



BATCH :  
LESSON :  
DATE :  
SUBJECT :

146 - 149

SQL

24.07.2023

SQL - 1

ZOOM GİRİŞLERİNZİ LÜTFEN **LMS** SİSTEMİ ÜZERİNDEN YAPINIZ





# PostgreSQL

1. Ders

24.07.2023

B149 AWS & DevOps

B146 Cyber Security

# Bugün ne yapıyoruz?

- Veritabanı Temel Kavramlar
- Veri Türleri ve Sınırlamaları
- PostgreSQL' de DB ve Tablo Oluşturma



Postgre**SQL**

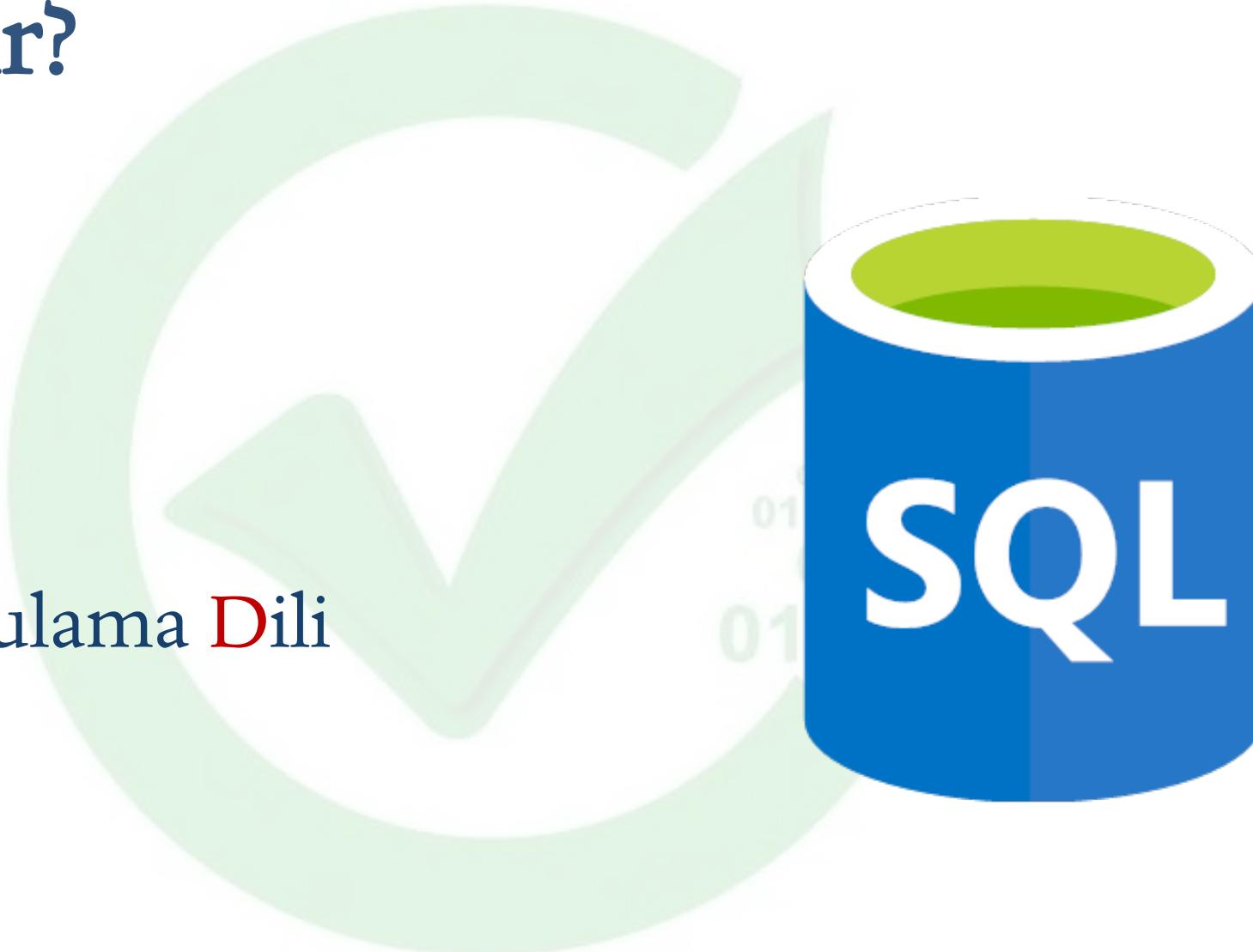
# SQL NEDİR?

Structred

Query

Language

- Yapısal Sorgulama Dili



# Veritabanı (Database)

**Kaydol**  
Hızlı ve kolaydır.

Adın  Soyadın

Cep telefonu numarası veya e-posta

Yeni şifre

Doğum Tarihi  1 Eki 2020

Cinsiyet  Kadın  Erkek  Özel

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

**Kaydol**

- **Data:** İşlenmemiş ham bilgilerdir. Database' ler belirli bir kurala göre bir araya getirilmiş verilerin kalıcı olarak (**persist**) depolanmasını sağlayan sistemlerdir.

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



# Database

- 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 (**Structured Query Language**) kullanılır.

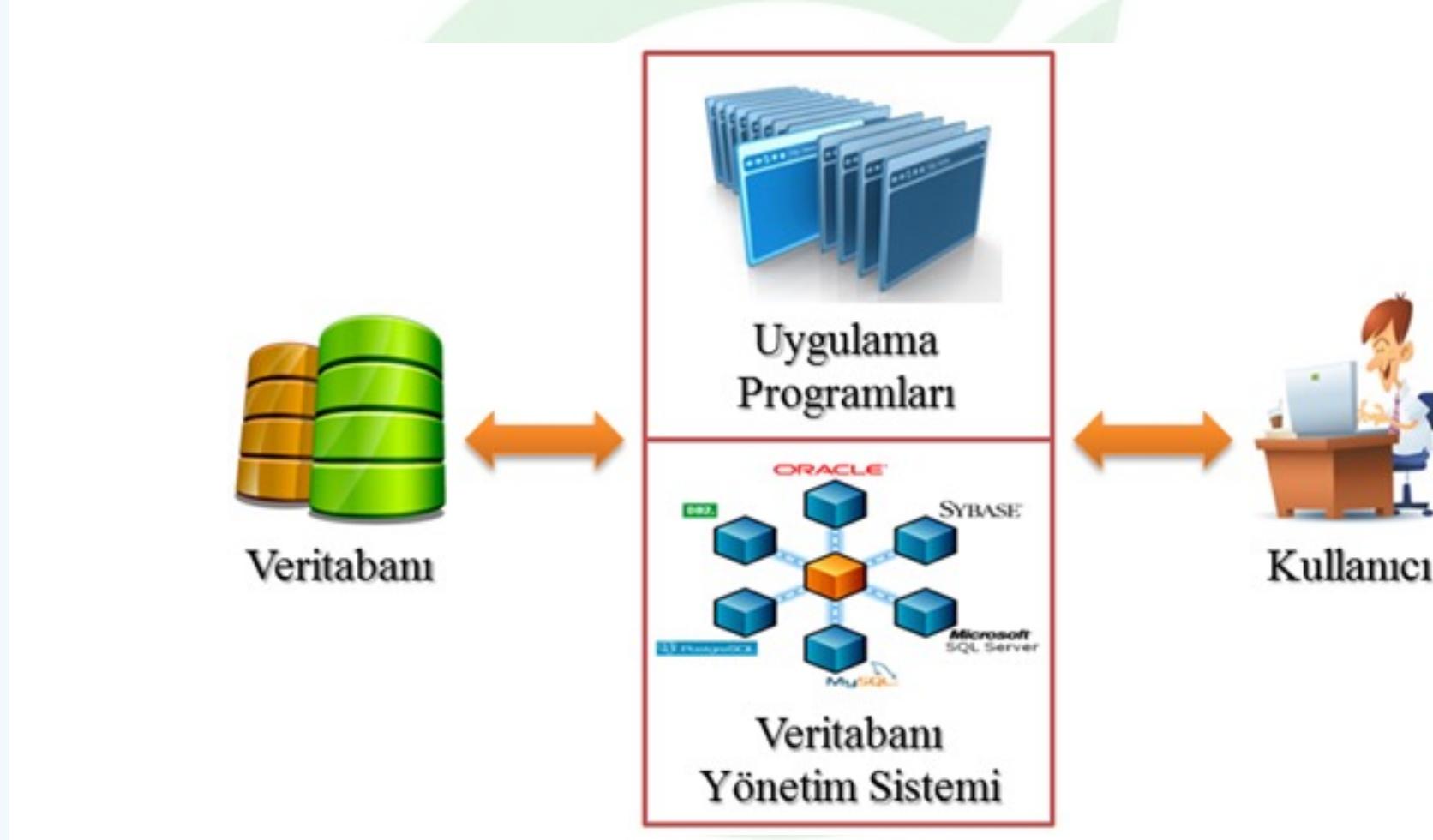


# Database Faydaları Nelerdir?

- Yüksek miktarda bilgi depolanabilir
  - Oluşturma, Okuma, Değiştirme ve Silme kolaylığı**Create, Read, Update, Delete (CRUD)**
- Veri girişin kolay ve kontrollü olması
- Dataya ulaşım kolaylığı
- Güvenlik

