ali Can', 85.60, '14-Oct-2020');
```

- 2) Bazi Field'lere data eklemek için

```
INSERT INTO ogrenciler(ogrenci_id,isim_soyisim) VALUES (123456789, 'Ali Can');
```

---

# Table Nasıl Olusturulur?

## 1) Create Table

### Practice1:

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

## 2) Var olan tablodan yeni tablo olusturmak

“*tedarikçi\_ziyaret*” isminde bir tabloyu

“tedarikciler” tablosundan olusturun.

Icinde “tedarikci\_ismi”, “ulasim\_tarihi” field’lari olsun.

---

---

# Tabloya Nasıl Data Eklenir (INSERT INTO )?

## Practice2:

“*ogretmenler*” isminde tablo oluşturun. İçinde “*kimlik\_no*”, “*isim*”, “*brans*” ve “*cinsiyet*” field’leri olsun.

“*ogretmenler*” tablosuna bilgileri aşağıdaki gibi olan bir kişi ekleyin.

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

## Ekleme :

“*ogretmenler*” tablosuna bilgileri aşağıdaki gibi olan bir kişi ekleyin.

*Kimlik\_no: 567597624, isim: Ali Veli*

---

---

# DQL – Data Query Language **SELECT** KOMUTU

Tablodaki **T**um **F**ield'leri **C**agirma

**SELECT** \*  
**FROM** ogretmenler;      Tablodaki tum datalari getirir

**SELECT** isim  
**FROM** ogretmenler;      İsim field'ındaki dataları getirir



---

# CONSTRAINT

**UNIQUE** - Bir sütundaki tüm değerlerin BENZERSİZ/TEKRARSIZ yani tek olmasını sağlar.

**NOT NULL** - Bir Sütunun NULL içermemesini yani boş olmamasını sağlar.

NOT NULL kısıtlaması için constraint ismi tanımlanmaz. Bu kısıtlama veri türünden hemen sonra yerleştirilir

**PRIMARY KEY** - Bir sütünün NULL içermemesini ve sütundaki verilerin BENZERSİZ olmasını sağlar. (NOT NULL ve UNIQUE)

**FOREIGN KEY** - Başka bir tablodaki Primary Key'i referans göstermek için kullanılır. Böylelikle, tablolar arasında ilişki kurulmuş olur.

**Check** - Bir sütuna yerleştirilebilecek değer aralığını sınırlamak için kullanılır .

---

---

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

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

```
CREATE TABLE ogrenciler2
(  
id char(4) UNIQUE,  
isim varchar(50),  
not_ortalamasi real,  
adres varchar(100),  
kayit_tarihi date  
);
```

---

---

## Bir field'in "NULL" deger almamasi nasil saglanir?

"isim" field'inin "NULL" deger kabul etmemesi icin , isim field'i olusturulurken Data Type'dan sonra "NOT NULL" yazmak gerekir

```
CREATE TABLE ogrenciler3  
(  
  id char(4) UNIQUE,  
  isim varchar(50) NOT NULL,  
  not_ortalamasi real,  
  adres varchar(100),  
  kayit_tarihi date  
);
```

---

---

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

**Ornek :** Personel isminde bir tablo olusturun, icinde id,isim,soyisim,email,ise\_baslama\_tarihi ve maas fieldlari olsun, isim field'ı bos birakilmasin

```
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');  
- Bu eklemeyi kabul etmez
```

---

## 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'inde hic bir data NULL olamaz

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

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

```
CREATE TABLE ogrenciler4  
(  
id char(4) PRIMARY KEY,  
isim varchar(50) NOT NULL,  
not_ortalamasi real,  
adres varchar(100),  
kayit_tarihi date  
);
```

---



---

## Bir Tabloya “Primary Key” Nasıl Eklenir

2) **CONSTRAINT** Keyword Kullanılarak Primaray Key Tanımlanabilir

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

```
CREATE TABLE ogrenciler5  
(  
  id char(4),  
  isim varchar(50) NOT NULL,  
  not_ortalamasi real,  
  adres varchar(100),  
  kayit_tarihi date,  
  CONSTRAINT ogrenciler_pk PRIMARY KEY(id)  
);
```

---

---

## Bir Tabloya “Primary Key” Nasıl Eklenir

### Practice 3:

“sehirler” isimli bir Table oluşturun.  
Tabloda “alan\_kodu”, “isim”, “nufus”  
field’leri olsun. Isim field’i bos  
birakilmasin.  
1.Yontemi kullanarak “alan\_kodu”  
field’ini “Primary Key” yapin

---

# Tabloya “Foreign Key” Nasıl Eklenir ?

- Foreign Key iki tablo arasında Relation oluşturmak için kullanılır.
- Foreign Key başka bir tablonun Primary Key’ine bağlıdır.
- Referenced table (baglanılan tablo, Primary Key’in olduğu Tablo) *parent table* olarak adlandırılır. Foreign Key’in olduğu tablo ise *child table* olarak adlandırılır.
- Bir Tabloda birden fazla Foreign Key olabilir
- Foreign Key NULL değeri alabilir

**Note 1:** “Parent Table” olmayan bir id’ye sahip datayı “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 ?

SİRKETLER Tablosu

Primary Key

SİRKET_ID	SİRKET	PERSONEL_SAYISI
100	Honda	12000
101	Ford	18000
102	Hyundai	10000
103	Toyota	21000

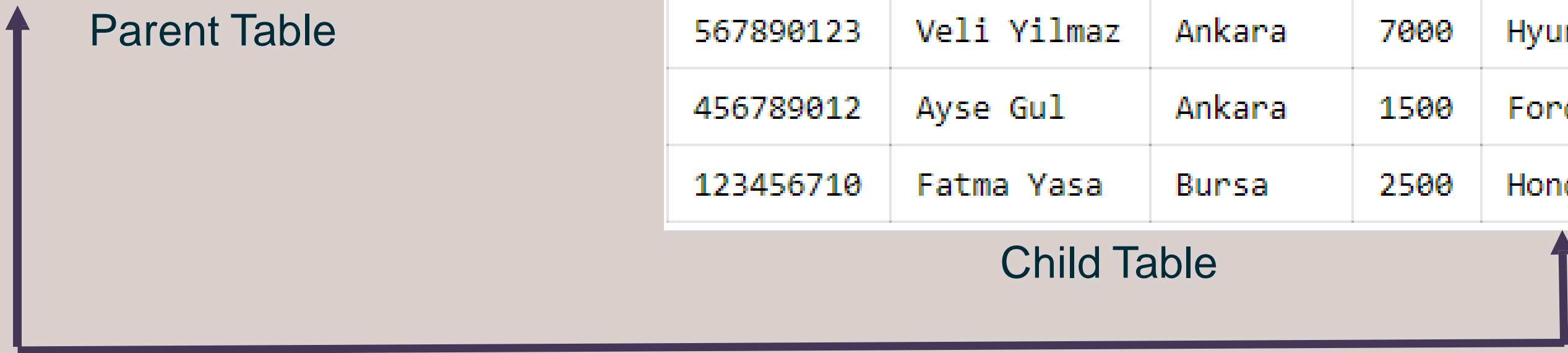
Parent Table

PERSONEL Tablosu

Foreign Key

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

Child Table



# Tabloya “Foreign Key” Nasıl Eklenir ?

## Practice 4:

“tedarikciler3” isimli bir tablo oluşturun. Tabloda “tedarikci\_id”, “tedarikci\_ismi”, “iletisim\_isim” field’lari olsun ve “tedarikci\_id” yi **Primary Key** yapin.

“urunler” isminde baska bir tablo oluşturun “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 tedarikciler3 (tedarikci_id)
```

```
);
```

```
CREATE TABLE tedarikciler3
```

```
(
```

```
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 5:

“*calisanlar*” isimli bir Tablo olusturun. Icinde “id”, “isim”, “maas”, “ise\_baslama” field’lari olsun. “id” yi **Primary Key** yapin, “isim” i **Unique**, “maas” i **Not Null** yapin.

“*adresler*” isminde baska bir tablo olusturun.Icinde “adres\_id”, “sokak”, “cadde” ve “sehir” fieldlari olsun. “adres\_id” field’i ile Foreign Key oluşturun.

