



BATCH :

B107 AWS/DEVOPS

LESSON :

SQL

DATE :

07.01.2023

SUBJECT :

SQL

ZOOM GİRİŞLERİNİZİ LÜTFEN LMS SİSTEMİ ÜZERİNDEN YAPINIZ



/ techproeducation

TECHPROEDUCATION



techproeducation.com



+1 (917) 768-7466

---

# SQL

Structured Query Language  
Yapılandırılmış Sorgu Dili

TECHPROED

---

# DATABASE

**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**

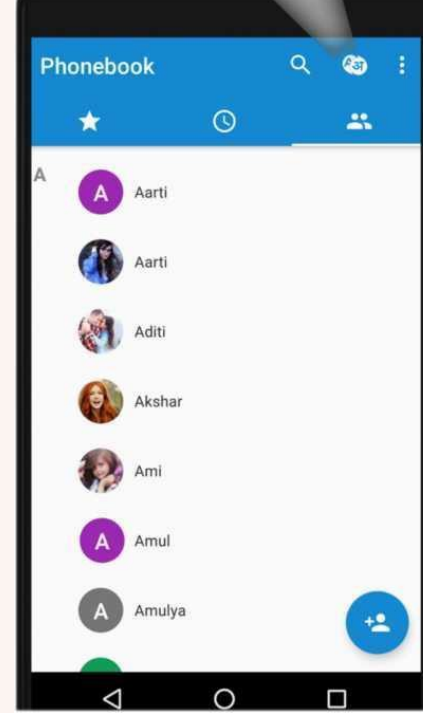
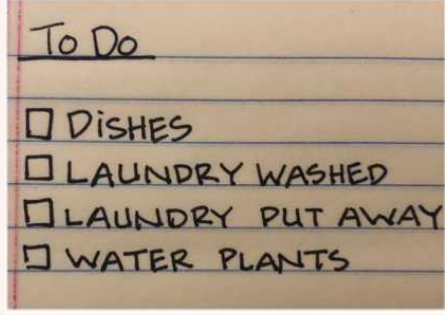
```
public class facebook {  
  
    public static void main(String[] args) {  
        Scanner scan = new Scanner(System.in);  
        System.out.println("Enter your name");  
        String name = scan.nextLine();  
  
        System.out.println("Enter your surname");  
        String surname = scan.nextLine();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



# THE CHIPROD



# DATABASE (VERITABANI) 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 (**DataBaseManagementSystem**) ile kontrol edilir.

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

# DATABASE'IN FAYDALARI NELERDİR

- 1) Yüksek miktarda bilgi depolanabilir
- 2) Oluşturma, Okuma, Değiştirme ve Silme kolaylığı  
**Create, Read, Update, Delete (CRUD)**
- 3) Girisin kolay ve kontrollü olması
- 4) Dataya ulaşım kolaylığı
- 5) Güvenlik

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



TECHNOLOGY



# DATABASE VALIDATION(DOĞRULAMA) TESTİ

**Kaydol** ×  
Hızlı ve kolaydır.

Doğum Tarihi ?

Cinsiyet ?  
☐ Kadın ☐ Erkek ☐ Özel

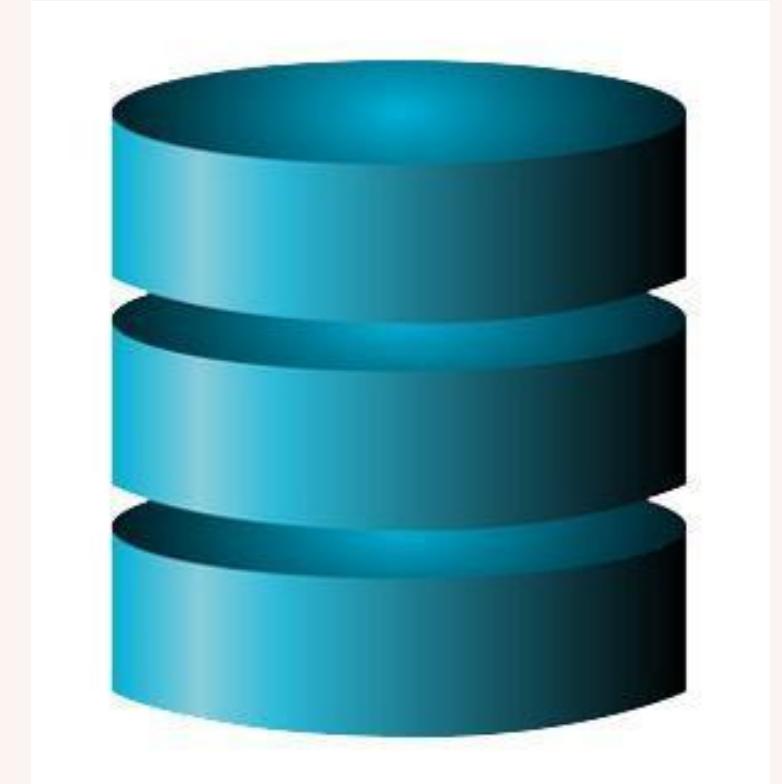
Kaydol düğmesine tıklayarak, Koşullarımızı, Veri İlkemizi ve Çerezler İlkemizi kabul etmiş olursun. Bizden SMS Bildirimleri alabilir ve bu bildirimleri istediğin zaman durdurabilirsin.

**Kaydol**

User Interface



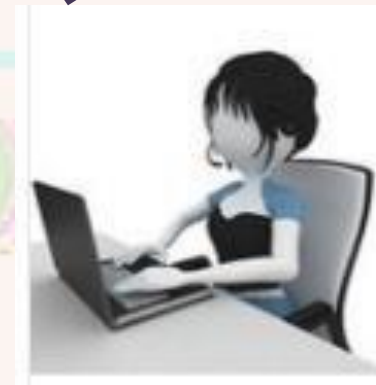
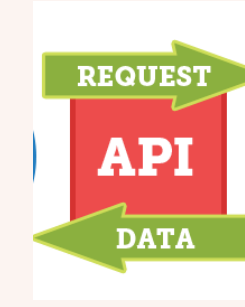
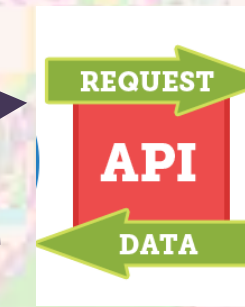
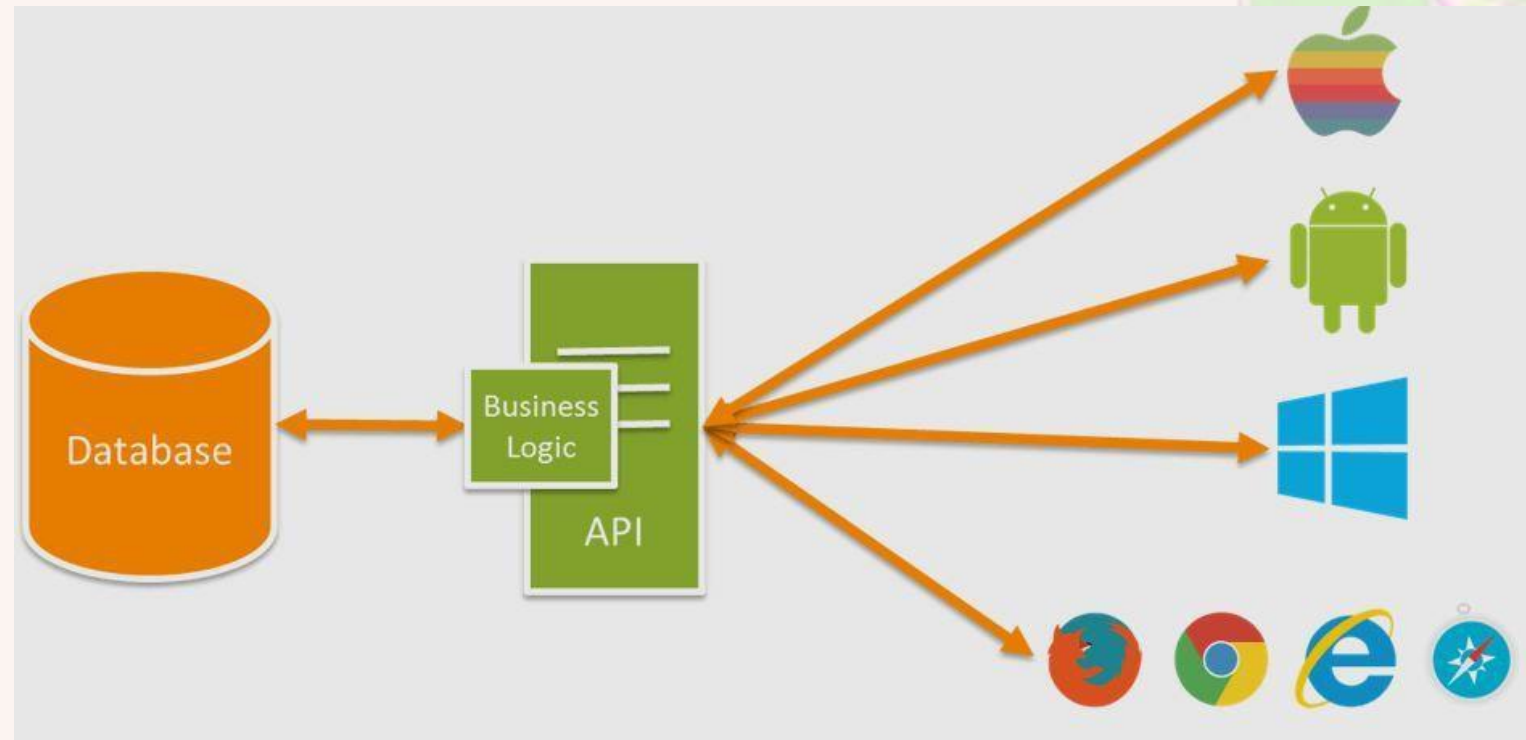
API



Database

# API

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



TECHNOLOGY



---

## END To END (E2E) Testing

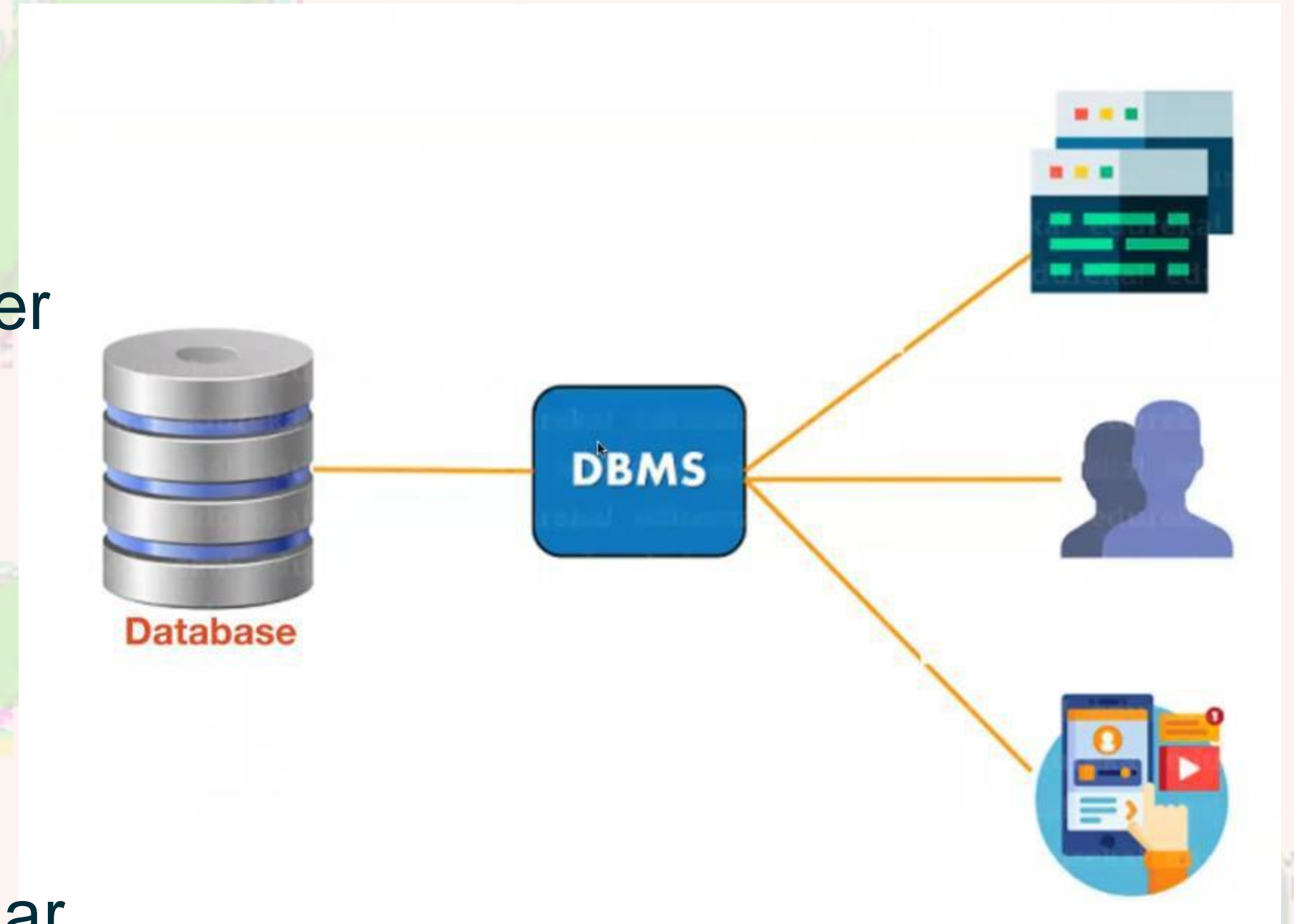
- 1 Eger datayi User Interface (UI) kullanarak yolladiysaniz
    - A) Datayi UI dan arama fonksiyonunu kullanarak dogrula (Selenium)
    - B) Datayi SQL kodlarini kullanarak dogrula (SQL + Selenium)
    - C) Datayi API kodlarini kullanarak dogrula (API + Selenium)
  - 2) Eger datayi SQL kodlarini kullanarak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



## Data Base Management System (DBMS)

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

- Database'e erişimi düzenler
- **C**reate, **R**ead, **U**ppdate, **D**eleete işlemlerini düzenler
- Data güvenliğini sağlar
- Formlar oluşturur ve formları işler,
- Sorgular oluşturur ve sorgular iletilir,
- Raporlar oluşturur ve raporları işletir,
- Uygulamayı kontrol eder
- Diğer uygulamalarla (Application) iletişimi sağlar.



# 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) ^====

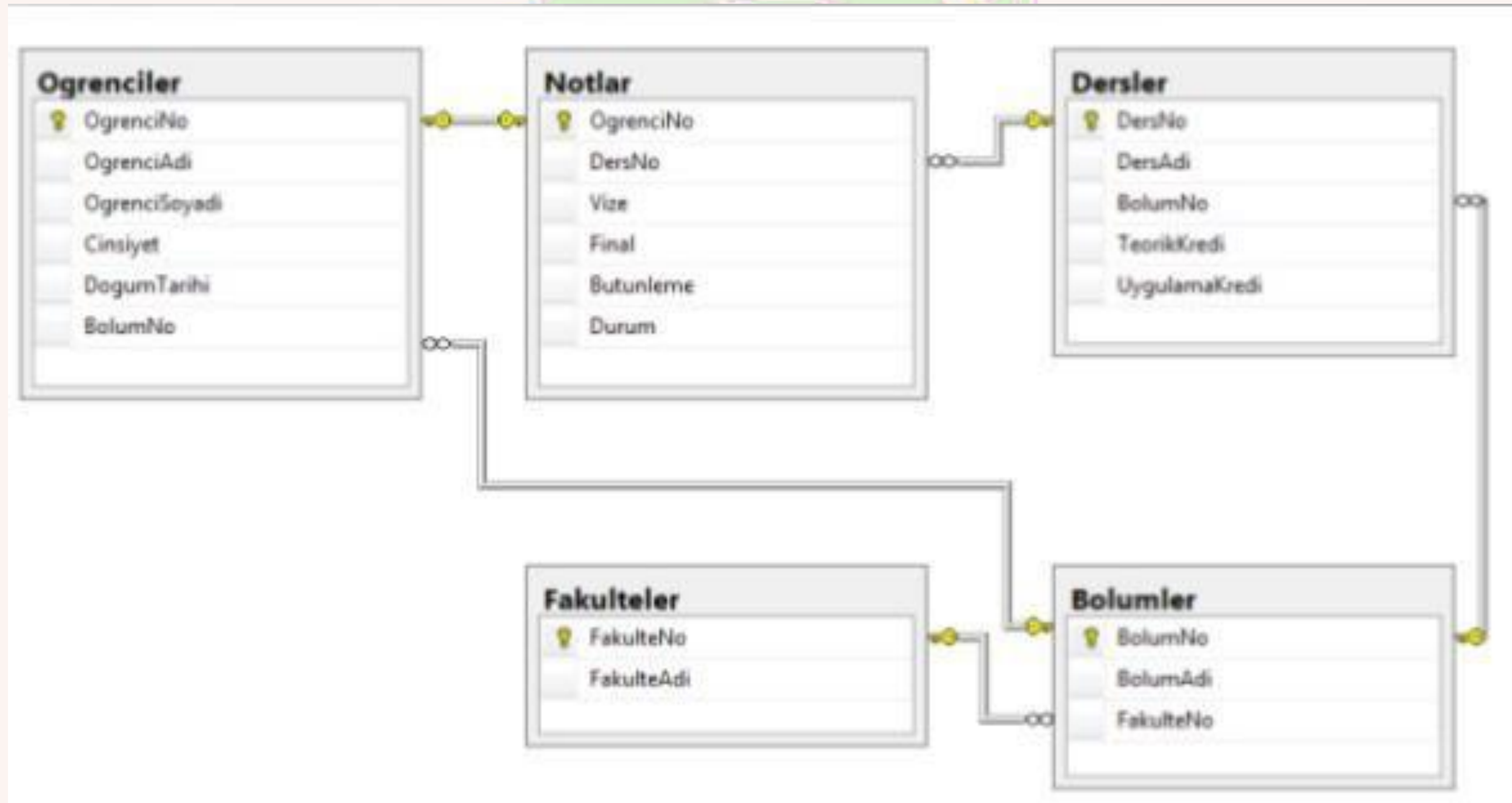
Column (Field) ^====

Column (Field) ^====