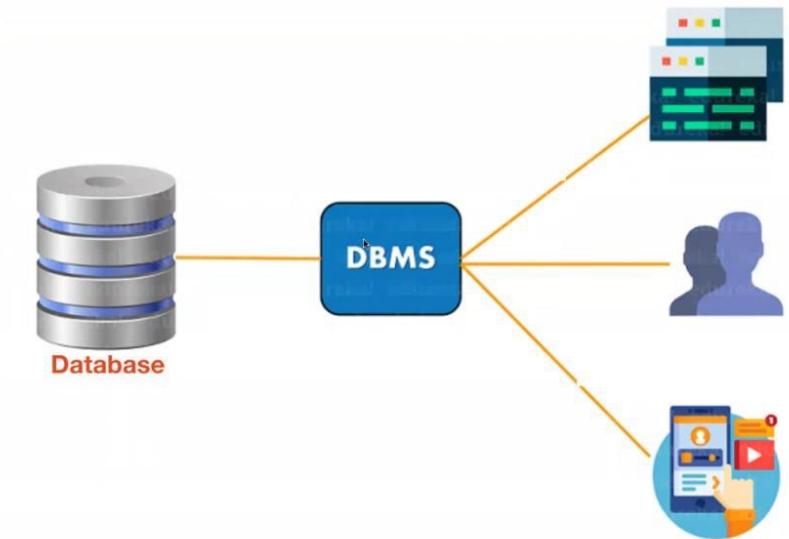


# Veritabanı Yönetim Sistemi



# DataBase Management System (DBMS)

- Veritabanlarını yönetmek, kullanmak, geliştirmek ve bakımını yapmak için kullanılan yazılımlardır.
- Database' e erişimi düzenler.
- Create, Read, Update, Delete işlemlerini düzenler.
- Data güvenliğini sağlar.
- Formlar oluşturur ve formları işler.



# Tablo (Table)

Headers====>

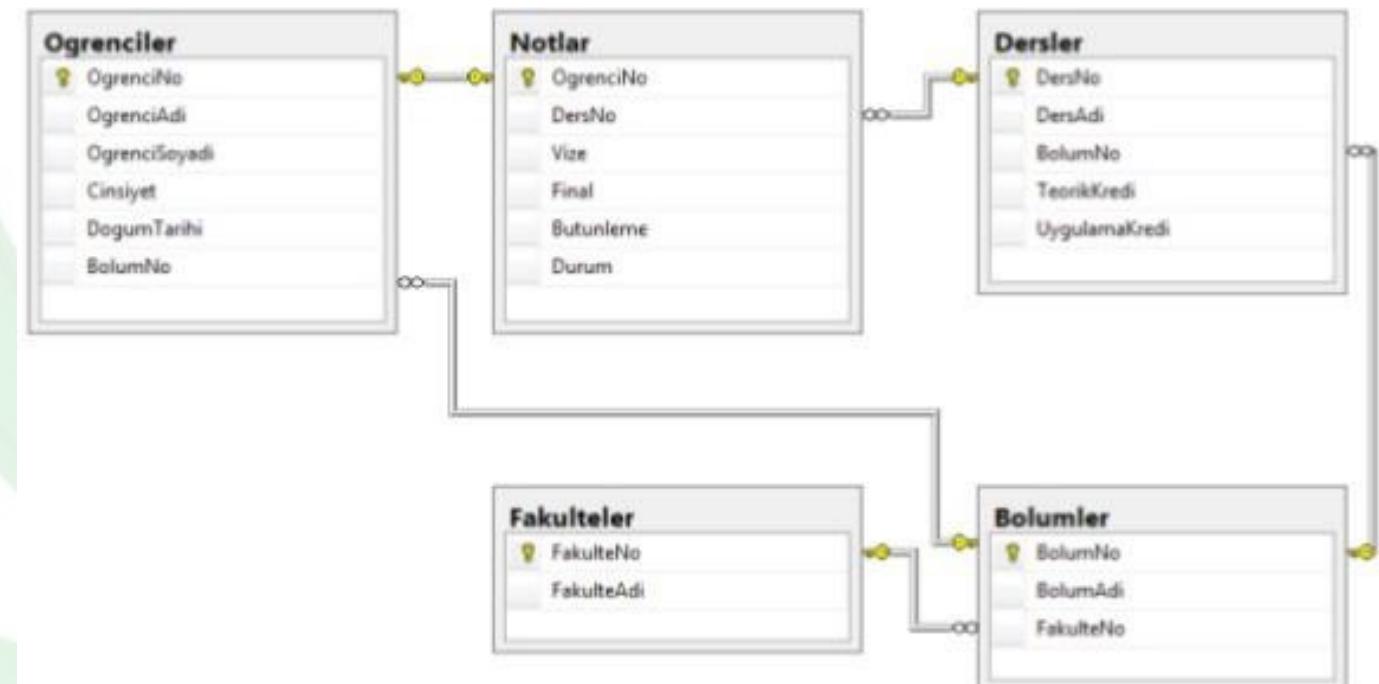
contactID	name	company	email
1	Bill Gates	Microsoft	bill@XBoxOneRocks.com
2	Steve Jobs	Apple	steve@rememberNewton.com
3	Linus Torvalds	Linux Foundation	linus@gnuWho.org
4	Andy Harris	Wiley Press	andy@aharrisBooks.net

Row (Record) =====>

Column (Field) =  
=

# İlişkisel DB Yönetim Sistemi (RDBMS)

- Tekrar eden verileri amacıyla veritabanı yapılandırılan sistemleridir.
- **Örneğin;** sürekli gelen müşterilerin bilgilerinin kayıt altına alınması
  - İş gücü kaybı
  - Kullanıcı hataları sorunu



# İlişkisel DB Yönetim Sistemi (RDBMS)

- SQL tablolar dataları ilişkili tablolarda depolar.
- Tablolar arası ilişkiler net olmalıdır.
- Tablolar arası geçiş kolay olmalıdır.
- Tabloların ve ilişkilerin bütününe **SCHEMA** denir.

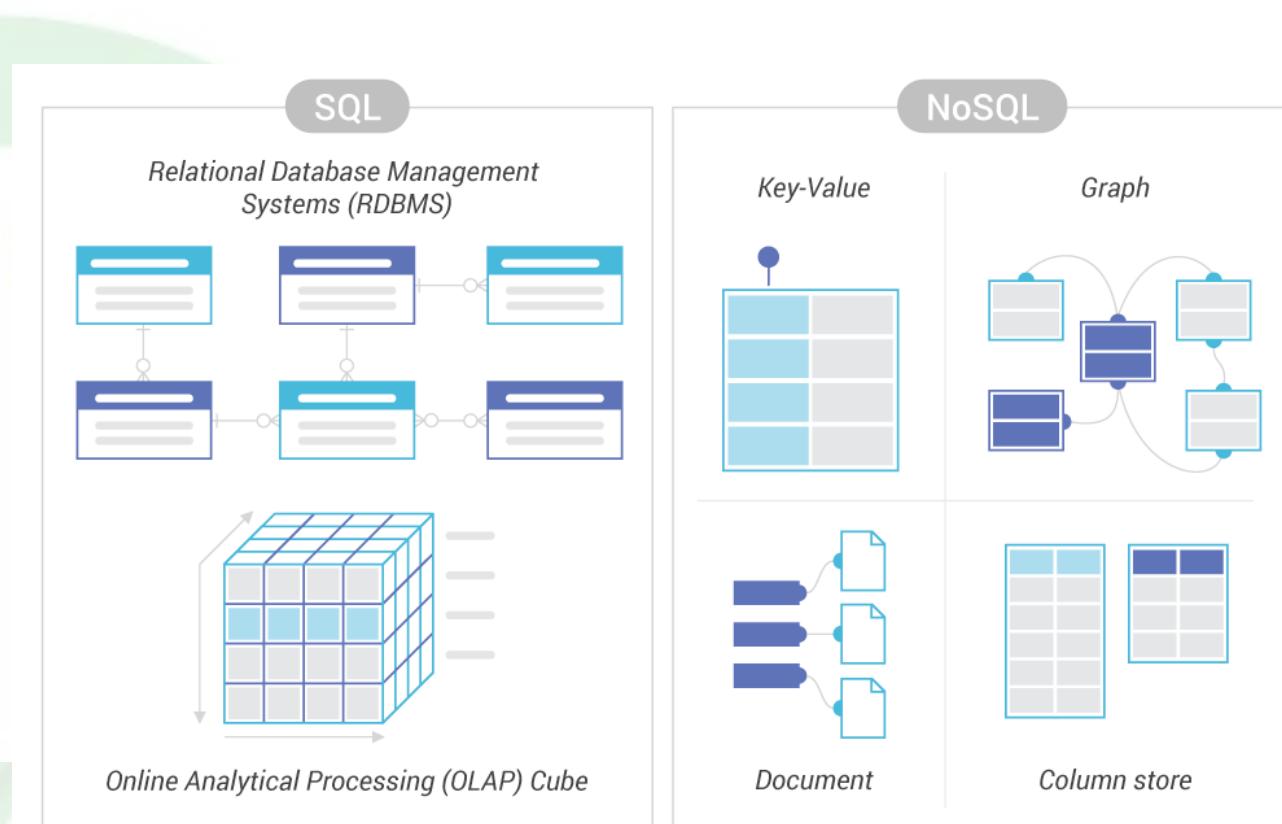
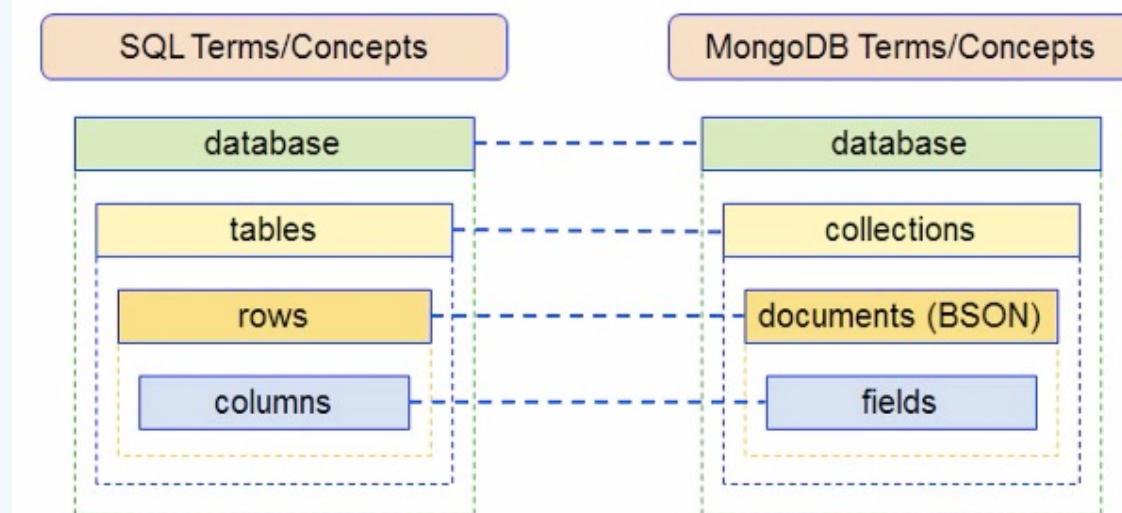
id	ogrenci_adi	ogrenci_soyadi
1	Elif	Türkmen
2	Ayşe	Sarı
3	Ender	Kaya
4	Ali	Demir
5	Adem	Salih