```
INSERT INTO calisanlar VALUES('10002', 'Mehmet Yilmaz', 12000, '2018-04-14');
INSERT INTO calisanlar VALUES('10008', null, 5000, '2018-04-14');
INSERT INTO calisanlar VALUES('10010', Mehmet Yilmaz, 5000, '2018-04-14');
INSERT INTO calisanlar VALUES('10004', 'Veli Han', 5000, '2018-04-14');
INSERT INTO calisanlar VALUES('10005', 'Mustafa Ali', 5000, '2018-04-14');
INSERT INTO calisanlar VALUES('10006', 'Canan Yaş', NULL, '2019-04-12');
INSERT INTO calisanlar VALUES('10003', 'CAN', 5000, '2018-04-14');
INSERT INTO calisanlar VALUES('10007', 'CAN', 5000, '2018-04-14');
INSERT INTO calisanlar VALUES('10009', 'cem', '', '2018-04-14');
INSERT INTO calisanlar VALUES('', 'osman', 2000, '2018-04-14');
INSERT INTO calisanlar VALUES('', 'osman can', 2000, '2018-04-14');
INSERT INTO calisanlar VALUES('10002', 'ayse Yilmaz', 12000, '2018-04-14');
INSERT INTO calisanlar VALUES( null, 'filiz ', 12000, '2018-04-14');
```

```
INSERT INTO adresler VALUES('10003','Mutlu Sok', '40.Cad.','IST');
INSERT INTO adresler VALUES('10003','Can Sok', '50.Cad.','Ankara');
INSERT INTO adresler VALUES('10002','Ağa Sok', '30.Cad.','Antep');
```

-- Parent tabloda olmayan id ile child a ekleme yapamayiz

```
INSERT INTO adresler VALUES('10012','Ağa Sok', '30.Cad.','Antep');
```

-- FK'ye null değeri atanabilir.

```
INSERT INTO adresler VALUES(NULL,'Ağa Sok', '30.Cad.','Antep');
INSERT INTO adresler VALUES(NULL,'Ağa Sok', '30.Cad.','Maraş');
```

---

## Tabloya “CHECK Constraint” Nasıl Eklenir ?

**CHECK** ile bir alana girilebilecek değerleri sınırlandırabiliriz.

Mesela tablomuzda YAŞ bir alanı **number** data tipinde yani sayısal alan olarak belirlemiş olabiliriz. Ancak bu alan negatif sayı girilmesi anlamsız olacağı için CHECK yapısını kullanarak negatif giriş yapılmasını engelleyebiliriz.

**Ornek** : şehirler2 tablosu oluşturalım, nüfusun negatif değer girilmemesi için sınırlandırma (Constraint) koyalım

```
CREATE TABLE şehirler2
(
  alan_kodu int PRIMARY KEY,
  isim varchar(20) NOT NULL,
  nüfus int CHECK (nüfus>0)
);
```

---

---

# WHERE ile Kullanilan Mantiksal 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

---

---

# DQL - SELECT KOMUTU

Tablodaki **B**elli **B**ir **F**ield'i **C**agirma

**SELECT** isim

**FROM** calisanlar;    Tablodan sadece isim field'indaki tum datalari getirir

**SELECT** isim

**FROM** calisanlar

**WHERE** maas>5000;    Tablodan sadece maas'ı 5000 den buyuk olanlarin isim field'indaki datalari getirir

---

# DQL - SELECT KOMUTU

Tablodan **B**irden **F**azla **F**ield'i **C**agirma

**SELECT** adres,isim

**FROM** ogrenciler;

Tablodan adres ve isim field'indeki tum datalari getirir

ADRES	ISIM
Ankara	Ali Can
Ankara	Merve Gul
Istanbul	Kemal Yasa

**SELECT** adres,isim

**FROM** ogrenciler

**WHERE** yazili\_not>80;

Tablodan yazili notu 80'den büyük olan kayitlarin adres ve isim field'indeki datalari getirir

ADRES	ISIM
Ankara	Merve Gul
Istanbul	Kemal Yasa

---

# DQL - SELECT KOMUTU

Tablodan Bir Kayıda Ait Birden Fazla Field'i Çağırma

SELECT adres,isim

FROM ogrenciler

WHERE id=123 ;      Tablodan id'si 123 olan kaydın adres ve isim field'indeki dataları getirir



---

## DML - DELETE KOMUTU

\* **DELETE** FROM tablo\_adi; Tablonun tüm içeriğini siler.

- Veriyi seçerek silmek için WHERE komutu kullanılır

\* **DELETE** FROM tablo\_adi WHERE sutun\_adi = veri;

Tabloda istediğiniz veriyi siler.

```
CREATE TABLE ogrenciler6
(
id int,
isim VARCHAR(50),
veli_isim VARCHAR(50),
yazili_notu int
);
```

-- id'si 124 olan öğrenciyi siliniz.

-- ismi Kemal Yasa olan satırını siliniz.

```
INSERT INTO ogrenciler6 VALUES(123, 'Ali Can', 'Hasan',75);
INSERT INTO ogrenciler6 VALUES(124, 'Merve Gul', 'Ayse',85);
INSERT INTO ogrenciler6 VALUES(125, 'Kemal Yasa', 'Hasan',85);
INSERT INTO ogrenciler6 VALUES(126, 'Nesibe Yilmaz', 'Ayse',95);
INSERT INTO ogrenciler6 VALUES(127, 'Mustafa Bak', 'Can',99);
INSERT INTO ogrenciler6 VALUES(127, 'Mustafa Bak', 'Ali', 99);
```

---

---

## DML - DELETE KOMUTU

-- ismi Nesibe Yilmaz veya Mustafa Bak olan kayıtları silelim.

**DELETE** FROM ogrenciler

WHERE isim = 'Nesibe Yilmaz' or isim='Mustafa Bak';

### Practice 7:

-- İsmi Ali Can ve id'si 123 olan kaydı siliniz.

-- id 'si 126'dan büyük olan kayıtları silelim.

-- id'si 123, 125 veya 126 olanları silelim.

“**DELETE FROM** ogrenciler” -tablodaki tum datalari siler, fakat tabloyu silmez.

“**DELETE FROM** ogrenciler”, kodunu kullanınca bos bir tablo kalir.

---

---

# DML - DELETE - TRUNCATE

TRUNCATE komutu DELETE komutu gibi bir tablodaki verilerin tamamını siler.

Ancak, seçmeli silme yapamaz.

TRUNCATE TABLE where ..... OLMAZ

**TRUNCATE** TABLE öğrenciler;

-- tablodaki verileri siler

---

---

# ON DELETE CASCADE

- Her defasında önce child tablodaki verileri silmek yerine ON DELETE CASCADE silme özelliğini aktif hale getirebiliriz.

Bunun için FK olan satırın en sonuna ON DELETE CASCADE komutunu yazmak yeterli

```
CREATE TABLE talebeler  
(  
  id CHAR(3) primary key,  
  isim VARCHAR(50),  
  veli_isim VARCHAR(50),  
  yazili_notu int  
);
```