TECHNICAL



# RELATIONANAL DATABASES (ILISKILI TABLOLAR)



# RELATIONAL DATABASES (İLISKILI TABLOLAR)

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

➤ Tablolar arası ilişkiler net olmalıdır.

➤ Tablolar arası geçiş kolay olmalıdır

➤ Tabloların ve ilişkilerin butunüne SCHEMA denir

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

id	ogrenci_id	ders_id
1	1	3
2	1	5
3	2	1
4	3	4
5	4	2
6	4	3

id	ders_adi
1	Matematik
2	Tarih
3	Edebiyat
4	Yazılım
5	İstatistik

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



# Cok Kullanilan Relational Databases(SQL Database)



**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.

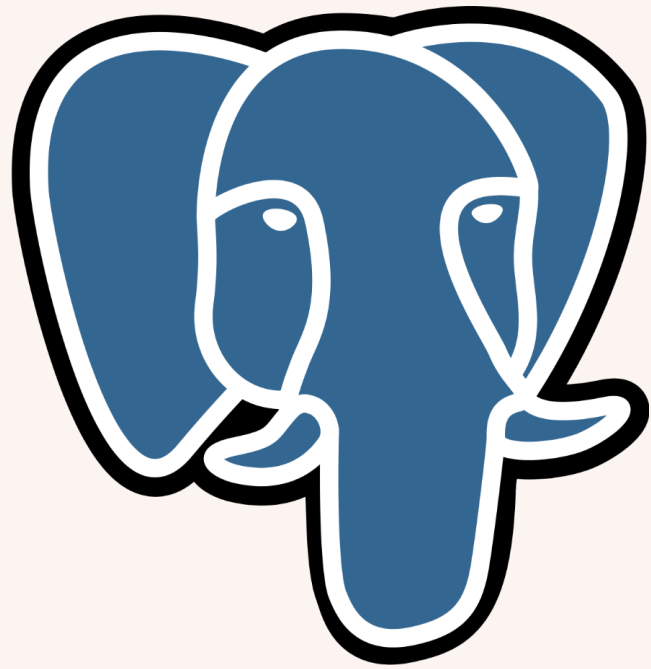


**MySQL Server** : İsveçli MySQL firması tarafından geliştirildi.  
2010'da Oracle satın aldı

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

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

# Cok Kullanilan Relational Databases(SQL Database)



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

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

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

**ORACLE**  
DATABASE



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

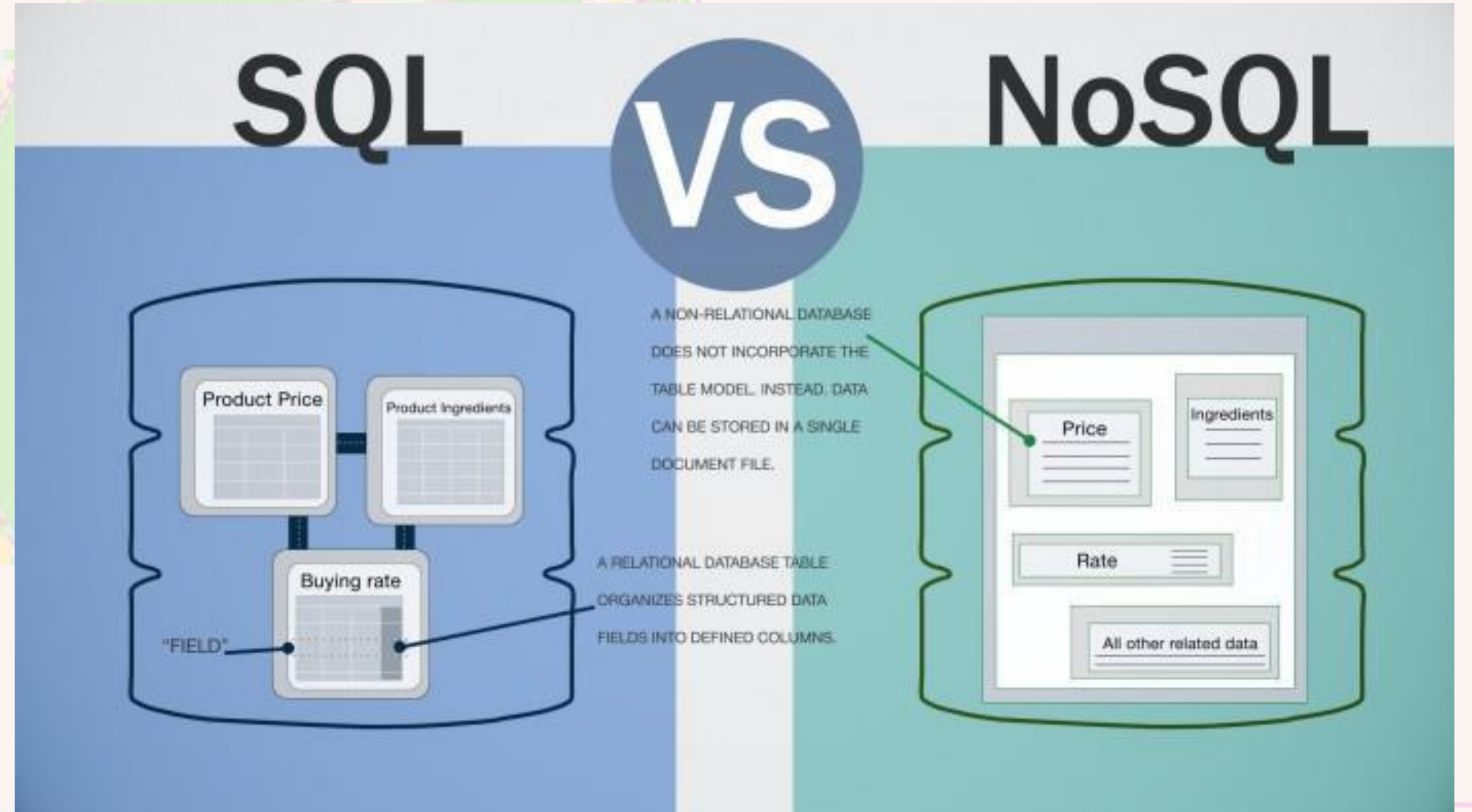
**PL/SQL** SQL komutlarini ozellikle karsilamak uzere dizayn edilmistir.

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



# 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 Komutları

## SQL komutları 4 ana gruba ayrılır:

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

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

**SELECT** : Veritabanındaki verileri alır.

### 2. Veri Kullanma Dili (Data Manipulation Language - DML)

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

**INSERT** : Veritabanına yeni veri ekler.

**UPDATE** : Veritabanındaki verileri günceller.

**DELETE** : Veritabanındaki verileri siler.

---



---

# SQL 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.

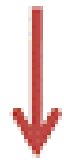
**REVOKE** : Bir kullanıcıya verilen yetkiyi geri almak için kullanılır.

---

# 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.

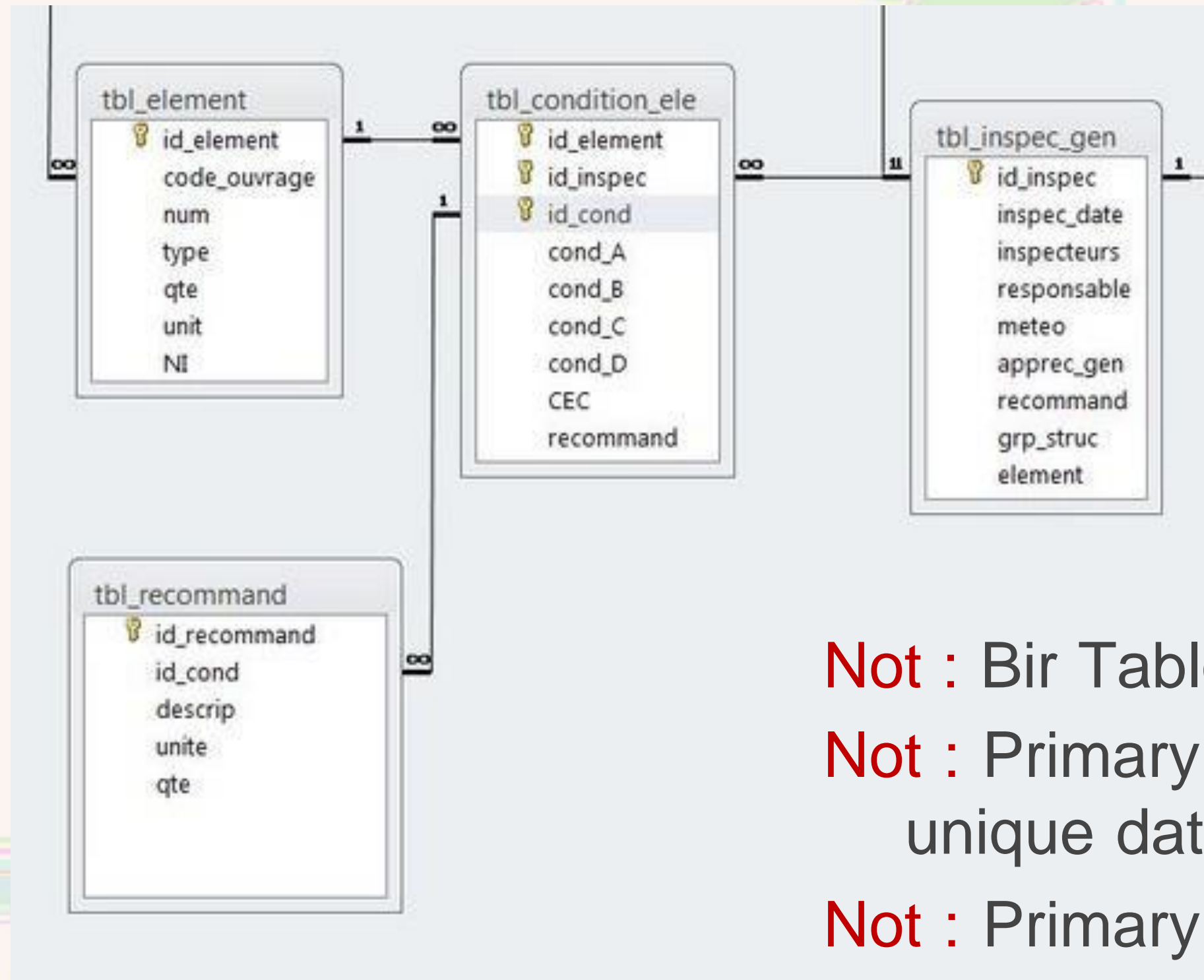
Relational veri tabanlarında (relational database management system) mutlaka birincil anahtar olmalıdır.

**Not :** Bir Tabloda en fazla 1 tane primary Key olabilir.

**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

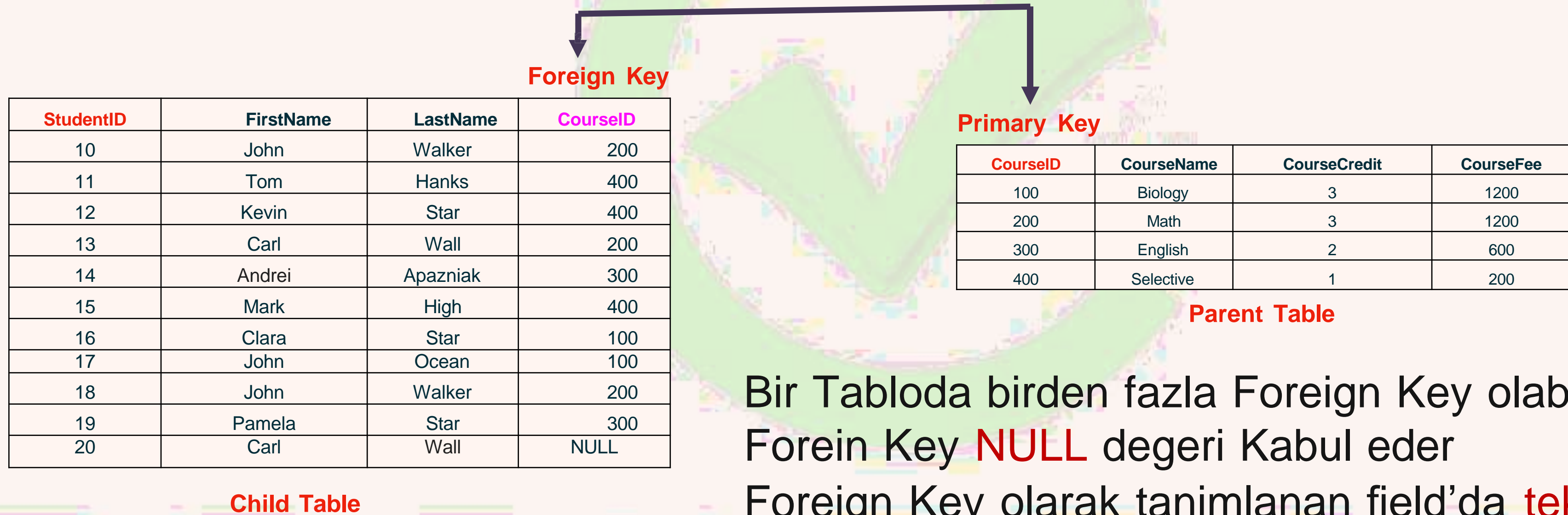
Email	FirstName	LastName
JWalker@gmail.com	John	Walker
THanks@gmail.com	Tom	Hanks
KStar@gmail.com	Kevin	Star
CWall@gmail.com	Carl	Wall
AApazniak@gmail.com	Andrei	Apazniak
MHigh@gmail.com	Mark	High
CStar@gmail.com	Clara	Star
JOcean@gmail.com	John	Ocean
JWalker01@gmail.com	John	Walker
PStar@gmail.com	Pamela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

Genel olarak kayıt eklenmeden önce üretilen sıra numarası gibi sayisal degerlere **Surrogate Key** denir

# Foreign Key

**Foreign Key** iki tablo arasında relation oluşturmak için kullanılır

**Foreign Key** başka bir tablodaki Primary Key ile ilişkilendirilmiş olmalıdır



Bir Tabloda birden fazla Foreign Key olabilir