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.

# SQL vs NoSQL



- SQL veritabanları yapılandırılmış veri modeli ve şema tabanlı yaklaşım sunarken,
- NoSQL veritabanları daha esnek veri modelleri ve şema özgürlüğü sunarak, büyük ölçekli ve dağıtık uygulamalara uygun alternatifler sunar.
- Veritabanı seçimi, uygulamanın gereksinimlerine, veri yapısına ve ölçeklenebilirlik ihtiyacına bağlı olarak belirlenir.

# Summary

# SQL Komutları

- Veri Sorulama 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.
- Veri Kullanma Dili (Data Manipulation Language - **DML**)
  - DML komutları ile veritabanlarında bulunan verilere işlem yapılır. DML ile veritabanına yeni kayıt ekleme, mevcut kayıtları güncelleme ve silme işlemleri yapılır.
  - **INSERT** : Veritabanına yeni veri ekler.
  - **UPDATE** : Veritabanındaki verileri günceller.
  - **DELETE** : Veritabanındaki verileri siler.

# SQL Komutları

- 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.
- Veri Kontrol Dili (Data Control Language - **DCL**)
  - DCL komutları ile kullanıcılara veritabanı ve tablolardan 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**).
- Satırlara ait değerlerin karışmaması adına bu alana ait bilginin tekrarlanması gereklidir.
- Çoğunlukla tek bir alan (id, user\_id, e\_mail, username, national\_identification\_number vb.) olarak kullanılsa da birden fazla alanın birleşimiyle de oluşturulabilir.

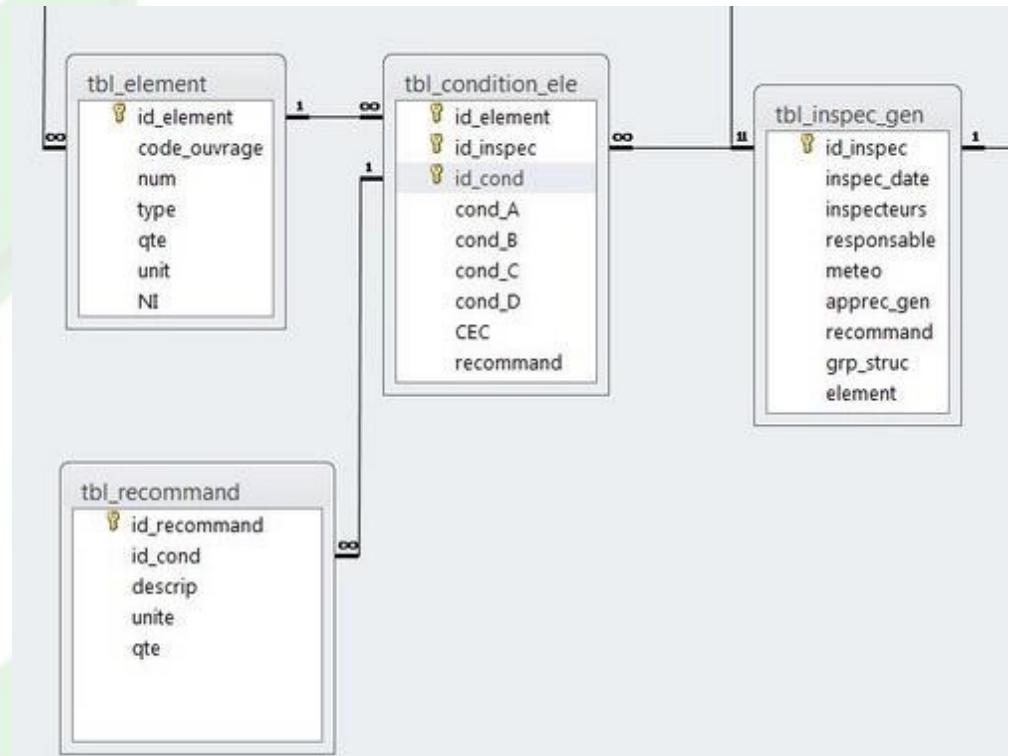
Primary Keys



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

# Primary Key

- 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 turlu datayı içerebilir. Sayı, String.
- Not : Her tabloda Primary Key olması **zorunlu değildir**.



# Foreign Key

- Foreign Key iki tablo arasında relation olusturmak icin kullanılır
- Foreign Key baska bir tablodaki Primary Key ile ilişkilendirilmiş olmalıdır.

Child Table

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

Parent Table

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

Bir Tabloda birden fazla Foreign Key olabilir Foreign Key NULL degeri kabul eder. Foreign Key olarak tanımlanan field' da tekrarlar olabilir.

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

# Foreign Key

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

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

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

# One to One Relation

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

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

- Tom Hanks'in adresi nedir?
- John Walker'in eyaleti nedir?
- ID'si 15 olan kisinin sehri nedir?

# 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

- Biology dersi alan öğrenciler kimler?
- Selective ders alan öğrencilerin isimleri ?
- CourseFee 600 olan öğrencilerin isimleri ?

# 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

- Öğretmeni Mark Adam olan öğrencilerin isimleri nedir?
- Kevin Star'in öğretmenlerinin isimleri nedir?
- Pamela Star'in öğretmenlerinin isimleri nedir?

# SQL Data Types (String)

Data Type	Acıklama
<b>char(size)</b>	<p>Maximum boyutu <b>2000 byte</b> olur.</p> <p><b>1 karakter 1 byte</b> kullanır. “<b>size</b>” database’<b>e eklenecek karakter sayisidir.</b></p> <p>“char” data tipinden <b>uzunlugu sabit</b> dataları depolar. (Strings)</p> <p>“char” SSN, zip kodu gibi uzunluğu sabit dataları depolamak için idealdir.</p>
<b>nchar(size)</b>	<p>Maximum boyutu <b>2000 byte</b> olur.</p> <p><b>1 karakter 2 byte</b> kullanır</p> <p>“<b>size</b>” depolanacak <b>karakter sayısı</b>’dir.</p> <p>“<b>nchar</b>”</p> <p><b>Unicode</b></p> <p>dataları</p> <p>depolamak için</p> <p>kullanılır.</p> <p>Genellikle <b>farklı dillerdeki karakterler</b> için</p> <p>kullanılır</p> <p><b>Uzunluğu belli Stringler</b> için</p> <p>kullanılır.</p>
<b>varchar2(size)</b>	<p>Maximum boyutu <b>4000 byte</b> olur.</p> <p><b>1 karakter 1 byte</b> kullanır.</p> <p>“<b>size</b>” database’<b>e eklenecek max.</b> karakter sayisidir.</p> <p><b>Degisken uzunluktaki stringler</b> için kullanılır.</p>
<b>nvarchar2(size)</b>	<p>Maximum boyutu <b>8000 byte</b> olur.</p> <p><b>1 karakter 2 byte</b> kullanır</p> <p>“<b>size</b>” depolanacak <b>karakter sayısı</b>’dir.</p> <p><b>Degisken uzunluktaki stringlerin Unicode degerleri</b> için kullanılır.</p>