```
CREATE TABLE notlar(  
  talebe_id char(3),  
  ders_adi varchar(30),  
  yazili_notu int,  
  CONSTRAINT notlar_fk FOREIGN KEY (talebe_id)  
  REFERENCES talebeler(id)  
  on delete cascade  
);
```

-- **on delete cascade** sayesinde parent taki silinen bir kayıt ile ilişkili olan tüm child kayıtlarını silebiliriz  
-- cascade yoksa önce child temizlenir sonra parent

---

---

## ON DELETE CASCADE

```
INSERT INTO talebeler VALUES(123, 'Ali  
Can', 'Hasan',75);  
INSERT INTO talebeler VALUES(124,  
'Merve Gul', 'Ayse',85);  
INSERT INTO talebeler VALUES(125,  
'Kemal Yasa', 'Hasan',85);  
INSERT INTO talebeler VALUES(126,  
'Nesibe Yilmaz', 'Ayse',95);  
INSERT INTO talebeler VALUES(127,  
'Mustafa Bak', 'Can',99);
```

```
INSERT INTO notlar VALUES ('123','kimya',75);  
INSERT INTO notlar VALUES ('124', 'fizik',65);  
INSERT INTO notlar VALUES ('125', 'tarih',90);  
INSERT INTO notlar VALUES ('126', 'Matematik',90);
```

```
DELETE from notlar where talebe_id='123';-- child  
-- child tablodaki veriyi sildiğimiz zaman sadece child'daki veri  
silinir. parent taki veri silinmez.
```

```
DELETE from talebeler where id='126';-- parent  
-- parent tablodaki veriyi sildiğimiz zaman child'daki veride  
silinir.
```

```
DELETE FROM talebeler; -- Parent tablo ile birlikte child tablo  
daki verileride siler
```

```
DROP TABLE talebeler CASCADE; -- ilişkili tablolardan parent  
olan talebeler tablosunu siler
```

---

# IN CONDITION

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

**AND** (ve): Belirtilen şartların her ikisinde gerçekleşiyorsa o kayıt listelenir.

**OR** (veya): Belirtilen şartlardan biri gerçekleşirse, kayıt listelenir.

```
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 müşteriler 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 mantiksal ifade ile tanımlayabileceğimiz durumları tek komutla yazabilme imkanı verir. Yazdığımız 2 sınırdaki aralıga 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

# NOT BETWEEN CONDITION

**NOT BETWEEN Condition** iki mantıksal ifade ile tanımlayabileceğimiz durumları tek komutla yazabilme imkanı verir. Yazdığımız 2 sınırdaki aralığa 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 > 40;
```

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

urun_id integer	musteri_isim character varying (50)	urun_isim character varying (50)
10	Mark	Orange
10	Mark	Orange
10	Adem	Orange

---

## Practice 6

- id'si 1003 ile 1005 arasında olan personel bilgilerini listeleyiniz
- D ile Y arasındaki personel bilgilerini listeleyiniz
- D ile Y arasında olmayan personel bilgilerini listeleyiniz
- Maaşı 70000 ve ismi Sena olan personeli listeleyiniz

```
CREATE table personel  
(  
id char(4),  
isim varchar(50),  
maas int  
);
```

```
insert into personel values('1001', 'Ali Can', 70000);  
insert into personel values('1002', 'Veli Mert', 85000);  
insert into personel values('1003', 'Ayşe Tan', 65000);  
insert into personel values('1004', 'Derya Soylu', 95000);  
insert into personel values('1005', 'Yavuz Bal', 80000);  
insert into personel values('1006', 'Sena Beyaz', 100000);
```

---

# SUBQUERIES

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

1) **WHERE** den sonra kullanilabilir

```
CREATE TABLE calisanlar2
(
id int,
isim VARCHAR(50),
sehir VARCHAR(50),
maas int,
isyeri VARCHAR(20)
);
```

```
INSERT INTO calisanlar2 VALUES(123456789, 'Ali Seker', 'Istanbul', 2500, 'Vakko');
INSERT INTO calisanlar2 VALUES(234567890, 'Ayse Gul', 'Istanbul', 1500, 'LCWaikiki');
INSERT INTO calisanlar2 VALUES(345678901, 'Veli Yilmaz', 'Ankara', 3000, 'Vakko');
INSERT INTO calisanlar2 VALUES(456789012, 'Veli Yilmaz', 'Izmir', 1000, 'Pierre Cardin');
INSERT INTO calisanlar2 VALUES(567890123, 'Veli Yilmaz', 'Ankara', 7000, 'Adidas');
INSERT INTO calisanlar2 VALUES(456789012, 'Ayse Gul', 'Ankara', 1500, 'Pierre Cardin');
INSERT INTO calisanlar2 VALUES(123456710, 'Fatma Yasa', 'Bursa', 2500, 'Vakko');
```

```
CREATE TABLE markalar
(
marka_id int,
marka_isim VARCHAR(20),
calisan_sayisi int
);
```

```
INSERT INTO markalar VALUES(100, 'Vakko', 12000);
INSERT INTO markalar VALUES(101, 'Pierre Cardin', 18000);
INSERT INTO markalar VALUES(102, 'Adidas', 10000);
INSERT INTO markalar VALUES(103, 'LCWaikiki', 21000);
```

---

---

# SUBQUERIES

-- Çalışan sayısı 15.000'den çok olan markaların isimlerini ve bu markada çalışanların isimlerini ve maaşlarını listeleyin.

```
SELECT isim, maas, isyeri from calisanlar  
where isyeri in (select marka_isim from markalar where calisan_sayisi>15000);
```

-- marka\_id'si 101'den büyük olan marka çalışanlarının isim, maaş ve şehirlerini listeleyiniz.

```
SELECT isim, maas, sehir from calisanlar  
where isyeri in (select marka_isim from markalar where marka_id > 101 );
```

-- Ankara'da çalışan olan markaların marka id'lerini ve çalışan sayılarını listeleyiniz.

---

---

# SUBQUERIES

## AGGREGATE METOT KULLANIMI

- Aggregate Metotlari (SUM,COUNT, MIN, MAX, AVG) Subquery içinde kullanılabilir.

Ancak, Sorgu tek bir değer döndürüyor olmalıdır.

SYNTAX: sum() şeklinde olmalı sum ile () arasında boşluk olmamalı

select **max**(maas) from calisanlar

select **sum**(maas) from calisanlar

select **avg**(maas) from calisanlar

select **round**(avg(maas),2) from calisanlar →  $19000/7 = 2714,29$

select **min**(maas) from calisanlar

select **count**(maas) from calisanlar

---



# SUBQUERIES

-- Her markanın id'sini, ismini ve toplam kaç şehirde bulunduğunu listeleyen bir SORGU yazınız.

```
SELECT marka_id, marka_isim,  
(SELECT count(sehir) FROM calisanlar where marka_isim=isyeri) as sehir_sayisi  
FROM markalar;
```

Çalışanlar Tablosu

id integer	isim character varying (50)	sehir character varying (50)	maas integer	isyeri character varying (20)
123456789	Ali Seker	Istanbul	2500	Vakko
234567890	Ayse Gul	Istanbul	1500	LCWaikiki
345678901	Veli Yilmaz	Ankara	3000	Vakko
456789012	Veli Yilmaz	Izmir	1000	Pierre Cardin
567890123	Veli Yilmaz	Ankara	7000	Adidas
456789012	Ayse Gul	Ankara	1500	Pierre Cardin
123456710	Fatma Yasa	Bursa	2500	Vakko

Markalar Tablosu

marka_id integer	marka_isim character varying (20)	calisan_sayisi integer
100	Vakko	12000
101	Pierre Cardin	18000
102	Adidas	10000
103	LCWaikiki	21000

marka_id integer	marka_isim character varying (20)	sehir_sayisi bigint
100	Vakko	3
101	Pierre Cardin	2
102	Adidas	1
103	LCWaikiki	1

---

# SUBQUERIES

-- Her markanın ismini, çalışan sayısını ve o markaya ait çalışanların toplam maaşını listeleyiniz

```
SELECT marka_isim, calisan_sayisi,  
(SELECT sum(maas) FROM calisanlar WHERE marka_isim = isyeri) as toplam_maas FROM markalar;
```

-- Her markanın ismini, çalışan sayısını ve o markaya ait çalışanların maksimum ve minimum maaşını listeleyen bir Sorgu yazınız.