Foreign Key **NULL** değeri kabul eder

Foreign Key olarak tanımlanan field'da **tekrarlar** olabilir

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



# 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 yoneticisi 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 taşımiyorsa, 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 en fazla 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

Duplication(Cift Kullanım)’a izin vermez

TECHNIPROED

---

# Örnek Okul Tablosunun Bir Parçası

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

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

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

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

ogrenci 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	Pensylvania
18	1242 W 5th Street	22133	Baytown	Texas
19	1243 SE 55th Street	74352	Beachwood	Ohio
20	1244 SW 17th Street	22314	Avery	Texas

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

# Related Tablolarla 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?
- 3) Pamela Star'in ogretmenlerinin isimleri nedir?

# SQL Data Types

## String Data Types

Data Type	Aciklama
<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.

Alphabets	CHAR ( 4 )	Data Size	VARCHAR ( 4 )	Data Size
' '	' '	4 bytes	' '	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

# SQL Data Types

## Numeric Data Types

Data Type	Aciklama
<b>number(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>number(5, 2)</b>” virgulden once 3,virgulden sonra 2 rakam olan sayi ==&gt; <b>123,45</b></p> <p>2) “<b>number(4, 2)</b>” ==&gt; <b>123,45</b> ==&gt; <b>error verir</b></p> <p>3) “<b>number(7)</b>” ondalik kısmi olmayan 7 basamakli sayi demektir ==&gt; <b>12345,67’i kabul eder ama 12345 olarak depolar</b> <b>Note:</b> “<b>number(7)</b>” ve “<b>number(7, 0)</b>” ayni seydir</p> <p>4) “<b>number(7, -2)</b>” rounds the numeric value to hundreds. ==&gt; <b>1234567,89</b> ==&gt; <b>1234600</b></p>



# 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>dd - MMM - yy</b>”. <b>Ornegin</b> 13 - Apr - 20</p> <p>Tarih formatini “<b>ALTER SESSION SET NLS_DATE_FORMAT = “YYYY-MM-DD</b>” kodu kullanilarak degistirilebilir. Koddan sonra tarih 2020 - 04 – 13 olur.</p>

TECHPROF

---

# SQL Data Types

## BLOB Data Types

<i>Data Type</i>	<i>Aciklama</i>
<b>BLOB</b>	“ <b>BLOB</b> ” , “ <b>B</b> inary <b>L</b> arge <b>O</b> Bjects” demektir “ <b>BLOB</b> ” resim,video,ses gibi datalari binary formatina cevirerek depolar.

TECHPROED

---



# 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.

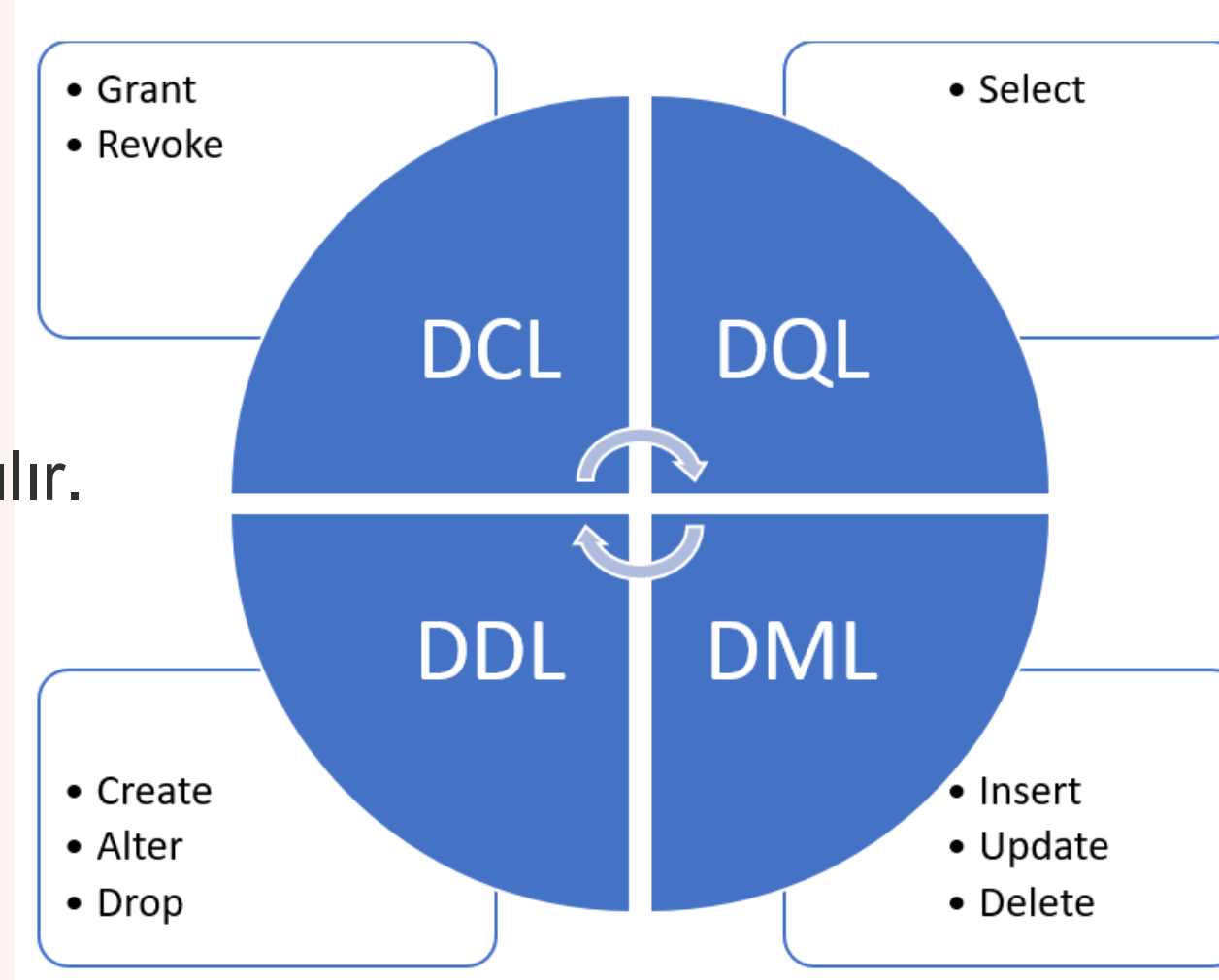
**SELECT** : Veritabanındaki verileri alır.

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.



---

# SQL

Structured Query Language  
Yapılandırılmış Sorgu Dili

Tablo Oluşturma  
Tabloya Data Ekleme  
Constraints

TECHPROED

---

---

# Table Nasıl Olusturulur?

## 1) Create from Scratch