# SQL Data Types (Numeric)

Data Type	Acıklama
number(p, s)	<p>“Precision” (p) sayidaki rakam sayisidir “Scale” (s) virgulden sonar kac rakam oldugunu belirler Ornegin: <b>1234,56 ==&gt; Precision : 6, Scale : 2.</b></p> <p>Precision ( p ) can range from <b>1 to 38</b> Scale ( s ) can range from <b>-84 to 127</b></p> <p>1) “number(<b>5, 2</b>)” virgulden once 3,vigulden sonra 2 rakam olan sayi ==&gt; <b>123,45</b></p> <p>2) “number(<b>4, 2</b>) ==&gt; <b>123,45 ==&gt; error verir</b></p> <p>3) “number(<b>7</b>)” ondalik kismi olmayan 7 basamakli sayı demektir ==&gt; <b>12345,67’i kabul eder ama 12345 olarak depolar</b> <b>Note:</b> “number(<b>7</b>)” ve “number(<b>7, 0</b>)” ayni seydir</p> <p>4) “number(<b>7, -2</b>)” rounds the numeric value to hundreds. ==&gt; <b>1234567,89 ==&gt; 1234600</b></p>

# SQL Data Types (Numeric)

## 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**)

<b>Data Type</b>	<b>Açıklama</b>
<b>DATE</b>	<p>“<b>DATE</b>” data tipi tarih ve zamani depolamak için kullanılır. Saniyenin virgullu kısmını da alır. “<b>DATE</b>” yıl, ay, gün, saat, dakika, ve saniye içerir.</p> <p>Standart “<b>Date Format</b>”, “<b>dd - MMM - yy</b>”. Örneğin 13 - Apr - 20</p> <p>Tarih formatını “<b>ALTER SESSION SET NLS_DATE_FORMAT = “YYYY-MM-DD”</b> kodu kullanılarak değiştirebilir. Koddan sonra tarih 2020 - 04 – 13 olur.</p>

# Summary

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

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

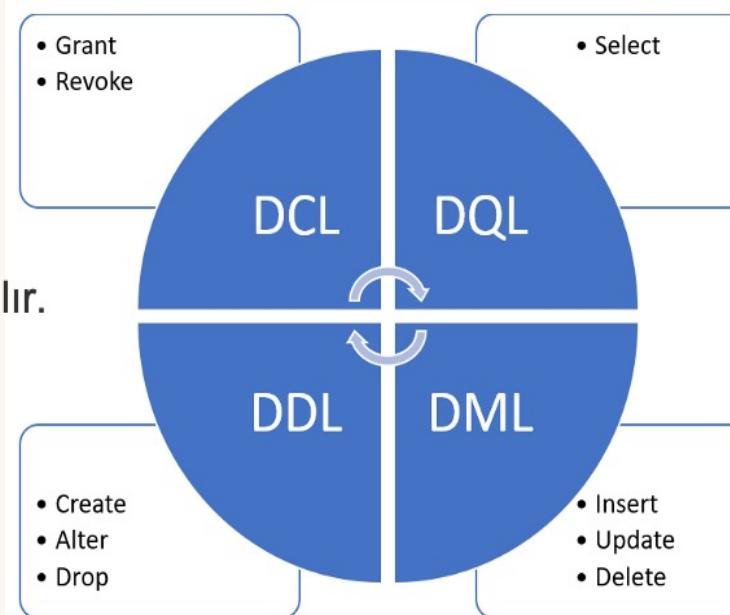
**REVOKE** : Bir kullanıcıya verilen  
yetkiyi geri almak için kullanılır.

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)

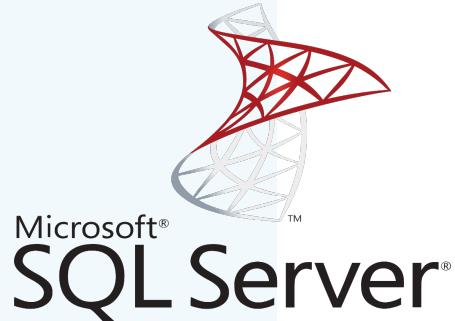
veritabanına yeni kayıt ekleme, mevcut kayıtları güncelleme ve silme işlemleri yapılır.

**INSERT** : Veritabanına yeni veri ekler.

**UPDATE** : Veritabanındaki verileri günceller.

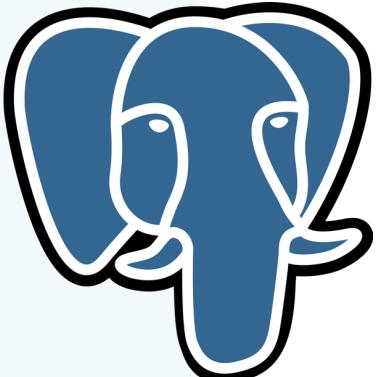
**DELETE** : Veritabanındaki verileri siler.

# En Popüler RDBMS



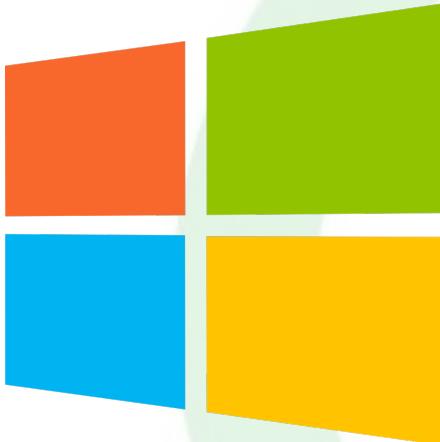
- **SQL Server**: Microsoft tarafından geliştirilmiştir.
- **Negatif**: Lisans Ücreti – Kurumsal Kullanıcılar için binlerce dolar ödenmesi gereklidir.
- **Pozitif**: Zengin bir user interface' e sahip ve çok büyük dataların kullanılmasında sorunsuz çalışır.
  
- **MySQL Server**: İsveçli MySQL firması tarafından geliştirildi. 2010'da Oracle satın aldı
- **Negatif**: Eszamanlı çok fazla işlem girildiğinde çalışmayı durdurabilir.
- **Pozitif**: Açık kaynak. Online destek ve ücretsiz çok fazla dokuman var.

# En Popüler RDBMS



- **PostgreSQL Server :** Created by a computer science professor Michael Stonebraker.
  - **Negatif:** İleri seviye yönetimi diğerlerine göre zayıf.
  - **Pozitif:** Yeni nesil olarak ortaya çıktı. Kisiselleştirme mumkundur, zor görevler için ideal olabilir.
- 
- **PL/SQL** Oracle database sunucuları içinde depolanır
  - **PL/SQL** SQL komutlarını özellikle karşılamak üzere dizayn edilmiştir.
  - **Pros:** PL/SQL yüksek güvenlik seviyesi sağlar ve Object-Oriented Programming'e uyumludur. Kompleks ve yüksek miktarda veri içeren yapılar için en çok tercih edilen Relational DB' dir.

# PostgreSQL Kurulum



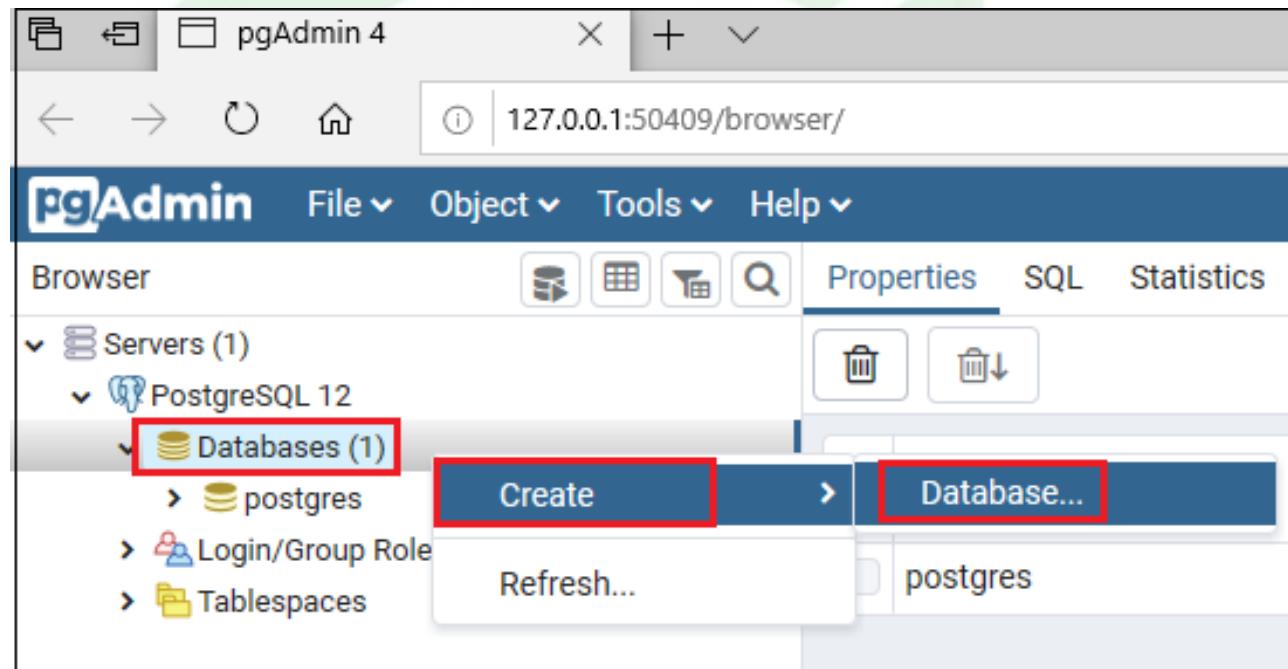
**Kurulum Linki**

<https://www.postgresql.org/download/>

# Veritabanı İşlemleri

-- 1- Yorum Satırı

# Veritabanı İşlemleri



-- 2- Database Oluşturma - DDL

CREATE DATABASE ms;

-- SQL komutlarında case sensitive yoktur ancak değer atamalarında vardır.