```
SELECT marka_isim, calisan_sayisi, (SELECT max(maas) FROM calisanlar WHERE marka_isim = isyeri) as max_maas,  
                                   (SELECT min(maas) FROM calisanlar WHERE marka_isim = isyeri) as min_maas  
FROM markalar;
```

---

# 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 değer olması veya olmaması durumunda işlem yapılmasını sağlar.

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

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

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

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

---

# EXISTS CONDITION

--MART VE NİSAN aylarında aynı URUN\_ID ile satılan ürünlerin URUN\_ID'lerini listeleyen ve aynı zamanda bu ürünleri MART ayında alan MUSTERI\_ISIM 'lerini listeleyen bir sorgu yazınız.

**SELECT** urun\_id,musteri\_isim **FROM** mart  
**WHERE exists** (**select** urun\_id **FROM** nisan **WHERE** mart.urun\_id=nisan.urun\_id)

MART		
urun_id	musteri_isim	urun_isim
10	Mark	Honda
20	John	Toyota
30	Amy	Ford
20	Mark	Toyota
10	Adam	Honda
40	John	Hyundai
20	Eddie	Toyota

NİSAN					
urun_id	musteri_isim	urun_isim		urun_id	musteri_isim
10	Hasan	Honda		10	Mark
10	Kemal	Honda		20	John
20	Ayşe	Toyota		20	Mark
50	Yasar	Volvo		10	Adam
20	Mine	Toyota		20	Eddie

# EXISTS CONDITION

--Her iki ayda birden satılan ürünlerin URUN\_ISIM'lerini ve bu ürünleri NİSAN ayında satın alan MUSTERI\_ISIM'lerini listeleyen bir sorgu yazınız.

SELECT urun\_isim,musteri\_isim FROM nisan  
WHERE exists (SELECT urun\_isim FROM mart where mart.urun\_isim=nisan.urun\_isim)

MART			NİSAN		
urun_id	musteri_isim	urun_isim	urun_id	musteri_isim	urun_isim
10	Mark	Honda	10	Hasan	Honda
20	John	Toyota	10	Kemal	Honda
30	Amy	Ford	20	Ayşe	Toyota
20	Mark	Toyota	50	Yasar	Volvo
10	Adam	Honda	20	Mine	Toyota
40	John	Hyundai			
20	Eddie	Toyota			

--Her iki ayda ortak satılmayan ürünlerin URUN\_ISIM'lerini ve bu ürünleri NİSAN ayında satın alan MUSTERI\_ISIM'lerini listeleyen bir sorgu yazınız.

---

# Tablodaki Data Nasıl Update Edilir (UPDATE SET)?

CREATE TABLE tedarikciler -- parent

```
(  
vergi_no int PRIMARY KEY,  
firma_ismi VARCHAR(50),  
irtibat_ismi VARCHAR(50)  
);
```

```
INSERT INTO tedarikciler VALUES (101, 'IBM', 'Kim Yon');  
INSERT INTO tedarikciler VALUES (102, 'Huawei', 'Çin Li');  
INSERT INTO tedarikciler VALUES (103, 'Erikson', 'Maki Tammen');  
INSERT INTO tedarikciler VALUES (104, 'Apple', 'Adam Eve');
```

CREATE TABLE urunler -- child

```
(  
ted_vergino int,  
urun_id int,  
urun_isim VARCHAR(50),  
musteri_isim VARCHAR(50),  
CONSTRAINT fk_urunler FOREIGN KEY(ted_vergino)  
REFERENCES tedarikciler(vergi_no)  
on delete cascade  
);
```

```
INSERT INTO urunler VALUES(101, 1001,'Laptop', 'Ayşe Can');  
INSERT INTO urunler VALUES(102, 1002,'Phone', 'Fatma Aka');  
INSERT INTO urunler VALUES(102, 1003,'TV', 'Ramazan Öz');  
INSERT INTO urunler VALUES(102, 1004,'Laptop', 'Veli Han');  
INSERT INTO urunler VALUES(103, 1005,'Phone', 'Canan Ak');  
INSERT INTO urunler VALUES(104, 1006,'TV', 'Ali Bak');  
INSERT INTO urunler VALUES(104, 1007,'Phone', 'Aslan Yılmaz');
```

---



---

# Tablodaki Data Nasıl Update Edilir (UPDATE SET)?

-- vergi\_no'su 102 olan tedarikcinin firma ismini 'Vestel' olarak güncelleyiniz.

UPDATE tedarikciler

SET firma\_ismi = 'Vestel' WHERE vergi\_no=102;

-- vergi\_no'su 101 olan tedarikçinin firma ismini 'casper' ve irtibat\_ismi'ni 'Ali Veli' olarak güncelleyiniz.

UPDATE tedarikciler SET firma\_ismi = 'casper' WHERE vergi\_no=101;

UPDATE tedarikciler SET irtibat\_ismi = 'Ali Veli' WHERE vergi\_no=101;

UPDATE tedarikciler SET firma\_ismi = 'casper', irtibat\_ismi='Ali Veli' WHERE vergi\_no=101;

-- urunler tablosundaki 'Phone' değerlerini 'Telefon' olarak güncelleyiniz.

UPDATE urunler

SET urun\_isim = 'Telefon'

WHERE urun\_isim='Phone';

-- urunler tablosundaki urun\_id değeri 1004'ten büyük olanların urun\_id'sini 1 arttırın.

UPDATE urunler SET urun\_id = urun\_id+1 WHERE urun\_id > 1004;

-- urunler tablosundaki tüm ürünlerin urun\_id değerini ted\_vergino sutun değerleri ile toplayarak güncelleyiniz.

UPDATE urunler SET urun\_id = urun\_id + ted\_vergino;

---

---

# Tablodaki Data Nasıl Update Edilir (UPDATE SET)?

\* urunler tablosundan Ali Bak'ın aldığı ürünün ismini, tedarikci tablosunda irtibat\_ismi 'Adam Eve' olan firmanın ismi (firma\_ismi) ile değiştiriniz.

-- Bu update işlemini yapmadan önce, tabloları eski haline getirmeliyiz.

UPDATE urunler

SET urun\_isim = (select firma\_ismi from tedarikciler WHERE irtibat\_ismi = 'Adam Eve')

WHERE musteri\_isim='Ali Bak';

\* Urunler tablosunda laptop satın alan müşterilerin ismini, firma\_ismi Apple'ın irtibat\_isim'i ile değiştirin.

UPDATE urunler

SET musteri\_isim = (select irtibat\_ismi from tedarikciler WHERE firma\_ismi = 'Apple')

WHERE urun\_isim = 'Laptop'

---

# ALIASES

Aliases kodu ile tablo yazdirilirken, field isimleri sadece o cikti icin degistirilebilir