```
CREATE TABLE ogrenciler  
(  
id char(4),  
Isim_soyisim varchar(50),  
not_ort number(5,2),  
kayit_tarihi date  
);
```

## 2) Var olan tablodan yeni tablo olusturmak

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

TECHPROFED

---



---

# Tabloya Nasıl Data Eklenir (INSERT INTO )?

- **INSERT INTO** komutu, Oracle 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');

TECHPROED

---

# Table Nasıl Olusturulur?

## 1) Create from Scratch

### Practice1:

“tedarikciler” isminde bir tablo olusturun ve “tedarikci\_id”, “tedarikci\_ismi”, “tedarikci\_adres” ve “ulasim\_tarihi” field’lari olsun. Tablo olusturduktan sonra 3 tane kayıt ekleyelim.

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

## 2) Var olan tablodan yeni tablo olusturmak

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

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

# Tabloya Nasıl Data Eklenir (INSERT INTO )?

## Practice 1:

“*ogretmenler*” isminde bir SQL tablosu 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.

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

## Ekleme :

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

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

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



# 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 number(5,2),  
  adres varchar(100),  
  kayit_tarihi date  
);
```

T E C H P R O E D

# 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 number(5,2),  
  adres varchar(100),  
  kayit_tarihi date  
);
```

T E C H P R O E D

# Tabloya Nasil Data Eklenir (INSERT INTO )?

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

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

```
CREATE TABLE personel
(  
id char(10),  
isim varchar(50) NOT NULL,  
soyisim varchar(50),  
email varchar(50),  
ise_baslama_tar date,  
maas number(6)  
);
```

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

```
cannot insert NULL into ("SQL_LFGUHVR SOWWDACLEMH RMQG CJQ"."STUDENTS"."NAME")
```

TECHPROF



---

# SQL

Structured Query Language  
Yapılandırılmış Sorgu Dili

Primary Key, Foreign Key  
Check Constraint  
Data Update

TECHPROED

---

## Önceki Dersten Hatırladıklarımız

- 1 CREATE TABLE tabloismi (field1 ,field..); parantez icine field isimleri,data tipi ve size yazilir
- 2 CREATE TABLE tabloismi AS SELECT (field1 ,field2..) FROM anaTabloismi
- 3 SELECT \* FROM tabloismi bir tablodaki tum datolari bize veriyor
- 4 INSERT INTO tabloismi VALUES (degerler) tabloda bulunan tum field'lar icin field sirasina ve data tipine uygun olarak bilgi eklemek gereklidir, Constraints'lere dikkat edilmelidir
- 5 INSERT INTO tabloismi (eklemek istedigimiz field'lar) VALUES (ekleyecegimiz degerler)
- 6 CONSTRAINTS
  - NOT NULL : yazildigi field'in bos birakilmasini engeller (Sinirlandirir)
  - UNIQUE: yazildigi field'a eklenecek datalarin benzersiz olmasini control eder, null degerini kabul eder, birden fazla null degerini de Kabul eder
  - PRIMARY KEY : Bir tabloda en fazla bir tane PK bulunur.(olmayabilir,olursa 1 tane olur)
    - NULL degeri Kabul etmez
    - UNIQUE'dir duplication'a izin vermez
    - Bir tabloda PK varsa yazili oldugu kaydi tek basina ifade eder
    - PK gunluk hayatta bir anlam ifade ediyorsa (email gibi) NATURAL, gunluk hayatta anlam ifade etmeyen sira numarasi gibi PK ler icin SURROGATE, eger bir field degil de birden fazla field'in birlesmesiyle olusuyorsa COMPOSE PK denir.

## 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 number(5,2),  
  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 number(5,2),  
  adres varchar(100),  
  kayit_tarihi date,  
  CONSTRAINT ogrenciler_pk PRIMARY KEY(id)  
);
```

The logo for TECHNIPROOD is located at the bottom of the slide. It features the word "TECHNIPROOD" in a stylized, blocky font. Each letter is composed of multiple overlapping, semi-transparent shapes in various colors (blue, green, yellow, orange, red, purple), creating a vibrant, multi-colored effect. The letters are closely spaced and have a slight 3D appearance.

## Bir Tabloya “Primary Key” Nasıl Eklenir

### Practice 2:

“sehirler” isimli bir Table olusturun. Tabloda “alan\_kodu”, “isim”, “nufus” field’lari olsun. Isim field’i bos birakilmasin.

1.Yontemi kullanarak “alan\_kodu” field’ini “Primary Key” yapin

```
CREATE TABLE sehirler  
(  
alan_kodu number(3) PRIMARY KEY,  
isim varchar(20) NOT NULL,  
nufus number(8)  
);
```

### Practice 3:

“ogretmenler” isimli bir Table olusturun. Tabloda “id”, “isim”, “brans”, “cinsiyet” field’lari olsun. Id field’i tekrarli deger Kabul etmesin.

2.Yontemi kullanarak “id ve isim” field’lerinin birlesimini “primary key” yapin.(birleşim unique olacak... id nin farklı olması yeterli)

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

# 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.

TECHNIPROED



# 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

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

Parent Table

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 tedarikciler (tedarikci_id) );
```

```
CREATE TABLE tedarikciler
```

```
(  
  tedarikci_id char(10),  
  tedarikci_ismi varchar(50),  
  iletisim_isim varchar(50),
```

```
  CONSTRAINT tedarikci_pk PRIMARY KEY (tedarikci_id)
```

```
);
```

# Tabloya “Foreign Key” Nasıl Eklenir ?

## Practice 5:

“*tedarikciler*” isimli bir Tablo oluşturun. İçinde “*tedarikci\_id*”, “*tedarikci\_isim*”, “*iletisim\_isim*” field’leri olsun. “*tedarikci\_id*” ve “*tedarikci\_isim*” fieldlarını birleştirerek **Primary Key** oluşturun. “*urunler*” isminde başka bir tablo oluşturun. İçinde “*tedarikci\_id*” ve “*urun\_id*” fieldları olsun. “*tedarikci\_id*” ve “*urun\_id*” fieldlarını birleştirerek Foreign Key oluşturun. (ortak olacak sütunlar tamamen aynı elemanlardan oluşmalı=>ikisinin ilk sütunu, (101,'aa' ) olmalı mesela)

```
CREATE TABLE tedarikciler
```

```
(  
  tedarikci_id number(10),  
  tedarikci_isim varchar(50),  
  iletisim_isim varchar(50),  
  CONSTRAINT tedarikci_pk PRIMARY KEY  
    (tedarikci_id,tedarikci_isim)  
);
```

```
CREATE TABLE urunler
```

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



## 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** : sehirler2 tablosu olusturalim, nufusun negatif deger girilmemesi icin sinirlendirme (Constraint) koyalım

```
CREATE TABLE sehirler2
(
  alan_kodu number(3) PRIMARY KEY,
  isim varchar(20) NOT NULL,
  nufus number(8) CHECK (nufus>0)
);
```

---

# SQL

Structured Query Language  
Yapılandırılmış Sorgu Dili

Update Data  
Select

TECHPROED

---

# Önceki Dersten Hatırladıklarımız

1 Primary Key'ı 2 yöntemle tanımlayabiliyoruz

- PK yapmak istediğimiz field'ın data tipi ve size'ini yazdıktan sonra PRIMARY KEY yazarız
- Tüm fieldlar yazıldıktan sonra alt satıra geçip CONSTRAINT olarak tanımlıyoruz

CONSTRAINT pk\_isim PRIMARY KEY (field1,field2...)

2 Foreign Key : Baska tablodaki Primary Key ile relation oluşturmak için kullanılır, tabloları birbirine bağlar

- tanımlamak için en alt satırda CONSTRAINT oluşturulur. FK fieldını tek başına tanımlamak yetmez related olduğu PK de yazılmalıdır

CONSTRAINT fk\_ismi FOREIGN KEY (field1,field2..) REFERENCES parent\_tablo (PK)

- PK ve FK data tipleri uyumlu olmalıdır

3 Parent – Child Tablo : PK 'ın olduğu tablo Parent, FK'in olduğu tablo child olarak adlandırılır

- Parent table'da olmayan FK'i child tabloda kullanamayız, böyle bir FK içeren yeni kayıt oluşturmak istersek hata alırız ve ekleme yapılmaz

- Parent tablodan bir kayıt silmeden önce onunla ilişkili Child tablodaki tüm kayıtlar silinmelidir. Aksi takdirde hata alırız, SQL silme işlemine izin vermez

- Child table silinmeden parent table silinemez

- FK null değeri alabilir ancak bu durumda parent tablodan hiç bir degree ulaşamaz

4 Check Constraint : Bir field'da eklenecek değerlerin yazdığımız şartı gerçekleştirmesini kontrol eder ve şartı sağlamayan kayıtların eklenmesine izin vermez



# SELECT KOMUTU

## 1) Tablodaki Tum Field'leri Cagirma

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

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

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

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

TECHPROED

---

# 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

TECHNIPROED

---

# SELECT KOMUTU

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

```
SELECT adres  
FROM students;
```

Tablodan sadece adres field'indaki tum datalari getirir

ADRES
Ankara
Ankara
Istanbul