# Veritabanı İşlemleri

-- 3- Database Silme – DDL

```
DROP DATABASE ms;
```

-- Silme işleminin yapılabilmesi için Database' den Disconnect yapılması gerekmektedir.

# Veritabanı İşlemleri

## -- 4- Tablo Oluşturma – DDL

```
CREATE TABLE students(  
    id CHAR(4),  
    name VARCHAR(50),  
    grade REAL,  
    register_date DATE  
);
```

-- ; kullanmak zorunda değiliz. Birden fazla komutu çalıştırınca sorgunun bittiği  
-- yeri belirtmek için kullanılır. Daha readable bir yapı kavuşturmak istediğiniz zamanda bunu kullanabilirsiniz.

# Veritabanı İşlemleri

- 5- Var olan bir tablodan yeni bir tablo oluşturma

```
CREATE TABLE grades as SELECT name, grade FROM students;
```

- Sadece isim ve notları tutan bir tablo yapıldı.

# Veritabanı İşlemleri

-- 6- Tablo içine veri ekleme – DML

```
INSERT INTO students VALUES ('1000','Mehmet', 92.8, '2021-01-01');  
INSERT INTO students VALUES ('1001','Sungur', 81, '2022-10-08');  
INSERT INTO students VALUES ('1002','Kerem', 100, now());
```

-- Komutları çok seçenekde Execute edebilirsiniz. Bunun için ; kullanılması gerekmektedir.

# Veritabanı İşlemleri

-- 7- Tablonın bazı sütunlarına veri ekleme – DML

```
INSERT INTO students(name,grade) VALUES ('Aslı', 87);
```

# Veritabanı İşlemleri

-- 8- Tablonun tüm kayıtlarını getirme

```
SELECT * FROM students;
```

# Veritabanı İşlemleri

-- 9- Tablonun bazı kayıtlarını getirme

```
SELECT name,grade FROM students;
```

# Veritabanı İşlemleri

/\*----- ÖDEV -----

1) Tablo Oluşturma

"tedarikciler" isminde bir tablo olusturun,

"tedarikci\_id", "tedarikci\_ismi", "tedarikci\_adres"

ve "ulasim\_tarihi" field'lari olsun.

2) Var olan tablodan yeni tablo olusturmak

"tedarikci\_info" isminde bir tabloyu "tedarikciler" tablosundan olusturun.

Icinde "tedarikci\_ismi", "ulasim\_tarihi" field'lari olsun

3) Data ekleme

"ogretmenler" isminde tablo olusturun.

Icinde "kimlik\_no", "isim", "brans" ve "cinsiyet" field'lari olsun.

"ogretmenler" tablosuna bilgileri asagidaki gibi olan kisileri ekleyin.

kimlik\_no: 234431223, isim: Nana TechWorld, brans : DevOps, cinsiyet: kadin.

kimlik\_no: 234431224, isim: Mehmet Ince, brans : Security, cinsiyet: erkek.

4) Bazı fieldları olan kayıt ekleme

"ogretmenler" tablosuna bilgileri asagidaki gibi olan bir kisi ekleyin.

kimlik\_no: 567597624, isim: Stephane Maarek

\*/





BATCH : 146 - 149  
LESSON : SQL  
DATE : 25.07.2023  
SUBJECT : SQL - 2

ZOOM GİRİŞLERİNİZİ LÜTFEN **LMS** SİSTEMİ ÜZERİNDEN YAPINIZ





# PostgreSQL

2. Ders

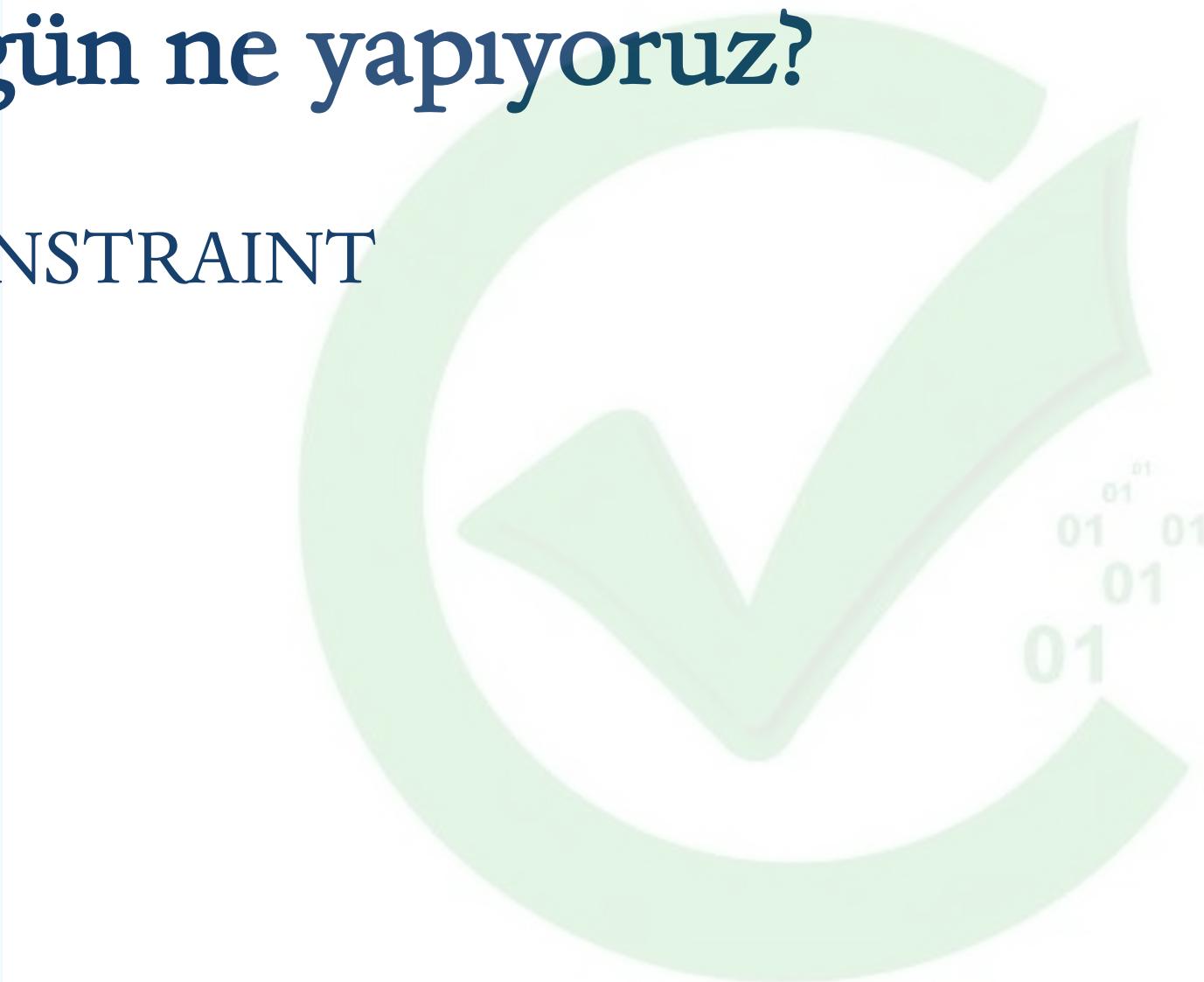
25.07.2023

B149 AWS & DevOps

B146 Cyber Security

# Bugün ne yapıyoruz?

- ◆ CONSTRAINT



# Constraint

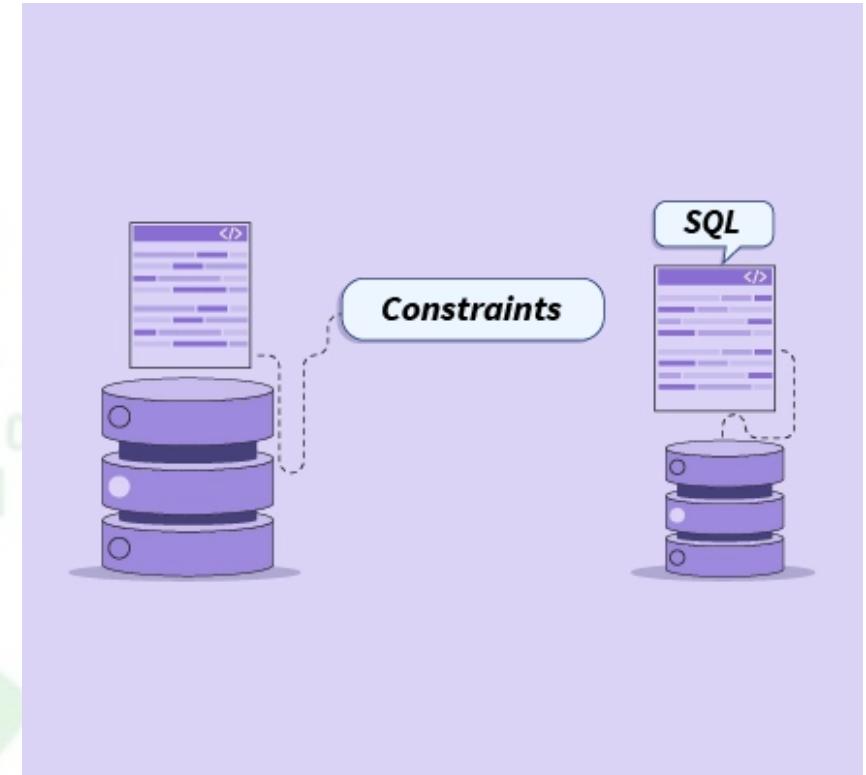
**UNIQUE:** Bir sütundaki tüm değerlerin Benzersiz/Tekrarsız yani tek olmasını sağlar.