```
CREATE TABLE calisanlar
(
calisan_id char(9),
calisan_isim varchar(50),
calisan_dogdugu_sehir varchar(50) );
```

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

calisan_id character (9)	calisan_isim character varying (50)	calisan_dogdugu_sehir character varying (50)
123456789	Ali Can	Istanbul
234567890	Veli Cem	Ankara
345678901	Mine Bulut	Izmir

```
SELECT calisan_id AS id, calisan_isim AS isim, calisan_dogdugu_sehir AS dogum_yeri
FROM calisanlar;
```

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

```
SELECT calisan_id AS id, calisan_isim || calisan_dogdugu_sehir AS isim_ve_dogum_yeri
FROM calisanlar;
```

ID	ISIM_VE_DOGUM_YERI
123456789	Ali CanIstanbul
234567890	Veli CemAnkara
345678901	Mine BulutIzmir

## Practice 8

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 unvani isci olanlarin isimlerini listeleyin
- c) Iscilerin tum bilgilerini listeleyin
- d) Soyadi Can,Cem veya Gul olanlarin unvanlarini ve maaslarini listeleyin
- e) Maasi 5000’den cok olanlarin emailve is baslama tarihlerini listeleyin
- f) Maasi 5000’den cok veya 7000’den az olanlarin tum bilgilerini listeleyin

# IS NULL CONDITION

Arama yapılan field’da NULL degeri almıs kayıtları getirir.

```
CREATE TABLE insanlar
(  
  ssn char(9),  
  isim 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’, ‘Izmir’);  
INSERT INTO insanlar (ssn, adres) VALUES(456789012, ‘Bursa’);  
INSERT INTO insanlar (ssn, adres) VALUES(567890123, ‘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 NULL;
```

SSN	NAME	ADDRESS
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

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

SSN	NAME	ADDRESS
456789012	Isim Girilmemis	Bursa
567890123	Isim Girilmemis	Denizli

# 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

Insanlar tablosundaki ismi Mine olanlari SSN sirali 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 numarasi da kullanilabilir

Insanlar tablosundaki soyismi Bulut olanlari isim sirali 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

---

# ORDER BY field\_name DESC CLAUSE

İsim ve soyisim değerlerini soyisim kelime uzunluklarına göre sıralayınız

```
SELECT isim, soyisim  
FROM insanlar  
ORDER BY LENGTH (soyisim);
```

Tüm isim ve soyisim değerlerini aynı sütunda çağırarak her bir sütun değerini uzunluğuna göre sıralayınız

```
SELECT CONCAT (isim, ' ', soyisim) AS isim_soyisim  
FROM insanlar  
ORDER BY LENGTH(isim)+LENGTH(soyisim);
```

---

# GROUP BY CLAUSE

**Group By** komutu sonuçları bir veya daha fazla sütuna göre gruplamak 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 alinan toplam urunleri bulun

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

ISIM	ALINAN_TOPLAM_MEYVE
Veli	10
Ayse	8
Ali	9
Hasan	4

# GROUP BY CLAUSE

ISIM	URUN_ADI	URUN_MIKTAR
Ali	Elma	5
Ayşe	Armut	3
Veli	Elma	2
Hasan	Uzum	4
Ali	Armut	2
Ayşe	Elma	3
Veli	Uzum	5
Ali	Armut	2
Veli	Elma	3
Ayşe	Uzum	2

2) Urun ismine gore urunu alan toplam kisi sayisi

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

URUN_ADI	URUNU_ALAN_KISI_SAYISI
Elma	4
Uzum	3
Armut	3

3) Alinan kilo miktarina gore musteri sayisi

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

URUN_MIKTAR	URUN_MIKTARINI_ALAN_KISI_SAYISI
2	4
5	2
4	1
3	3

# 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;

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

2) sehre gore toplam personel sayisini bulun

SELECT sehir, COUNT(isim) AS calisan\_sayisi  
FROM personel  
GROUP BY sehir;

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_MAAS	EN_FAZLA_MAAS
Honda	3500	5500
Ford	4500	6000
Toyota	4500	4500
Tofas	7000	7000

# HAVING CLAUSE

**HAVING**, AGGREGATE FUNCTION'lar ile birlikte kullanılan FİLTRELEME 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) Aynı isimdeki kişilerin aldığı toplam gelir 10000 liradan fazla ise ismi ve toplam maaşı gösteren sorgu yazınız

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

ISIM	TOPLAM_MAAS
Mehmet Ozturk	16500

3) Eğer bir şehirde çalışan personel sayısı 1'den fazla şehir ismini ve personel sayısını 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 alinan MAX maas 5000'den dusukse sehir ismini ve MAX maasi veren sorgu yaziniz

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

SEHIR	MAX_MAAS
Bursa	4500

# UNION OPERATOR

İki farklı sorgulamanın sonucunu birleştiren işlemidir. Seçilen **Field SAYISI** ve **DATA TYPE**'i aynı olmalıdır.

```
CREATE TABLE personel
```

```
(  
  id int,  
  isim varchar(50),  
  sehir varchar(50),  
  maas int,  
  sirket varchar(20)  
  INSERT INTO personel VALUES(123456789, 'Ali Yilmaz', 'İstanbul', 5500, 'Honda');  
  INSERT INTO personel VALUES(234567890, 'Veli Şahin', 'İstanbul', 4500, 'Toyota');  
  INSERT INTO personel VALUES(345678901, 'Mehmet Öztürk', 'Ankara', 3500, 'Honda');  
  INSERT INTO personel VALUES(456789012, 'Mehmet Öztürk', 'İzmir', 6000, 'Ford');  
  INSERT INTO personel VALUES(567890123, 'Mehmet Öztürk', 'Ankara', 7000, 'Tofas');  
  INSERT INTO personel VALUES(456789012, 'Veli Şahin ', 'Ankara', 4500, 'Ford');  
  INSERT INTO personel VALUES(123456710, 'Hatice Şahin', 'Bursa', 4500, 'Honda');  
);
```

1) Maası 4000'den çok olan işçi isimlerini ve 5000 liradan fazla maaş alan şehirleri gösteren sorguyu yazınız

```
SELECT sehir AS işçi_veya_sehir_ismi ,maas  
FROM personel  
WHERE maas >5000  
UNION  
SELECT isim AS işçi_veya_sehir_ismi , maas  
FROM personel  
WHERE maas > 4000;
```

İSÇİ_VEYA_SEHİR_İSMİ	MAAS
Ali Yılmaz	5500
Ankara	4500
Ankara	7000
Bursa	4500
Hatice Şahin	4500
İstanbul	4500
İstanbul	5500
İzmir	6000
Mehmet Öztürk	6000
Mehmet Öztürk	7000
Veli Şahin	4500
Veli Şahin	4500

# UNION OPERATOR

2) Mehmet Ozturk ismindeki kisilerin aldigi maaslari ve Istanbul'daki personelin maaslarini bir tabloda gosteren sorgu yaziniz

```
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 kullanirsaniz tum tabloyu istediginiz siralamaya gore siralar

```
ORDER BY maas;
```

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

# UNION OPERATOR

3) Sehirlerden odenen ucret 3000'den fazla olanlari ve personelden ucreti 5000'den az olanlari bir tabloda maas miktarina gore sirali olarak gosteren sorguyu yaziniz

```
SELECT sehir AS isci_veya_sehir_ismi , maas
FROM personel
WHERE maas>3000
UNION
SELECT isim AS isci_veya_sehir_ismi , maas
FROM personel
WHERE maas<5000;
```

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

# UNION OPERATOR

## 2 Tablodan Data Birleştirme

Personel isminde bir tablo oluşturun. İçinde id, isim, şehir, maaş ve şirket alanları olsun.