```
SELECT adres  
FROM students  
WHERE yazili_not=85;
```

Tablodan sadece yazili notu 85 olanlarin adres field'indaki datalari getirir

ADRES
Ankara
Istanbul

TECHPROJED



# SELECT KOMUTU

## 3) Tablodan Birden Fazla Field'i Çağırma

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

Tablodan adres ve isim field'indeki tüm dataları 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 kayıtların adres ve isim field'indeki dataları getirir

ADRES	ISIM
Ankara	Merve Gul
Istanbul	Kemal Yasa

---

# SELECT KOMUTU

4) 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

TECHPROED

---

# Tablodaki Data Nasıl Update Edilir (UPDATE SET)?

Mart\_satislar isimli bir tablo oluşturun, içinde urun\_id, musteri\_isim, urun\_isim ve urun\_fiyat field'leri olsun

```
CREATE TABLE mart_satislar  
(  
  urun_id number(10),  
  musteri_isim varchar(50),  
  urun_isim varchar(50),  
  urun_fiyat number(10)  
);
```

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

- 1) İsmi hatice olan müşterinin urun\_id'sini 30,urun\_isim'ini Ford yapın
- 2) Toyata marka araçlara %10 indirim yapın
- 3) İsmi Ali olanların urunfiyatlarını %15 artırsın
- 4) Honda araçların urun kodu'nu 50 yapın



# Tablodaki Data Nasıl Update Edilir (UPDATE SET)?

1) Bir tedarikciler tablosu oluşturun içinde id, isim ve iletişim\_isim field'ları olsun. Id ve isim'i beraber Primary Key yapın

```
CREATE TABLE tedarikciler
(
  id number(10),
  isim varchar(50),
  iletişim_isim varchar(50),
  CONSTRAINT tedarikci_pk PRIMARY KEY (id, isim)
);
```

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

```
UPDATE tedarikciler
SET isim = 'KRM', iletişim_isim = 'Kemal Kan'
WHERE id =1;
```

2) İçine 3 kayıt ekleyin (1, 'ACB', 'Ali Can'), (2, 'RDB', 'Veli Gul'), (3, 'KMN', 'Ayse Gulmez').

```
INSERT INTO tedarikciler VALUES (1, 'ACB', 'Ali Can');
INSERT INTO tedarikciler VALUES (2, 'RDB', 'Veli Gul');
INSERT INTO tedarikciler VALUES (3, 'KMN', 'Ayse Gulmez');
```

4) İsmi RDB olan tedarikcinin iletişim isim'ini Kemal Yasa yapın

```
UPDATE tedarikciler
SET iletişim_isim='Kemal Yasa'
WHERE isim='RDB';
```

# Tablodaki Data Nasıl Update Edilir (UPDATE SET)?

**Practice 11:** verilen tablolara göre aşağıdaki işlemleri yapın.

a) Urnler tablosundan Ali Can'ın aldığı ürünün ismini, tedarikci tablosunda irtibat\_isim Merve Temiz olan şirketin ismi ile değiştirin

b) TV satın alan müşterinin ismini, Apple'ın irtibat\_isim'i ile değiştirin

```
CREATE TABLE tedarikci
(
  id number(5) PRIMARY KEY,
  isim varchar(50), irtibat_isim
  varchar(50)
);
```

```
INSERT INTO tedarikci VALUES(100, 'IBM', 'Ali Can');
INSERT INTO tedarikci VALUES(101, 'APPLE', 'Merve Temiz');
INSERT INTO tedarikci VALUES(102, 'SAMSUNG', 'Kemal Can');
INSERT INTO tedarikci VALUES(103, 'LG', 'Ali Can');
```

```
CREATE TABLE urunler
(
  tedarikci_id number(5),
  urun_id number(11),
  urun_isim varchar(50),
  musteri_isim varchar(50),
  CONSTRAINT urunler_fk FOREIGN KEY(tedarikci_id) REFERENCES tedarikci(id )
);
```

```
INSERT INTO urunler VALUES(100, 1001, 'Laptop', 'Suleyman');
INSERT INTO urunler VALUES(101, 1002, 'iPad', 'Fatma');
INSERT INTO urunler VALUES(102, 1003, 'TV', 'Ramazan');
INSERT INTO urunler VALUES(103, 1004, 'Phone', 'Ali Can');
```

# Tablodaki Data Nasil Update Edilir (UPDATE SET)?

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

tedarikci

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

urunler

a) Urunler tablosundan Ali Can'in aldigi urunun ismini, tedarikci tablosunda irtibat\_isim Merve Temiz olan sirketin ismi ile degistirin

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

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

```
UPDATE urunler
SETmusteri_isim=(SELECT irtibat_isim
                 FROM tedarikci
                 WHERE isim='APPLE')
WHERE urun_isim='TV';
```



---

# SQL

Structured Query Language  
Yapılandırılmış Sorgu Dili

Update Data  
Delete Data

TECHPROED

---

## Önceki Dersten Hatırladıklarımız

1 **UPDATE**: bu komutla tablomuzdaki istediğimiz dataları değiştirebiliyoruz. WHERE komutu ile değiştirmek istediğimiz dataları net olarak tanımlayıp SET komutu ile yeni değerleri atayabiliriz.

UPDATE tabloismi

SET fieldname=İstenenDeğer

WHERE değiştirmek istediğimiz Datanın Tanımı (tüm mantıksal operatörleri kullanabiliriz =,<,>,OR,AND)

2 **SELECT** : tablodan istediğimiz dataları okumamızı/listelememizi sağlar

- SELECT \* FROM tabloismi : Tablodaki tüm datayı bize getirir
- SELECT \* FROM tabloismi WHERE istenenSart : sarta uyan kayıtlara ait tüm bilgileri getirir
- SELECT fieldismi FROM tabloismi WHERE istenenSart : istenen sarta uyan kaydın yazılan field veya field'lardaki datalarını getirir

3 İhtiyac olursa iç içe sorgu yazılabilir. Sorgunun WHERE veya SET kısmında başka bir sorgu ile elde edilen data kullanılıyorsa buna **SUBQUERY** denir.

---



# Tablodaki Data Nasil Update Edilir (UPDATE SET)?

1) Ogrenciler6 tablosu olusturun. Icinde id,isim,veli\_isim ve not\_ort field'lari olsun. Id field'i birlikte Primary Key olsun.

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

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

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

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

```
INSERT INTO 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, 'Fatma Gun', 'Mehmet',90);
INSERT INTO ogrenciler6 VALUES(127, 'Ali Can', 'Mehmet',91);
```

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

```
INSERT INTO notlar VALUES ('123','kimya',75);
INSERT INTO notlar VALUES ('124', 'fizik',65);
INSERT INTO notlar VALUES ('125', 'tarih',90);
INSERT INTO notlar VALUES ('126', 'Matematik',90);
INSERT INTO notlar VALUES ('127', 'Ingilizce',90);
```

5) Tum ogrencilerin yazili notlarini notlar tablosundaki ile update edin

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



# Tablodaki Data Nasıl Update Edilir (UPDATE SET)?

## Practice 9:

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

## Practice 10:

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

# Tablodan Data Nasıl Silinir (Delete)?

```
CREATE TABLE ogrenciler
```

```
(
```

```
id char(3),
```

```
isim varchar(50),
```

```
veli_isim varchar(50),
```

```
yazili_notu number(3),
```

```
CONSTRAINT ogrenciler_pk PRIMARY KEY (id)
```

```
);
```

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

- 1) “**DELETE FROM** ogrenciler” tablodaki tüm dataları siler, fakat tabloyu silmez.  
“**DELETE FROM** ogrenciler”, kodunu kullanınca boş bir tablo kalır.

no data found

# Tablodan Data Nasıl Silinir (Delete)?

2) **CREATE TABLE** ogrenciler  
(  
id char(3),  
isim varchar(50),  
veli\_isim varchar(50),  
yazili\_notu number(3),  
**CONSTRAINT** ogrenciler\_pk  
**PRIMARY KEY** (id)  
);

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

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

“ **DELETE FROM** ogrenciler **WHERE** isim = 'Ali Can' ” kodu isim olarak Ali Can girilen kaydi (record) siler.

“ **DELETE FROM** ogrenciler **WHERE** yazili\_notu = 85 ” kodu not olarak 85 girilen kaydi (record) siler.

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

no data found



# Tablodan Data Nasil Silinir (Delete)?

3) **CREATE TABLE** ogrenciler  
(  
id char(3),  
isim varchar(50),  
veli\_isim varchar(50),  
yazili\_notu number(3),  
**CONSTRAINT** ogrenciler\_pk  
**PRIMARY KEY** (id)  
);

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

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

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

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

no data found

## Practice 12:

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

- 1) Verilen bilgilerin olduğu bir tablo oluşturun
- 2) Tüm isci'lerin maasına %20 zam yapın
- 3) Muhendis'lerin maasını 7000 yapın
- 4) Muhasebecinin adını 'Sena' soyadını 'Yılmaz' yapın
- 5) Maası 5000 den büyük olanları silin

TECHPROED

---

# Tablodan Data Nasıl Silinir (TRUNCATE)?

- “Truncate” kodu kullanılarak bir tablodaki kayıtlar silinirse verilerin geri getirilme ihtimali olmaz
- “Truncate” kodu geri getirilmesini (rolling back) istemeyeceğiniz tabloları silmek için kullanılır.

TRUNCATE TABLE customers

DELETE FROM customers;

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

DELETE FROM ile sildiğimiz kayıtları geri getirebiliriz ama TRUNCATE ile silinen kayıtlar geri getirilemez.

---



# Tablo Nasıl Silinir (Drop)?

## (Tum Tablo Icerigi ve Tablo Yapisi )

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

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

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

**DROP TABLE** ogrenciler; ogrenciler Tablosunu siler ve RcyleBin'e yollar  
**FLASHBACK TABLE** ogrenciler **TO BEFORE DROP**; dosyayi geri alır.  
**PURGE TABLE** ogrenciler; RcyleBin'de olan dosyayi geri getirilemeyecek sekilde siler  
**DROP TABLE** ogrenciler **PURGE**; Kullandigimiz bir dosyayi getirilemeyecek sekilde siler

**UYARI:** Purge kullandigimizda Tabloyu ve datalari geri getirmek mumkun degildir.

**Purge Kullanmanin Amaci:** Hassas bilgileri silmek istediginizde baska insanların o bilgiye ulasamayacagindan emin olursunuz

---

# SQL

Structured Query Language  
Yapılandırılmış Sorgu Dili

In, Between

TECHPROFED

---



---

## Önceki Dersten Hatırladıklarımız

1 Subquery : iç içe query demektir. Birden fazla tabloyu ilgilendiren update'ler için kullanıyoruz. Update yapılacak tablodan kodumuzu yazmaya başlıyoruz, sonra diğer tablodan bilgi almak için subquery ekleyip SELECT komutu ile ilgili datayı alıyoruz

2 DELETE komutu ile field silmeyiz ancak record silebiliriz, field ismi seçemediğimiz için kodumuz DELETE FROM tablo ismi şeklinde başlıyor, eğer silmek istediğimiz recordlar için bir şart belirteceksek WHERE komutu ekleyebiliriz.

- DELETE FROM ile silinen kayıtlar geri getirilebilir
- DELETE FROM tablo ismi; tablodaki tüm record'ları siler ama tablo kalır

3- TRUNCATE komutu ile tablo yapısı bozulmadan tablodaki TÜM kayıtlar silinir.

- TRUNCATE ile silinen kayıtlar geri getirilemez

4 DROP TABLE tablo ismi; tabloyu geri getirilebilecek şekilde siler. RECYCLE BIN'e gönderir.

5 PURGE komutu ile tablo geri getirilemeyecek şekilde silinir. Ancak tabloyu direkt PURGE yapmak mümkün değildir. PURGE recycle bin'deki tablolar için çalışır. Bunun için önce tabloyu DROP TABLE ile siler, sonra PURGE ile geri getirilmeyecek şekilde sileriz.

---



# IN CONDITION

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

```
CREATE TABLE musteriler
(
urun_id number(10),
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');
```

```
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');
```

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

# BETWEEN CONDITION

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

```
CREATE TABLE musteriler  
(  
  urun_id number(10),  
  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 listeleyniz

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

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

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

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



# 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 number(10),  
  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 listeleyniz

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

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

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

2) İsminin ilk harfi E ile J arasında olmayan kişilerin tüm bilgilerini listeleyn



## Practice 13

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

- Yukarda verilen “personel” tablosunu olusturun
- Tablodan maasi 5000'den az veya unvani isci olanlarin isimlerini listeleyin
- Iscilerin tum bilgilerini listeleyin
- Soyadi Can,Cem veya Gul olanlarin unvanlarini ve maaslarini listeleyin
- Maasi 5000'den cok olanlarin emailve is baslama tarihlerini listeleyin
- Maasi 5000'den cok veya 7000'den az olanlarin tum bilgilerini listeleyin

TECHPROED

# Tekrar Sorulari

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

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

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

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

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

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

# Tekrar Sorularinin Cevaplari

- 1) DELETE ile TRUNCATE arasindaki fark nedir ?
  - A) TRUNCATE tum kayitlari siler, DELETE istersek tum kayitlari,istersek belirli kayitlari siler
  - B) DELETE ile sildigimiz datalari ROLLBACK yapabiliriz, TRUNCATE ile silinenler geri getirilemez
  - C) DELETE ile WHERE komutunu kullanabiliriz ama TRUNCATE ile kullanamayiz
- 2) DELETE ile DROP arasindaki fark nedir?  
DELETE kayitlari siler, DROP ise tab;olari.
- 3) DROP ile DROP PURGE arasindaki fark nedir?  
DROP ile sildigimiz dosyalar RECYLEBIN'e gider. PURGE RECYLEBIN'deki dosyalari geri getirilmeyecek sekilde siler. DROP PURGE beraber kullanilrsa geri getirilmeyecek sekilde silinir.