**NOT NULL:** Bir sütunun NULL içermemesini sağlar. NOT NULL kısıtlaması için constraint ismi tanımlanmaz. Bu kısıtlama veri türünden hemen sonra yerleştirilir.

**PRIMARY KEY:** Bir sütunun NULL içermemesini ve sütundaki verilerin BENZERSİZ olmasını sağlar. (NOT NULL e UNIQUE)

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

**CHECK:** Bir sütuna yerleştirilecek değer aralığını sınırlamak için kullanılır.



# UNIQUE

Bir sütunun «tekrarsız» yapmak için, sütunun Data Type' dan sonra «**UNIQUE**» yazılır.

```
CREATE TABLE developers(  
    id SERIAL,  
    name VARCHAR(50),  
    email VARCHAR(50) UNIQUE,  
    salary REAL,  
    prog_lang VARCHAR(20)  
);
```



# NOT NULL

Bir sütunun «boş bırakmamak» için, sütunun Data Type' dan sonra «NOT NULL» yazılır.

Önemli Not: Sütun değerinin "" (empty) ile NULL olması **aynı şey değildir**.

**CREATE TABLE** developers(

```
    id SERIAL,  
    name VARCHAR(50) NOT NULL,
```

```
    email VARCHAR(50),
```

```
    salary REAL,
```

```
    prog_lang VARCHAR(20)
```

);

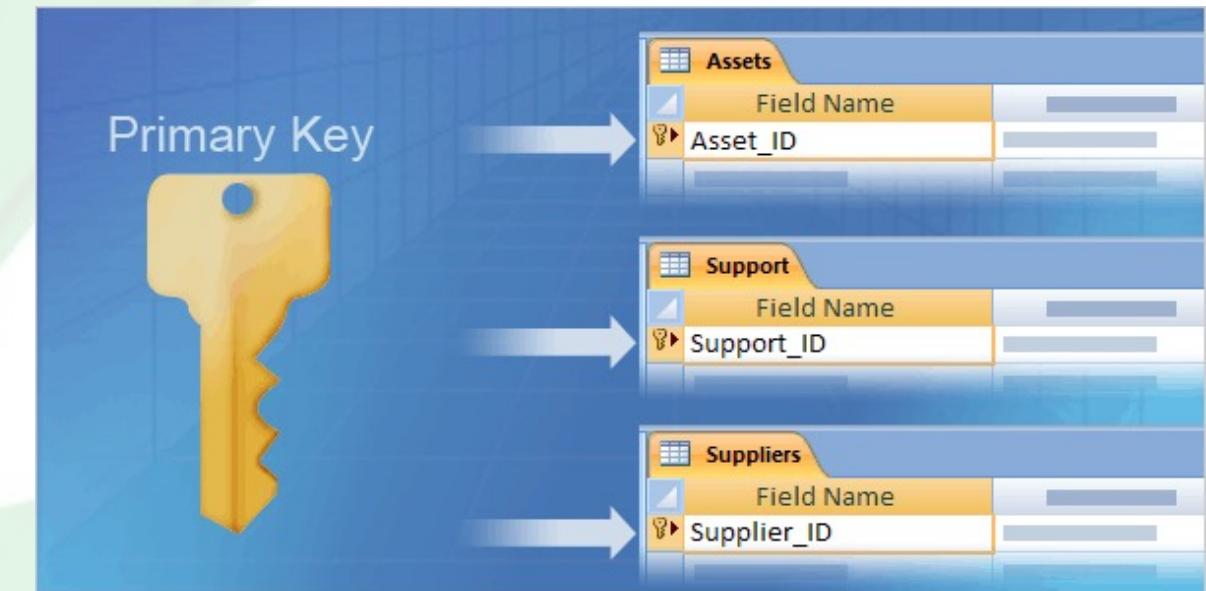
This is not a NULL Value

StudentID	StudentName	City	marks
1	Raj		45
2	Sham	Mumbai	NULL
3	Tom	Pune	66
4	Ram	Pune	58
5	Joy	Mumbai	NULL
6	Robin	NULL	NULL

# PRIMARY KEY

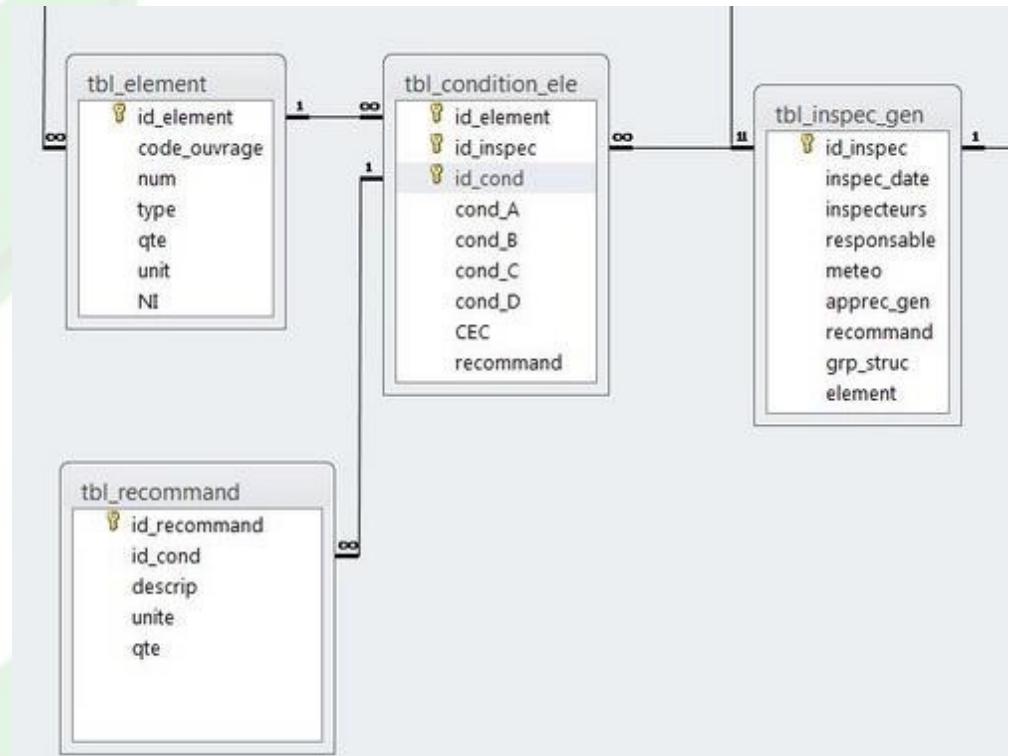
Bir veri tablosunuda yer alan her satır için bir vekil / tanımlayıcı görevi görür. «NOT NULL» ve «UNIQUE» olur. Sütunun Data Type' dan sonra «PRIMARY KEY» yazılır.

```
CREATE TABLE devops(  
    id INT PRIMARY KEY,  
    name VARCHAR(50),  
    email VARCHAR(50),  
    salary REAL,  
    prog_lang VARCHAR(20)  
);
```



# 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 turlu datayı içerebilir. Sayı, String.
- Not : Her tabloda Primary Key olması **zorunlu değildir**.



# FOREIGN KEY

Bir veri tablosunuda yer alan her satır için bir vekil / tanımlayıcı görevi görür. «NOT NULL» ve «UNIQUE» olur. Sütunun Data Type' dan sonra «PRIMARY KEY» yazılır.

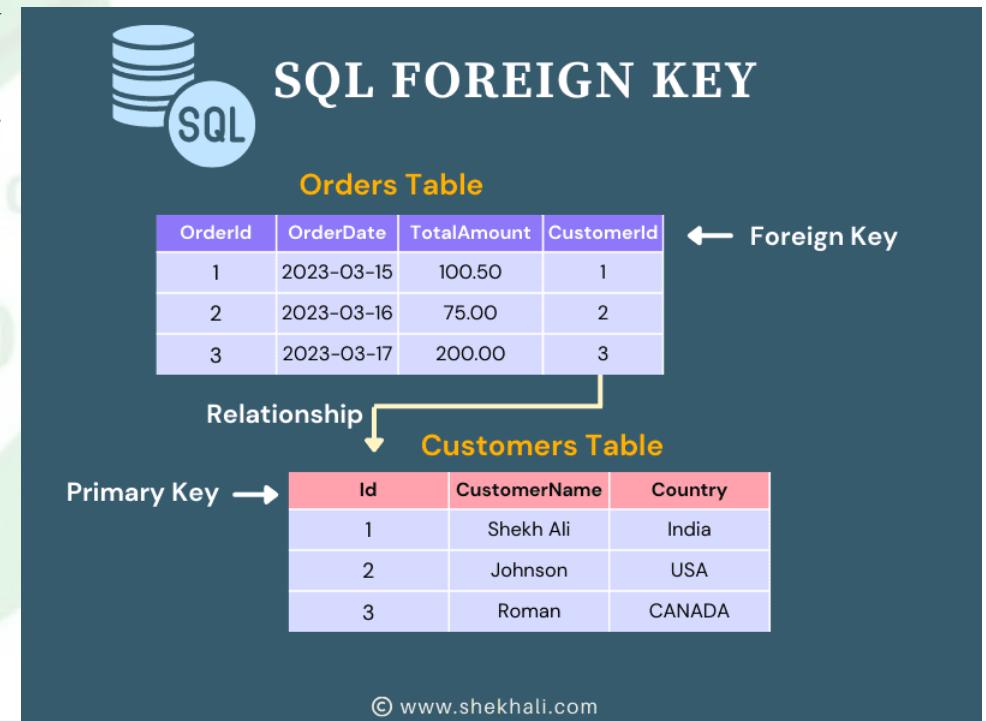
**Not1:** «Parent Table»' da olmayan bir id' ye sahip datayı «Child Table»' a ekleyemezsiniz.