Id'yi 2. yöntemle PK yapın

```
CREATE TABLE personel
(
  id int,
  isim varchar(50),
  şehir varchar(50),
  maaş int,
  şirket varchar(20),
  CONSTRAINT personel_pk PRIMARY KEY (id)
);
```

```
INSERT INTO personel VALUES(123456789, 'Ali Yılmaz', 'İstanbul', 5500, 'Honda');
INSERT INTO personel VALUES(234567890, 'Veli Şahin', 'İstanbul', 4500, 'Toyota');
INSERT INTO personel VALUES(345678901, 'Mehmet Öztürk', 'Ankara', 3500, 'Honda');
INSERT INTO personel VALUES(456789012, 'Mehmet Öztürk', 'İzmir', 6000, 'Ford');
INSERT INTO personel VALUES(567890123, 'Mehmet Öztürk', 'Ankara', 7000, 'Tofas');
INSERT INTO personel VALUES(456715012, 'Veli Şahin', 'Ankara', 4500, 'Ford');
INSERT INTO personel VALUES(123456710, 'Hatice Şahin', 'Bursa', 4500, 'Honda');
```

Personel\_bilgi isminde bir tablo oluşturun. İçinde id, tel ve çocuk sayısı alanları olsun. Id'yi FK yapın ve personel tablosu ile ilişki kurun

```
CREATE TABLE personel_bilgi (
  id int,
  tel char(10) UNIQUE ,
  çocuk_sayısı 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_maas
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

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

**NOT : Union islemi yaparken**

1)Her 2 QUERY'den elde edeceginiz tablolarin sutun sayilari esit olmalı

2)Alt alta gelecek sutunlarin data type'lari ayni olmalı



# 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 sorguyu 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 sorguyu UNION ALL ile iki kere yazarak calistirin

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

UNION ALL

```
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** birleştirmek için kullanılır, Aynı kayıt birden fazla olursa, sadece bir tanesini alır.

**UNION ALL** ise tekrarlı elemanları, tekrar sayısınca yazar.

**NOT : UNION ALL ile birleştirmelerde de**

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

# UNION ALL OPERATOR

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

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

SELECT sehir,maas  
FROM personel  
WHERE maas>4000;

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

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

SELECT isim,maas  
FROM personel  
WHERE maas<5000;

3) Iki sorguyu UNION ve UNION ALL ile birlestirin

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

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

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

# INTERSECT OPERATOR

1) Personel tablosundan Istanbul veya Ankara'da calisanlarin id'lerini yazdir

ID
123456789
234567890
345678901
567890123
456715012

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

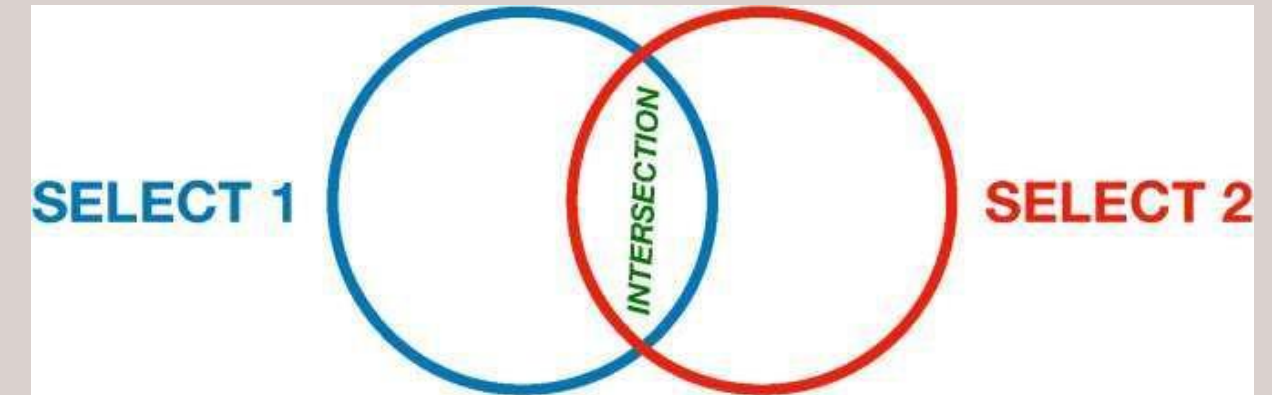
2) Personel\_bilgi tablosundan 2 veya 3 cocugu olanlarin id lerini yazdirin

ID
345678901
456789012
567890123
456789012

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

3) Iki sorguyu INTERSECT ile birlestirin

ID
345678901
567890123



# INTERSECT OPERATOR

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

ID
234567890
345678901
456789012
567890123
456715012
123456710

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

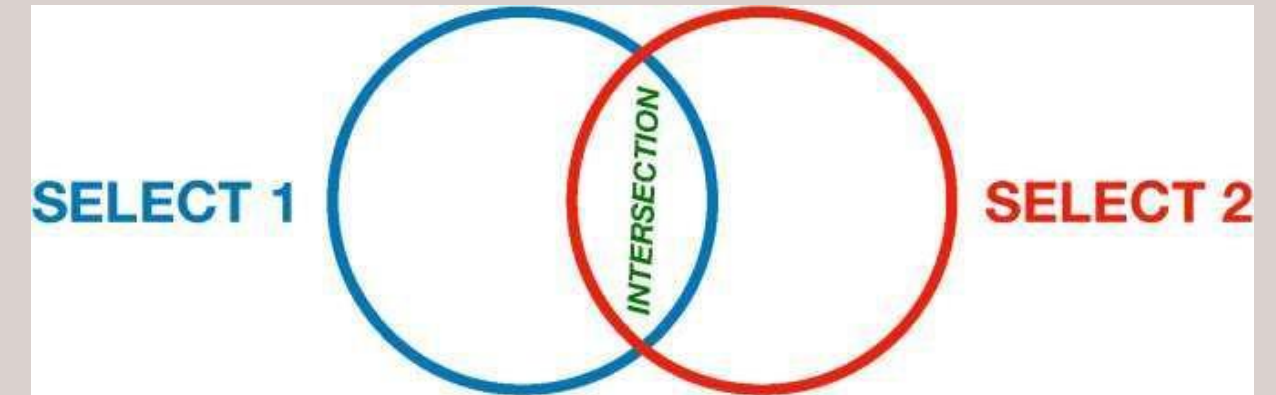
2) Personel\_bilgi tablosundan 2 veya 3 cocugu olanlarin id lerini yazdirin