4)Asagidaki sorgu ile ayni sonucu veren bir sorgu yaziniz.

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

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

5) Asagidaki sorgu ile ayni sonucu veren bir sorgu yaziniz

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

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

6) Asagidaki sorgu ile ayni sonucu veren bir sorgu yaziniz

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

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



# KISA TEKRAR

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

```
CREATE TABLE personel
```

```
(  
  id number(9),  
  isim varchar(50),  
  sehir varchar(50),  
  maas number(20),  
  sirket varchar(20),  
  CONSTRAINT personel_pk PRIMARY KEY (id)  
);
```

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

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

```
CREATE TABLE personel_bilgi
```

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

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

# KISA TEKRAR

ID	ISIM	SEHIR	MAAS	SIRKET
123456789	Ali Yilmaz	Istanbul	5500	Honda
234567890	Veli Sahin	Istanbul	4500	Toyota
345678901	Mehmet Ozturk	Ankara	3500	Honda
456789012	Mehmet Ozturk	Izmir	6000	Ford
567890123	Mehmet Ozturk	Ankara	7000	Tofas
456789152	Veli Sahin	Ankara	4500	Ford
123456710	Hatice Sahin	Bursa	4500	Honda

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

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

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

# Tekrar Sorulari

**SORU 2)** Persone tablosundan ucreti 4500 veya 5000 olanlari maaslarini %10 artirin

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

**SORU 3)** Persone tablosundan maasi 4950 olanlari silin

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

```
ORA-02292: integrity constraint (SQL_AHZDXVGZXDBVBVLYOPIBABNDG.PERSONEL_BILGI_FK) violated - child record found  
ORA-06512: at  
"SYS.DBMS_SQL", line 1721
```



---

# Tekrar Sorulari

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

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

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

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

TECHNIPROED

---

---

# Tekrar Sorulari

**SORU 6)** Personel\_bilgi tablosundan datalari geri getirilemeyecek sekilde silin

```
TRUNCATE TABLE personel_bilgi;
```

**SORU 7)** Persone tablosundan maasi 4000 ile 5000 arasinda olanlari silin

```
DELETE FROM personel  
WHERE maas BETWEEN 4000 AND 5000;
```

TECHPROF

---

---

# Tekrar Sorulari

**SORU 8)** Personel tablosundan maasi 5000 ile 6000 arasinda olmayanlari silin

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

**SORU 9)** Persone tablosunu geri getirilemeyecek sekilde silin

```
DROP TABLE personel PURGE; HATA VERIR
```

Once personel\_bilgi tablosunu silin

```
DROP TABLE personel_bilgi;
```

Persone tablosunu geri getirilemeyecek sekilde silin

---



---

# SQL

Structured Query Language  
Yapılandırılmış Sorgu Dili

## SUBQUERY

TECHPROED

---

# SUBQUERIES

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

1) **WHERE** den sonra kullanilabilir

```
CREATE TABLE personel2
(
  id number(9),
  isim varchar(50),
  sehir varchar(50),
  maas number(20),
  sirket varchar(20)
);
```

```
INSERT INTO personel2 VALUES(123456789, 'Ali Seker', 'Istanbul', 2500, 'Honda');
INSERT INTO personel2 VALUES(234567890, 'Ayse Gul', 'Istanbul', 1500, 'Toyota');
INSERT INTO personel2 VALUES(345678901, 'Veli Yilmaz', 'Ankara', 3000, 'Honda');
INSERT INTO personel2 VALUES(456789012, 'Veli Yilmaz', 'Izmir', 1000, 'Ford');
INSERT INTO personel2 VALUES(567890123, 'Veli Yilmaz', 'Ankara', 7000, 'Hyundai');
INSERT INTO personel2 VALUES(456789012, 'Ayse Gul', 'Ankara', 1500, 'Ford');
INSERT INTO personel2 VALUES(123456710, 'Fatma Yasa', 'Bursa', 2500, 'Honda');
```

```
CREATE TABLE sirketler
(
  sirket_id number(9),
  sirket varchar(20),
  personel_sayisi number(20)
);
```

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

# SUBQUERIES

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

```
SELECT isim, sirket
FROM personel2
WHERE sirket IN ( SELECT sirket
                  FROM sirketler
                  WHERE personel_sayisi > 15000);
```

SIRKET
Ford
Toyota

Subquery sonucu

ISIM	SIRKET
Ayşe Gül	Toyota
Veli Yılmaz	Ford
Ayşe Gül	Ford

Query sonucu

2) Sirket\_id'si 101'den büyük olan sirketlerin verdikleri maaslari ve o maasi alanlari sehirlerini listeleyiniz

```
SELECT sehir, maas
FROM personel2
WHERE sirket IN (SELECT sirket
                 FROM sirketler
                 WHERE sirket_id > 101);
```

SIRKET
Hyundai
Toyota

Subquery sonucu

SEHIR	MAAS
Ankara	7000
Istanbul	1500

Query sonucu

SIRKET	PERSONEL_SAYISI
Honda	12000
Hyundai	10000
Ford	18000

3 -- Ankara'da personeli olan sirketlerin sirket id ve calisan sayilarini listeleyiniz



# SUBQUERIES

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

Ancak SELECT CLAUSE da kullanılan Subquery **SADECE 1 DEGER** donmelidir.

Dolayisiyla **SUM**, **COUNT**, **MIN**, **MAX** ve **AVG** gibi fonksiyonlar kullanilir.

Bu fonksiyonlara **AGGREGATE FUNCTION** denir.

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

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

FROM sirketler;

SIRKET	PERSONEL_SAYISI	(SELECTAVG(MAAS)FROMPERSONELWHERE SIRKETLER.SIRKET=PERSONEL.SIRKET)
Honda	12000	2666.6666666666666666666666666667
Ford	18000	1250
Hyundai	10000	7000
Toyota	21000	1500

# SUBQUERIES

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

```
SELECT şirket, personel_sayisi, (SELECT MAX(maaş)
                                FROM personel
                                WHERE şirketler.şirket=personel.şirket
                                ) AS şirketteki_Max_Maaş,

FROM şirketler;
```

SİRKET	PERSONEL_SAYISI	SİRKETTEKİ_MAX_MAAS
Honda	12000	3000
Ford	18000	1500
Hyundai	10000	7000
Toyota	21000	1500

# SUBQUERIES

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

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

SIRKET	PERSONEL_SAYISI	BULUNDUGU_SEHIR_SAYISI
Honda	12000	3
Ford	18000	2
Hyundai	10000	1
Toyota	21000	1



# SUBQUERIES

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

```
SELECT şirket, personel_sayisi, (SELECT MAX(maaş)
                                FROM personel
                                WHERE şirketler.şirket=personel.şirket
                                ) AS max_maaş,
    (SELECT MIN(maaş)
     FROM personel
     WHERE şirketler.şirket=personel.şirket
     ) AS min_maaş
FROM şirketler;
```

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

# SUBQUERIES

**SORU 5-** Her şirketin ismini, personel sayısını ve tablodaki işçilerine ödediği toplam maaşı listeleyen bir QUERY yazın.

```
SELECT şirket, personel_sayisi, (SELECT SUM(maaş)
                                FROM personel
                                WHERE şirketler.şirket=personel.şirket
                                ) AS toplam_maaş
FROM şirketler;
```

SİRKET	PERSONEL_SAYISI	TOPLAM_MAAS
Honda	12000	8000
Ford	18000	2500
Hyundai	10000	7000
Toyota	21000	1500

TECHPROED

---

# SQL

Structured Query Language  
Yapılandırılmış Sorgu Dili

EXISTS, IS NULL  
ORDER BY, ALIASES

TECHPROED

---



---

## Önceki Dersten Hatırladıklarımız

1 Subquery : iç içe query demektir. Birden fazla tabloyu ilgilendiren update'ler için kullanabiliriz. Update yapılacak tablodan kodumuzu yazmaya başlıyoruz, sonra diğer tablodan bilgi almak için subquery ekleyip SELECT komutu ile ilgili datayı alıyoruz

2 SELECT komutu ile de subquery kullanılabilir

- Eğer WHERE den sonra subquery kullanacaksak sorgunun sonucunu değerlendirmek gerekir.

- Eğer sorgu 1 sonuc getirecekse = isareti kullanabiliriz

- Ama sorgunun kaç sonuc getireceğini bilmiyorsak veya birden fazla sonuc getireceğini biliyorsak bu durumda = yerine IN kullanılır

- Eğer SELECT komutunun olduğu satırda SUBQUERY kullanacaksak sonucun 1 tek değer dondurmeleri gerekir, bunun için AVG,MIN,MAX,COUNT,SUM gibi AGGREGATE fonksiyonları kullanılır. Aggregate fonksiyonları araya , konulmak şartıyla istendigi kadar kullanılabilir

- Aggregate fonksiyonu yeni bir sütun oluşturduğunda sütun ismini atamak için AS istenimsim komutu kullanılır.