**Not2:** «Child Table» silmeden «Parent Table» silemezsiniz. Önce «Child Table» silinir, sonra «Parent Table» silinir.

**CREATE TABLE devops(**

```
    id INT PRIMARY KEY,  
    name VARCHAR(50),  
    email VARCHAR(50),  
    salary REAL,  
    prog_lang VARCHAR(20)
```

**);**



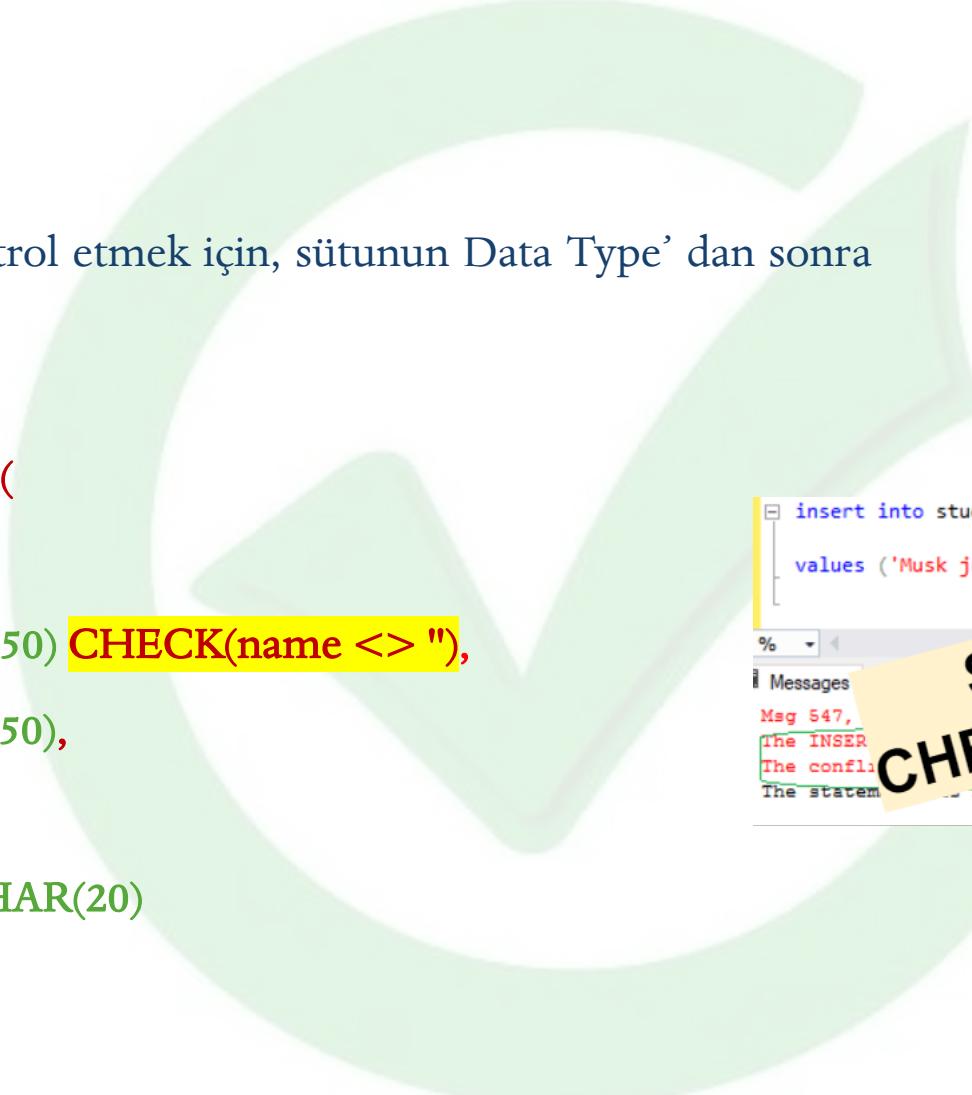
# CHECK

Bir sütunun değerlerini kontrol etmek için, sütunun Data Type' dan sonra «CHECK» yazılır.

**CREATE TABLE** developers(

```
    id SERIAL,  
    name VARCHAR(50) CHECK(name <> ""),  
    email VARCHAR(50),  
    salary REAL,  
    prog_lang VARCHAR(20)
```

);



SQL SERVER  
CHECK CONSTRAINT

?

```
insert into student (StudName ,studAge, Grade)  
values ('Musk jr' ,12 , 'C')  
Msg 547, Level 16, State 1  
The INSERT statement conflicted with the CHECK constraint "con_agrcheck".  
The conflict occurred in database "SqlTutorials", table "dbo.student", column 'studAge'.  
The statement has been terminated.
```



BATCH :  
LESSON :  
DATE :  
SUBJECT :

146 - 149

SQL

26.07.2023

SQL - 3

ZOOM GİRİŞLERİNZİ LÜTFEN **LMS** SİSTEMİ ÜZERİNDEN YAPINIZ





# PostgreSQL

3. Ders

26.07.2023

B149 AWS & DevOps

B146 Cyber Security

# Bugün ne yapıyoruz?

- WHERE



# WHERE

Bir tabloda sorgulamak istenilen verileri filtrelemek için kullanılır.

SELECT \* FROM students;

**students tablosundaki tümdataları getirir.**

SELECT \* FROM students WHERE yazılı\_not > 80;

**students tablosundaki yazılı\_notu 80' den büyük olan kayıtları getirir.**



MySQL

# WHERE Mantıksal Operatörler

- **=** ==> 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

Mantıksal Operatörler	
Operatör	Açıklama
<b>&amp;&amp;</b>	<b>Mantıksal VE</b> Tüm koşulların doğru olması sonucu true üretilir.
<b>  </b>	<b>Mantıksal VEYA</b> Bir koşulların doğru olması sonucu true üretilir.
<b>!</b>	<b>Değil (Tersi)</b> Koşul sonucu true ise false, false ise true yapar.

# DELETE WHERE

- DELETE FROM ogrenciler
  - Tablodaki tümdataları siler, fakat tabloyu silmez.
- “DELETE FROM ogrenciler”
  - Kullanım sonucunda boş bir tablo kalır.
- DELETE FROM ogrenciler WHERE isim = ‘Ali Can’
  - isim olarak Ali Can girilen kaydı (record) siler.
- DELETE FROM ogrenciler WHERE yazılı\_notu = 85
  - not olarak 85 girilen kaydı (record) siler.
- DELETE FROM ogrenciler WHERE isim = ‘Ali Can’ OR veli\_isim=‘Ayse’
  - isim olarak Ali Can veya veli\_isim olarak Ayse girilen kaydı (record) siler.

The screenshot shows two side-by-side tables in a SQL Server Management Studio results grid. The top table is labeled 'BEFORE' and the bottom table is labeled 'AFTER DELETE'. Both tables have columns: id, emp\_name, emp\_age, emp\_salary, join\_date, and phone. The 'before' table contains 10 rows of employee data. The 'after' table contains 8 rows, indicating that the row with id 10 has been deleted.

	id	emp_name	emp_age	emp_salary	join_date	phone
1	1	Mike	35	5000.50	2016-01-01 00:00:00.000	NULL
2	2	Michale	30	4500.00	2016-05-01 00:00:00.000	NULL
3	3	Jimmy	27	3000.00	2016-05-03 00:00:00.000	NULL
4	4	Shaun	30	3500.00	NULL	NULL
5	5	Ben	45	4500.00	2015-03-15 00:00:00.000	NULL
6	6	John	28	2600.00	2018-04-12 19:35:27.067	NULL
7	7	Mike	40	5500.00	NULL	NULL
8	8	Jay	36	5500.00	NULL	NULL
9	10	Imran	35	5500.00	2017-10-13 00:00:00.000	321-456-123456

	id	emp_name	emp_age	emp_salary	join_date	phone
1	1	Mike	35	5000.50	2016-01-01 00:00:00.000	NULL
2	2	Michale	30	4500.00	2016-05-01 00:00:00.000	NULL
3	3	Jimmy	27	3000.00	2016-05-03 00:00:00.000	NULL
4	4	Shaun	30	3500.00	NULL	NULL
5	5	Ben	45	4500.00	2015-03-15 00:00:00.000	NULL
6	6	John	28	2600.00	2018-04-12 19:35:27.067	NULL
7	7	Mike	40	5500.00	NULL	NULL
8	8	Jay	36	5500.00	NULL	NULL

# TRUNCATE

“Truncate” kodu kullanilarak bir tablodaki kayitlar silinirse datalarin geri getirilme ihtimali olmaz

“Truncate” kodu geri getirilmesini (rolling back) istemeyeceginiz tablolari silmek icin kullanilir.