ID
345678901
456789012
567890123
456789012

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

3) Iki sorguyu INTERSECT ile birlestirin

ID
345678901
456789012
567890123



# INTERSECT OPERATOR

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

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

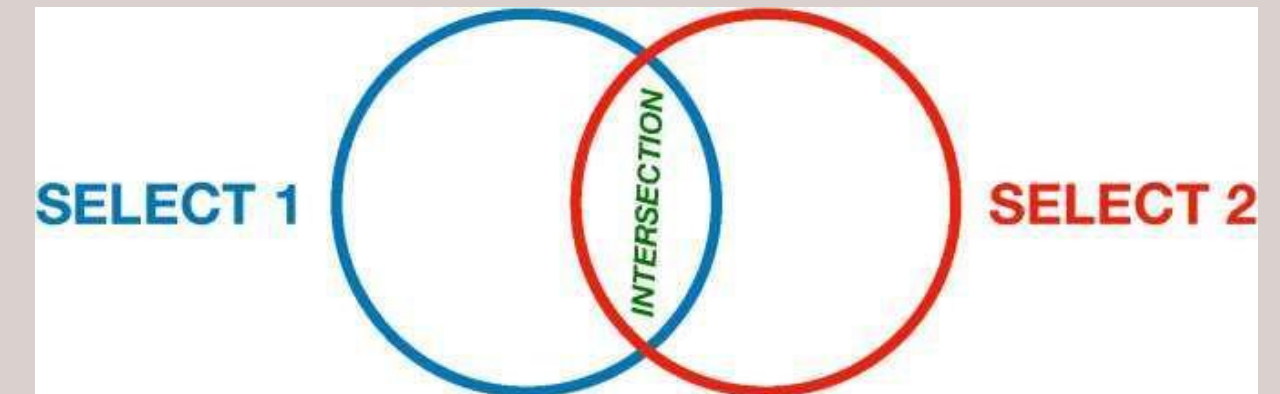
ISIM
Mehmet Ozturk

INTERSECT

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

INTERSECT

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





# EXCEPT OPERATOR

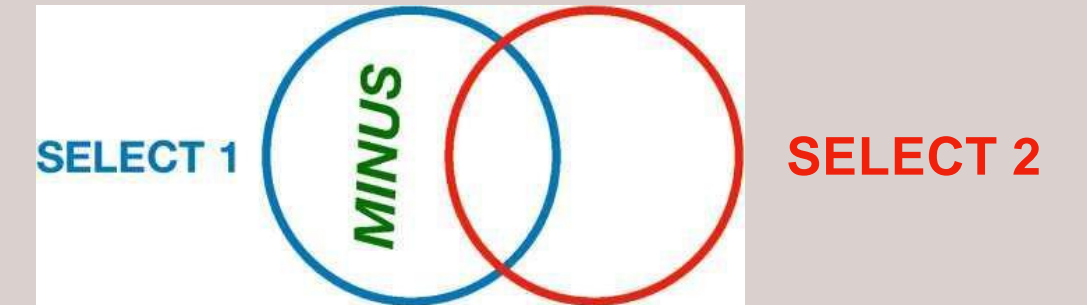
1) 5000'den az maas alip Honda'da calismayanlari yazdirin

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

EXCEPT

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

ISIM	SIRKET
Veli Sahin	Ford
Veli Sahin	Toyota



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

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

EXCEPT

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

ISIM	SEHIR
Mehmet Ozturk	Ankara
Mehmet Ozturk	Izmir

---

# JOINS

2 **Tablo**daki datalari Birleştirmek için kullanılır.

Su ana kadar gördüğümüz Union, Intersect ve Minus sorgu sonuçları için kullanılır  
Tablolar için ise **JOIN** kullanılır

5 Cesi Join vardır

- 1) INNER JOIN iki Tablodaki ortak datalari gösterir
  - 2) LEFT JOIN İlk datada olan tüm recordları gösterir
  - 3) RIGHT JOIN İkinci tabloda olan tüm recordları gösterir
  - 4) FULL JOIN İki tablodaki tüm recordları gösterir
  - 5) SELF JOIN Bir tablonun kendi içinde Join edilmesi ile oluşur.
-

# 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

TABLE 1

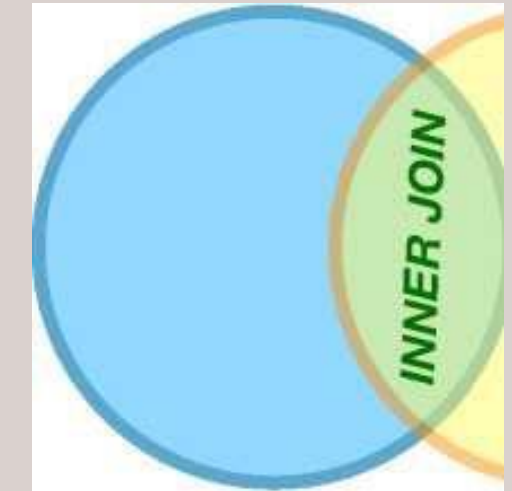


TABLE 2

**SORU)** İki Tabloda `sirket_id`'si aynı olanların `sirket_ismi`, `siparis_id` ve `siparis_tarihleri` ile yeni bir tablo oluşturun

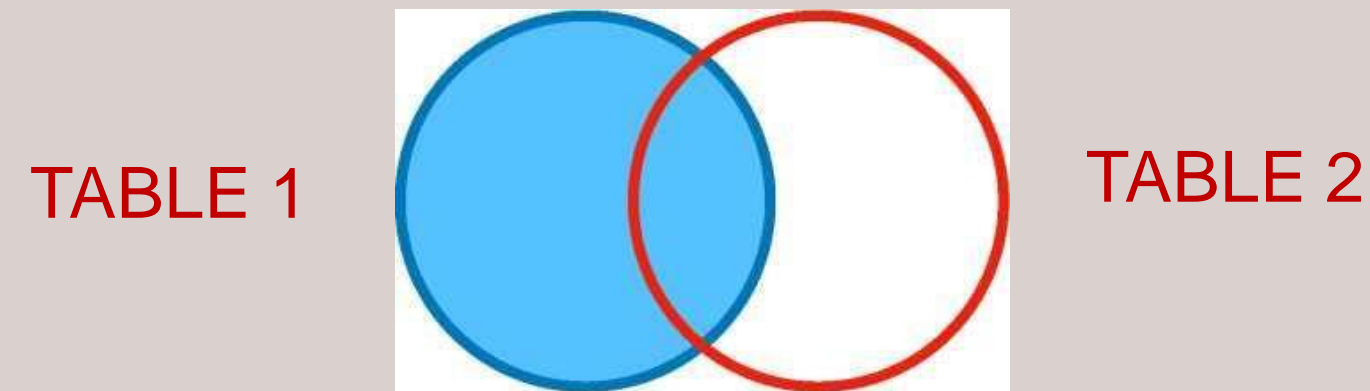
```
SELECT sirketler.sirket_isim, siparisler. siparis_id, siparisler. siparis_tarihi  
FROM sirketler INNER JOIN siparisler  
ON sirketler.sirket_id = siparisler.sirket_id;
```

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

**NOT :**

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

# LEFT JOINS



```
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) İlk tablodaki datalara 2.tablodan gelen ek datalar varsa bu ek datalar ortak datalar icin gosterilir ancak ortak olmayan datalar icin o kisimler bos kalir
- 3) İlk yazdiginiz Tablonun tamamini aldigi icin hangi tabloyu istedigimize karar verip once onu yazmaliyiz

# RIGHT JOINS

TABLE 1

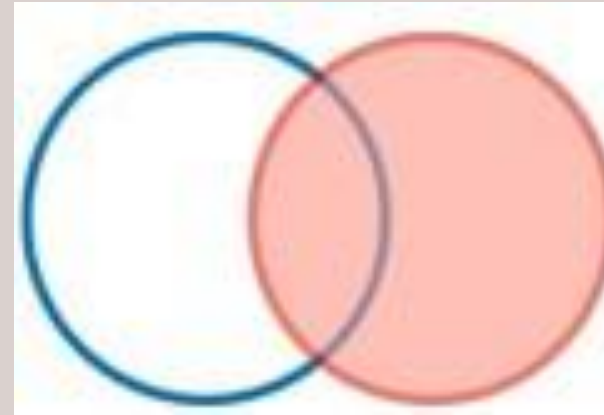


TABLE 2

```
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 gosterilir.
- 2) Ikinci tablodaki datalara 1.tablodan gelen ek datalar varsa bu ek datalar ortak datalar icin gosterilir ancak ortak olmayan datalar icin o kisimler bos kalir

# FULL JOINS

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

NOT :

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

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	-	-



# 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 kullanılarak **SELECT**, **INSERT**, **UPDATE**, veya **DELETE** statement ile calisan **wildcards**'a(özel sembol) izin verir.. Ve bize **pattern matching** yapma imkani verir.