---

# 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_satislar  
(  
    urun_id number(10),  
    musteri_isim varchar(50),  
    urun_isim varchar(50)  
);
```

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

```
CREATE TABLE nisan_satislar  
(  
    urun_id number(10),  
    musteri_isim varchar(50),  
    urun_isim varchar(50)  
);
```

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



# EXISTS CONDITION

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

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

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

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

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



# EXISTS CONDITION

Her iki ayda da satılan urun\_isimleri aynı ürünlerin urun\_isim'ini ve ürünleri nisan ayında alanların isimlerini getiren bir query yazınız..

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

URUN_ISIM	MUSTERI_ISIM
Honda	Hasan
Honda	Kemal
Toyota	Ayşe
Toyota	Mine

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

URUN_ISIM	MUSTERI_ISIM
Volvo	Yasar

# IS NULL CONDITION

Arama yapılan field'da NULL degeri almıs kayıtları getirir.

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

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

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

SSN	NAME	ADDRESS
456789012	-	Bursa
567890123	-	Denizli

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

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

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

# ALIASES

Aliases kodu ile tablo yazdırılırken, field isimleri sadece o çıktı için değiştirilebilir

```
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');
```

EMPLOYEE_ID	EMPLOYEE_NAME	EMPLOYEE_BIRTH_CITY
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



# 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 number(9)
);
```

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

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

TECHPROFOD



# GROUP BY CLAUSE

```
CREATE TABLE personel
```

```
(  
  id number(9),  
  isim varchar(50),  
  sehir varchar(50),  
  maas number(20),  
  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 **GÖRE** 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

---

# SQL

Structured Query Language  
Yapılandırılmış Sorgu Dili

HAVING CLAUSE  
UNION, UNION ALL,  
INTERSECT, MINUS

TECHPROED

---

---

## Önceki Dersten Hatırladıklarımız

1 EXISTS : subquery'de EXISTS(istenenSart) şeklinde kullanılır, istenen sarta uyan değer(ler) için ilk sorgunun çalışmasını sağlar. IN komutunun kullanımına benzer ancak farklılıkları var

- IN komutundan önce bir field ismi yazmak zorundayız ancak EXISTS için field ismi yazılmaz
- IN komutundan önce hangi field ismi yazılmissa IN komutundan sonraki parantezin o field'a ait sonuçları getirmesi gerekir ancak EXISTS için böyle bir sınırlama yoktur
  - IN komutu için olumsuz durum kullanımı yoktur ama NOT EXISTS kullanılır. Bunun için sorgu olumlu duruma göre yazılır ancak ilk sorgu SARTI SAGLAMAYAN değerler için çalışır

2 IS NULL : WHERE satırında kullanılır ve sorgulama sonucunda istenilen field'daki boş olan kayıt bilgilerini getirir.

IS NOT NULL : boş olmayan kayıtları getirir

---



---

## Önceki Dersten Hatırladıklarımız

3ORDER BY CLAUSE : Sorgu sonucunun istedigimiz bir veya birden fazla fielda göre sıralı olarak gelmesini sağlar.

- Eğer sıralama turunu belirtmezsek default olarak NATURAL ORDER şeklinde sıralama yapar
- İstersek ASCENDING veya DESCENDING olarak sıralama yapabiliriz . Eğer birden fazla field'a göre sıralama yapacaksa önce ilk sıralamayı yapar,sonra ilk sıralamada aynı olan değerleri 2. sıralama kriterine göre sıralar

4ALIASES : Tablo yapısını degistirmeden sorguda gelen fieldlara farklı isimlendirme yapmamıza imkan tanır

- AS : AS kullandıktan sonra istedigimiz ismi yazabiliriz
- || : ile CONCATINATE yapabiliriz. || bir field'a sabit bir string ekleyebilir veya 2 field'ı birlestirebiliriz

5- GROUP BY: Sorgu sonuclarini belli field'lara göre gruplandirmamiza imkan tanir.

- GROUP BY kullandigimizda sorguda gruplandirma yaptigimiz field'in olmasina dikkat etmeliyiz
-

# HAVING CLAUSE

**HAVING**, AGGREGATE FUNCTION'lar ile birlikte kullanılan FİLTRELEME komutudur.

```
CREATE TABLE personel  
(  
  id number(9),  
  isim varchar(50),  
  sehir varchar(50),  
  maas number(20),  
  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 ise 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

TECHPROED

# UNION OPERATOR

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

```
CREATE TABLE personel
```

```
(  
  id number(9),  
  isim varchar(50),  
  sehir varchar(50),  
  maas number(20),  
  sirket varchar(20)  
  INSERT INTO personel VALUES(123456789, 'Ali Yilmaz', 'İstanbul', 5500, 'Honda');  
  INSERT INTO personel VALUES(234567890, 'Veli Sahin', 'İstanbul', 4500, 'Toyota');  
  INSERT INTO personel VALUES(345678901, 'Mehmet Ozturk', 'Ankara', 3500, 'Honda');  
  INSERT INTO personel VALUES(456789012, 'Mehmet Ozturk', 'İzmir', 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) Maası 4000'den çok olan işçi isimlerini ve 5000 liradan fazla maaş alan şehirleri gösteren sorguyu yazınız

```
SELECT sehir ASisci_veya_sehir_ismi ,maas
```

```
FROM personel
```

```
WHERE maas >5000
```

```
UNION
```

```
SELECT isim ASisci_veya_sehir_ismi , maas
```

```
FROM personel
```

```
WHERE maas > 4000;
```

ISCI_VEYA_SEHIR_ISMI	MAAS
Ali Yilmaz	5500
Ankara	4500
Ankara	7000
Bursa	4500
Hatice Sahin	4500
İstanbul	4500
İstanbul	5500
İzmir	6000
Mehmet Ozturk	6000
Mehmet Ozturk	7000
Veli Sahin	4500
Veli Sahin	4500

# UNION OPERATOR

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

```
SELECT sehir AS isci_veya_sehir_ismi ,maas
FROM personel
WHERE sehir='Istanbul'
UNION
SELECT isim AS isci_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

TECHFLOED

# 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 number(9),
  isim varchar(50),
  şehir varchar(50),
  maaş number(20),
  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 relation kurun

```
CREATE TABLE personel_bilgi
```

```
(
  id number(9),
  tel char(10) UNIQUE ,
  çocuk_sayısı number(2),
  CONSTRAINT personel_bilgi_fk FOREIGN KEY (id) REFERENCES personel(id)
);
```

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

# 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

TECHPROF

# 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

TECHPROED

---



# 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

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

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

# 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');
```

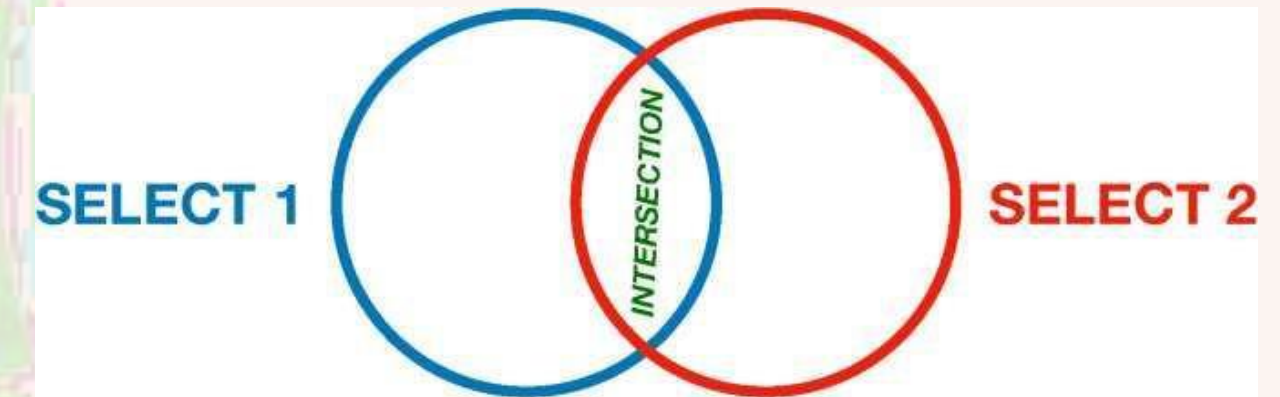
2) Personel\_bilgi tablosundan 2 veya 3 cocugu olanlarin id lerini yazdirin

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



TECHNORIDE

# 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;
```

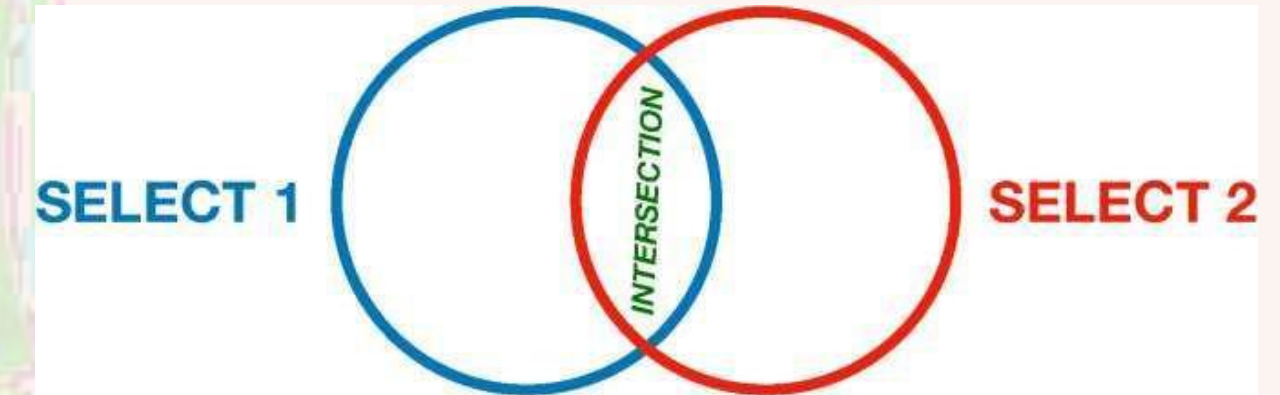
2) Personel\_bilgi tablosundan 2 veya 3 cocugu olanlarin id lerini yazdirin

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



TECHIR OED



# INTERSECT OPERATOR

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

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

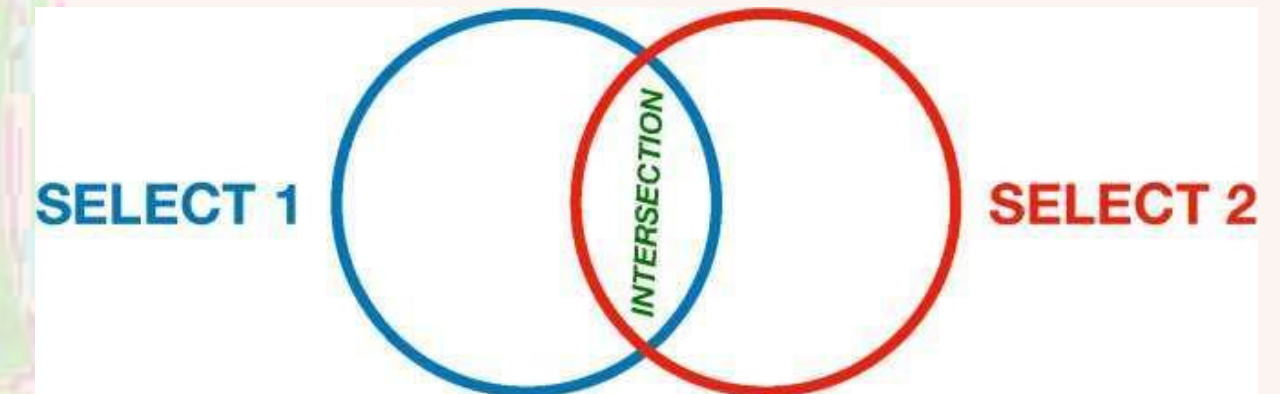
INTERSECT

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

INTERSECT

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

ISIM
Mehmet Ozturk



THE CHALLENGE

# MINUS OPERATOR

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

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

MINUS

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

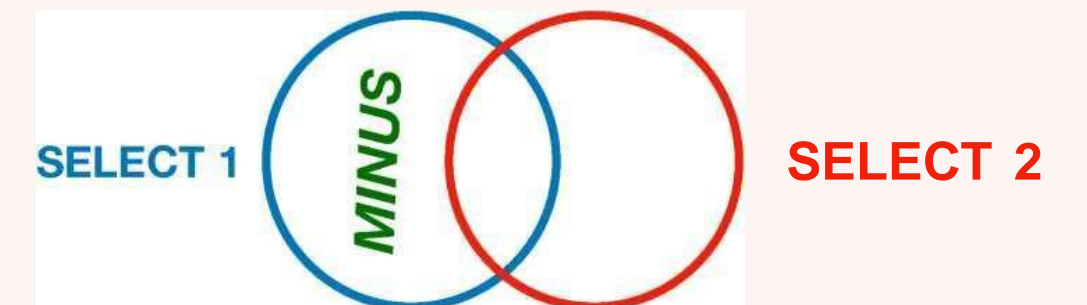
2) Ismi Mehmet Ozturk olup Istanbul'da calismayanlarin isimlerini ve sehirlerini listeleyin

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

MINUS

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

ISIM	SIRKET
Veli Sahin	Ford
Veli Sahin	Toyota



ISIM	SEHIR
Mehmet Ozturk	Ankara
Mehmet Ozturk	Izmir

---

# SQL

Structured Query Language  
Yapılandırılmış Sorgu Dili

JOINS,  
LIKE CONDITIONS

TECHPROED

---



---

## Önceki Dersten Hatırladıklarımız

- 1 **HAVING**, AGGREGATE FUNCTION'lar ile birlikte kullanılan FİLTRELEME komutudur.
- 2 **UNION** : İki farklı sorgulamanın sonucunu birleştiren işlemdir. İkinci sorgudan gelen sonuç ilk sorgudan gelmişse tekrar yazılmaz
  - Her 2 QUERY'den elde edeceğiniz tabloların sütun sayıları eşit olmalı
  - Alt alta gelecek sütunların data type'leri aynı olmalı
- 3 **UNION ALL**, UNION işlemi ile aynı işleve sahiptir , farklı ise tekrarlı elemanları, tekrar sayısını yazar.
- 4 **INTERSECT** : Her iki sorgu sonucunu karşılaştırır ve ortak olanları listeler
- 5 **MINUS** : Birinci sorguda olup, ikinci sorguda olmayan sonuçları listeler

TECHPROF

---

---

# JOINS

2 **T**ablodaki datalari Birleştirmek için kullanılır.

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

TECHPROF

---

# INNER JOINS

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

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

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



# INNER JOINS

TABLE 1

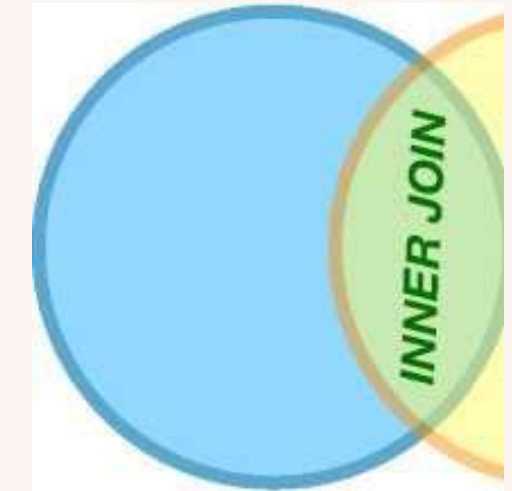


TABLE 2

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

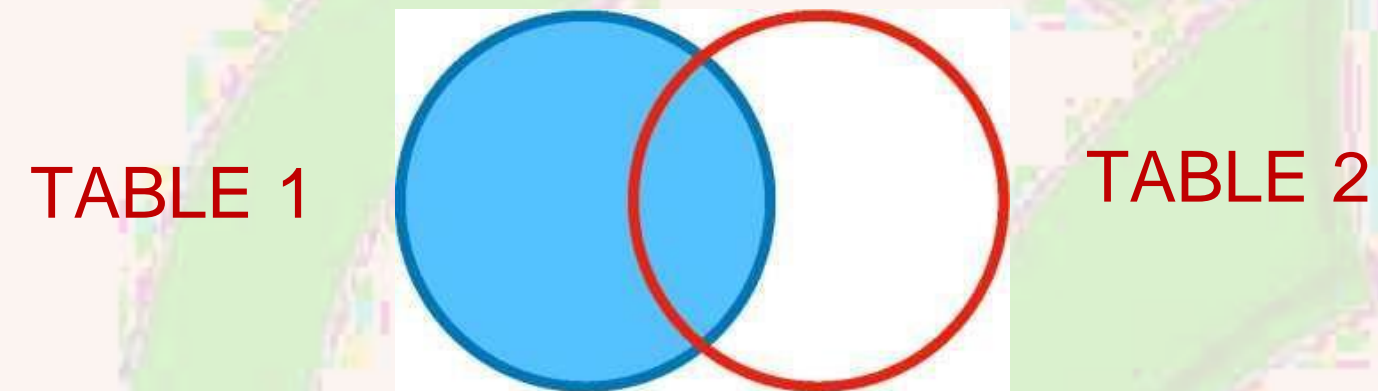
```
SELECT sirketler.sirket_isim, siparisler. siparis_id, siparisler. siparis_tarihi  
FROM sirketler INNER JOIN siparisler  
ON sirketler.sirket_id = siparisler.sirket_id;
```

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

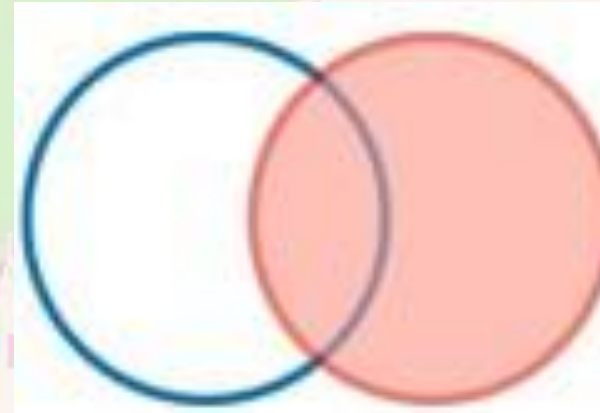


TABLE 2

```
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) İkinci 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	-	-

TECHPROFOD

# SELF JOINS

```
CREATE TABLE personel
```

```
(  
  id number(2),  
  isim varchar(20), title  
  varchar(60),  
  yonetici_id number(2)  
);
```

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

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 SELF 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 izin verir.. Ve bize pattern matching yapma imkani verir.

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

```
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
```

ID	ISIM	GELIR
1004	Fatma	42000

```
WHERE isim LIKE '_atma';
```

**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



---

# SQL

Structured Query Language  
Yapılandırılmış Sorgu Dili

REGEXP\_LIKE,  
OFFSET, FETCH NEXT  
PIVOT CLAUSES, ALTER

---

TECHPROED

---

## Önceki Dersten Hatırladıklarımız

1 JOIN : Bir tablonun kendisinden veya iki farklı tablodan istediğimiz field'leri kullanarak yeni tablo elde etmek için kullanılır.

- join işlemi yeni tablo oluşturmaz fakat yazdığımız sorgu sonuçlarını istediğimiz şekilde bir tabloda görmemizi sağlar
    - Daha önce gördüğümüz union,intersect,minus komutları tabloları birleştirmiyor, ayrı ayrı sorgular yapıp sorgu sonuçlarını birleştiriyordu. Hatta farklı tablolardan gelen ism ve şehir gibi değerleri tablo olarak yansıtamadığından alt alta yazmamız gerekiyordu.
    - INNER : her iki tabloda ortak olan recordlara ait her iki tablodaki bilgileri gösterir
    - LEFT JOIN: Birinci tablodaki tüm recordları ve ortak recordlara ait ikinci tablodaki bilgileri gösterir
    - RIGHT JOIN : İkinci tablodaki tüm recordları ve ortak recordlara ait birinci tablodaki bilgileri gösterir
    - FULL JOIN : Her iki tablodaki tüm record'ları gösterir, bir tabloda olup diğeri tabloda karşılığı olmayan bilgiler için null yazar
-

---

## Önceki Dersten Hatırladıklarımız

- SELF JOIN : Bir tablonun kendi içerisinde farklı bir kurala göre INNER JOIN yapılmasıdır.

Aynı tablo iki kere kullanılacağı için geçici olarak tabloya iki farklı isim verilir ve bu genelde tablo adının ilk harfi ile 1 ve 2 rakamlarının kullanılmasıyla olur (p1,p2 gibi)

FROM satırında tablo adı ile kısaltmalar birlikte yazılarak tanımlanır

**FROM personel p1 INNER JOIN personel p2**

ON komutundan sonra oluşturulan iki sanal tablo arasındaki kural tanımlanır

- LIKE : WHERE komutundan sonra kullanılır, wildcards kullanımina izin vererek PATTERN MATCHING (şekil benzetme) özelliğini kullanır
  - (%) : yazdığımız yerden sonra 0 veya daha fazla karakter olabilir
  - (\_) : sadece 1 tane karakteri gösterir

TECHPROF

---



# LIKE Condition

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

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

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

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

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

# LIKE Condition

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

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

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

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

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

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

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

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

ID	KELIME	HARF_SAYISI
1005	adem	4
1006	selim	5

# LIKE Condition

**SORU** : m veya f ile biten kelimelerin tüm bilgilerini yazdıran QUERY yazın

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

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

TECHPROED



# NOT LIKE Condition

**SORU 1** : ilk harfi h olmayan kelimelerin tum bilgilerini yazdiran QUERY yazin

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

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

**SORU 2** : a harfi icermeyen kelimelerin tum bilgilerini yazdiran QUERY yazin

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

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

# NOT LIKE Condition

**SORU 3** : ikinci ve ucuncu harfi 'de' olmayan kelimelerin tum bilgilerini yazdiran QUERY yazin

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

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

**SORU 4** : 2. harfi e,i veya o olmayan kelimelerin tum bilgilerini yazdiran QUERY yazin

```
SELECT *  
FROM kelimeler  
WHERE NOT REGEXP_LIKE (kelime, '[_eio]');
```

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

# UPPER – LOWER - INITCAP

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

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 musteri_urun;
```

URUN_ISIM
Elma
Portakal
Kaysi
Armut

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

MUSTERI_ISIM
Veli
Ayşe
Elif
Adem
Ali

Tabloda kaç farklı meyve vardır ?

```
SELECT COUNT(DISTINCT urun_isim) AS urun_cesit_sayisi  
FROM musteri_urun;
```

URUN_CESIT_SAYISI
4

TECHPR

# 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

# PIVOT CLAUSES

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

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

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

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

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

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



# ALTER TABLE STATEMENT

**ALTER TABLE** statement tabloda **add**, **modify**, veya **drop/delete columns** islemleri için kullanılır.

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

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

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

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

# ALTER TABLE STATEMENT

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

**ALTER TABLE** personel  
**ADD** ulke\_isim **varchar**(20) **DEFAULT** 'Turkiye';

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

2) Tabloya birden fazla sutun ekleme

**ALTER TABLE** personel  
**ADD** (cinsiyet **varchar**(20) , yas **number**(3));

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;

ISCILER
<b>Table</b> Status: Valid Created 17 minutes ago

## 6) MODIFY sutunlarin ozelliklerini degistirme

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

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

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

---

# SQL

Structured Query Language  
Yapılandırılmış Sorgu Dili

Genel Tekrar, Interview  
Sorulari ile Genel Tekrar

TECHPROED

---

---

## Önceki Dersten Hatırladıklarımız

- 1 REGEXP LIKE : [] içine yazılan kısaltmalarla bir karakterin neye benzediğini temsil eder
  - 2 UPPER(fieldismi) : yazılan field'daki tüm dataları büyük harfle yazılı olarak listeler  
LOWER(fieldismi) : yazılan field'daki tüm dataları küçük harfle yazar  
INITCAP(fieldismi) : yazılan field'daki dataların ilk harfi büyük, diğer harfler küçük olarak yazar
  - 3 DISTINCT(fieldismi): yazılan field'daki değerleri tekrarsız olarak listeler, Ürün sayısı,müşteri sayısı gibi sorgularda kullanılır
  - 4 FETCH NEXT 3 ROW : ilk 3 satırı(recordu) listeler  
OFFSET 5 ROW : 5 satır atlar
  - 5 PIVOT : bir tablodan seçtiğimiz 2 field'daki değerleri karşılaştırarak bir pivot tablosu oluşturur
  - 6 ALTER TABLE STATEMENTS
    - ADD : Tabloya bir field ekler, eğer eklenen field'a default olarak bir değer yazılmasını istiyorsak ADD komutundan sonra DEFAULT(istenendeğer) kodu kullanılır
      - MODIFY : Tabloda varolan bir field'ın özelliklerini değiştirir
      - DROP COLUMN istediğimiz field'ı tablodan siler
      - RENAME .... TO .... ; Tablo ismini değiştirir
      - RENAME COLUMN .... TO ... : field ismini değiştirir
-



# INTERVIEW QUESTION

CREATE TABLE personel

```
(  
  id number(9),  
  isim varchar(50),  
  sehir varchar(50),  
  maas number(20),  
  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 number(9),  
  isim varchar(50),  
  sehir varchar(50),  
  maas number(20),  
  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 kayıt olduğunu gösteren query yazın

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

COUNT(*)
6

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

KAYIT_SAYISI
6

6) Isciler tablosunda en yüksek maası alan kişinin tüm bilgilerini gösteren query yazın

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

TECHNIPROED

# 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

TECHPROF