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

```
INSERT INTO musteriler (id, isim, gelir) VALUES (1001, 'Ali', 62000);
```

```
INSERT INTO musteriler (id, isim, gelir) VALUES (1002, 'Ayse', 57500);
```

```
INSERT INTO musteriler (id, isim, gelir) VALUES (1003, 'Feride', 71000);
```

```
INSERT INTO musteriler (id, isim, gelir) VALUES (1004, 'Fatma', 42000);
```

```
INSERT INTO musteriler (id, isim, gelir) VALUES (1005, 'Kasim', 44000);
```

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 yazdıran QUERY yazın

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

ISIM	GELIR
Ayşe	57500
Feride	71000

**SORU :** Isminin icinde er olan musterilerin isimlerini ve gelir'lerini yazdıran QUERY yazın

```
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';
```

ID	ISIM	GELIR
1004	Fatma	42000

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

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

ID	ISIM	GELIR
1004	Fatma	42000
1005	Kasim	44000

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

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

ID	ISIM	GELIR
1002	Ayşe	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** => Daha karmaşık sorgular için herhangi bir kod, metin içerisinde istenilen yazı veya kod parçasının aranıp bulunmasını sağlayan kendine ait söz dizimi olan bir yapıdır.

(REGEXP\_LIKE) PostgreSQL de “ ~ ” karakteri ile kullanılır

```
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** : İlk harfi h, son harfi t olup 2. harfi a veya i olan 3 harfli kelimelerin tüm bilgilerini yazdıran QUERY yazın

```
SELECT *  
FROM kelimeler  
WHERE kelime ~ 'h[ai]t';
```

---

# LIKE Condition

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

```
SELECT *  
FROM kelimeler  
WHERE kelime ~ 'h[a-k]t';
```

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

**SORU** : Icinde m veya i olan kelimelerin tum bilgilerini yazdiran QUERY yazin

```
SELECT *  
FROM kelimeler  
WHERE kelime ~ '[mi]';
```

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

**SORU** : a veya s ile baslayan kelimelerin tum bilgilerini yazdiran QUERY yazin

```
SELECT *  
FROM kelimeler  
WHERE kelime ~ '^[as]';
```

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 kelime ~ '[mf]$';
```

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 kelime !~ '[_eio]';
```

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

---

## LIKE Condition

LIKE: Sorgulama yaparken belirli patternleri(KAlıp ifadelerle sorgu) kullanabilmezi sağlar

ILIKE: Sorgulama yaparken büyük/küçük harfe duyarsız olarak eşleştirir.

LIKE = ~~

ILIKE = ~~\*

NOT LIKE = !~~

NOT ILIKE = !~~\*

NOT REGEXP\_LIKE = !~\*

NOT REGEXP\_LIKE = !~

---

# UPPER – LOWER - INITCAP

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

**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

--DISTINCT clause, çağrılan terimlerden tekrarlı olanların sadece birincisini alır.

```
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');
```

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

SELECT DISTINCT urun\_isim  
FROM muster\_iurun;

URUN_ISIM
Elma
Portakal
Kaysi
Armut

SELECT DISTINCT muster\_iisim  
FROM muster\_iurun;

MUSTERI_ISIM
Veli
Ayşe
Elif
Adem
Ali

Tabloda kac farkli meyve vardir ?

SELECT COUNT(DISTINCT urun\_isim) AS urun\_cesit\_sayisi  
FROM muster\_iurun;

URUN_CESIT_SAYISI
4



# FETCH NEXT (SAYI) ROW ONLY- OFFSET

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	Ayşe	Armut
40	Veli	Kaysi

2) Sirali tablodan ilk 3 kaydi listeleyin

```
SELECT *  
FROM musteri_urun  
ORDER BY urun_id  
FETCH NEXT 3 ROW ONLY;
```

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  
OFFSET 3 ROW  
FETCH NEXT 4 ROW ONLY;
```

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

# ALTER TABLE STATEMENT

**ALTER TABLE** statement tabloda **add**, Type(**modify**)/Set, Rename veya **drop columns** islemleri için kullanılır.

**ALTER TABLE** statement tabloları yeniden isimlendirmek için de kullanılır.

```
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 field 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 field ekleme

```
ALTER TABLE personel  
ADD cinsiyet varchar(20) , ADD 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) TYPE/SET sutunlarin ozelliklerini degistirme

ALTER TABLE isciler















ALTER COLUMN ulke\_adi TYPE varchar(30),

ALTER COLUMN ulke\_adi SET NOT NULL;

**Not:** String data türünü numerik bir data türüne dönüştürmek istersek;

ALTER COLUMN fieldname

TYPE int USING(fieldname::int) şeklinde yaparız.

Columns						
		Name	Data type	Length/Precision	Scale	Not NULL?
		id	integer   v			<input checked="" type="checkbox"/>
		isim	character varying   v	50		<input type="checkbox"/>
		sehir	character varying   v	50		<input type="checkbox"/>
		maas	integer   v			<input type="checkbox"/>
		sirket	character varying   v	20		<input type="checkbox"/>
		ulke_adi	character varying   v	30		<input checked="" type="checkbox"/>
		cinsiyet	character varying   v	20		<input type="checkbox"/>

---

# TRANSACTION (Begin – Savepoint – rollback - commit)

- Transaction veritabanı sistemlerinde bir işlem başladığında başlar ve işlem bitince sona erer. Bu işlemler veritabanı oluşturma , veri silme , veri güncelleme, veriyi geri getirme gibi işlemler olabilir .

```
CREATE TABLE ogrenciler2  
(  
  id serial,  
  isim VARCHAR(50),  
  veli_isim VARCHAR(50),  
  yazili_notu real  
);
```

```
BEGIN;  
INSERT INTO ogrenciler2 VALUES(default, 'Ali Can', 'Hasan',75.5);  
INSERT INTO ogrenciler2 VALUES(default, 'Merve Gul', 'Ayse',85.3);  
savepoint x;  
INSERT INTO ogrenciler2 VALUES(default, 'Kemal Yasa', 'Hasan',85.6);  
INSERT INTO ogrenciler2 VALUES(default, 'Nesibe Yilmaz', 'Ayse',95.3);  
savepoint y;  
INSERT INTO ogrenciler2 VALUES(default, 'Mustafa Bak', 'Can',99);  
INSERT INTO ogrenciler2 VALUES(default, 'Can Bak', 'Ali', 67.5);  
ROLLBACK to y;  
COMMIT;
```

---



---

--Transaction kullanımında SERIAL data türü kullanımı tercih edilmez. Savepointten sonra eklediğimiz veride sayaç mantığı ile çalıştığı için sayacta en son hangi sayıda kaldıysa oradan devam eder

NOT :PostgreSQL de Transaction kullanımı için «Begin;» komutuyla başlarız sonrasında tekrar yanlış bir veriyi düzeltmek veya bizim için önemli olan verilerden sonra ekleme yapabilmek için "SAVEPOINT savepointismi" komutunu kullanırız ve bu savepointe dönebilmek için "ROLLBACK TO savepointismi" komutunu kullanırız ve rollback çalıştırıldığında savepoint yazdığımız satırın üstündeki verileri tabloda bize verir ve son olarak Transaction'ı sonlandırmak için mutlaka "COMMIT" komutu kullanılır.

---



---

# INTERVIEW QUESTION

CREATE TABLE personel

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

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

ISIM	ID
Brad Pitt	456789012

# 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)  
FROM personel);
```

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 yuksek 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