TRUNCATE TABLE customers;

DELETE FROM customers;

INTERVIEW QUESTION :

TRUNCATE, DELETE FROM VE DROP arasindaki fark nedir ?

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



# DROP

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

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

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

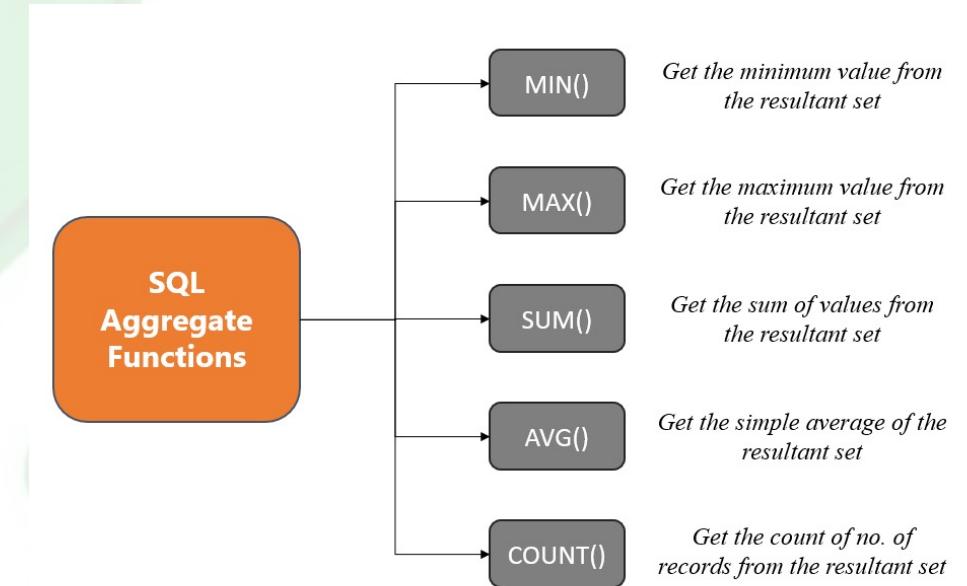
**DROP TABLE** ogrenciler; ogrenciler Tablosunu siler ve RecyleBin'e yollar  
**FLASHBACK TABLE** ogrenciler **TO BEFORE DROP**; dosyayı geri alır.  
**PURGE TABLE** ogrenciler; RecyleBin'de olan dosyayı geri getirilemeyecek şekilde siler  
**DROP TABLE** ogrenciler **PURGE**; Kullandığımız bir dosyayı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

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

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

# AGGREGATE

- Birden çok değer üzerinde bir hesaplama gerçekleştirmek ve hesaplama sonucunda tek bir değer döndürmek için kullanılır.
- SQL; avg, count, sum, min, max vb. İçeren birçok toplama işlevi sağlar.





BATCH :  
LESSON :  
DATE :  
SUBJECT :

146 - 149

SQL

27.07.2023

SQL - 4

ZOOM GİRİŞLERİNİZİ LÜTFEN **LMS** SİSTEMİ ÜZERİNDEN YAPINIZ





# PostgreSQL

4. Ders

27.07.2023

B149 AWS & DevOps

B146 Cyber Security

# Bugün ne yapıyoruz?

- SUBQUERIES



# SUBQUERIES

-SUBQUERY başka bir SQL sorgusunun içinde bulunan ve onun sonucundan yararlanarak çalışan bir sorgudur. Bu sayede, veritabanında bulunan verileri daha karmaşık ve detaylı bir şekilde sorgulama imkanı sağlar.

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

id integer	isim character varying (50)	sehir character varying (50)	maas integer	isyeri character varying (20)	marka_id integer	marka_isim character varying (20)	calisan_sayisi integer
123456789	Ali Seker	Istanbul	2500	Vakko	100	Vakko	12000
234567890	Ayse Gul	Istanbul	1500	LCWaikiki	101	Pierre Cardin	18000
345678901	Veli Yilmaz	Ankara	3000	Vakko	102	Adidas	10000
456789012	Veli Yilmaz	Izmir	1000	Pierre Cardin	103	LCWaikiki	21000
567890123	Veli Yilmaz	Ankara	7000	Adidas			
456789012	Ayse Gul	Ankara	1500	Pierre Cardin			
123456710	Fatma Yasa	Bursa	2500	Vakko			

# SUBQUERIES

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

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

SIRKET
Ford
Toyota

Subquery sonugu

ISIM	SIRKET
Ayse Gul	Toyota
Veli Yilmaz	Ford
Ayse Gul	Ford

Query sonugu

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

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

SIRKET
Hyundai
Toyota

Subquery sonugu

SEHIR	MAAS
Ankara	7000
Istanbul	1500

Query sonugu

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 kullanılır.

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

Dolayısıyla **SUM**, **COUNT**, **MIN**, **MAX** ve **AVG** gibi fonksiyonlar kullanılır.

Bu fonksiyonlara AGGREGATE FUNCTION denir.

**SORU 1-** Her sirketin ismini, personel sayisini ve personelin ortalama maasini listeleyen bir QUERY yazin.

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

# SUBQUERIES

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

```
SELECT sirket,personel_sayisi,(SELECT MAX(maas)
                                FROM personel
                               WHERE sirketler.sirket=personel.sirket
                            ) AS sirketteki_Max_Maas,
FROM sirketler;
```

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

# SUBQUERIES

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

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

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

# SUBQUERIES

**SORU 4-** Her sirketin ismini, personel sayisini ve personelin aldigı max. ve min. maasi listeleyen bir QUERY yazın.

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

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

# SUBQUERIES

**SORU 5-** Her sirketin ismini, personel sayisini ve tablodaki iscilerine odedigi toplam maasi listeleyen bir QUERY yazin.

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

SIRKET	PERSONEL SAYISI	TOPLAM MAAS
Honda	12000	8000
Ford	18000	2500
Hyundai	10000	7000
Toyota	21000	1500

# EXISTS

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

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

NİSAN		
urun_id	musteri_isim	urun_isim
10	Hasan	Honda
10	Kemal	Honda
20	Ayse	Toyota
50	Yasar	Volvo
20	Mine	Toyota

urun_id	musteri_isim
10	Mark
20	John
20	Mark
10	Adam
20	Eddie



BATCH :  
LESSON :  
DATE :  
SUBJECT :

146 - 149

SQL

28.07.2023

SQL - 5

ZOOM GİRİŞLERİNİZİ LÜTFEN **LMS** SİSTEMİ ÜZERİNDEN YAPINIZ





# PostgreSQL

5. Ders

28.07.2023

B149 AWS & DevOps

B146 Cyber Security

# Bugün ne yapıyoruz?

- SUBQUERIES



# UPDATE

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

```
CREATE TABLE mart_satislar  
(  
    urun_id number(10),  
    musteri_isim varchar2(50),  
    urun_isim varchar2(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) Ismi hatice olan musterinin urun\_id'sini 30,urun\_isim'ini Ford yapin
- 2) Toyata marka araclara %10 indirim yapin
- 3) Ismi Ali olanların urunfiyatlarını %15 artirin
- 4) Honda aracların urun kodu'nu 50 yapin

# UPDATE

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

```
CREATE TABLE tedarikciler
(
    id number(10),
    isim varchar2(50),
    iletisim_isim varchar2(50),
    CONSTRAINT tedarikci_pk PRIMARY KEY (id, isim)
);
```

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

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

- 2) Icine 3 kayit ekleyin (1, 'ACB', 'Ali Can'), (2, 'RDB', 'Veli Gul'), (3, 'KMN', 'Ayse Gulmez').

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

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

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

# ORDER BY

**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 varchar2(50),  
    soyisim varchar2(50),  
    adres varchar2(50)  
)
```

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

Insanlar tablosundaki datalari adres'e gore siralayin

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

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

# ORDER BY

İnsanlar tablosundaki ismi Mine olanları SSN sıralı olarak listeleyin

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

SSN	TSTM	SOYTSTM	ADRES
344678981	Mine	Yasa	Ankara
345678981	Mine	Bulut	Ankara

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

İnsanlar tablosundaki soyismi Bulut olanları isim sıralı olarak listeleyin

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

SSN	TSTM	SOYTSTM	ADRES
255789912	Mahmut	Bulut	İstanbul
345678981	Mine	Bulut	Ankara

# ORDER BY

İnsanlar tablosundaki tüm kayıtları SSN numarası büyükten küçüğe olarak sıralayın

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

SSN	TSTM	SOYTSTM	ADRES
345678901	Mine	Bulut	Ankara
345678901	Veli	Vilmaz	Istanbul
344578901	Mine	Yasa	Ankara
296789012	Mahmut	Bulut	Istanbul
234567899	Veli	Cem	Ankara
123456789	Allı	Can	Istanbul

İnsanlar tablosundaki tüm kayıtları isimler Natural sıralı, Soyisimler ters sıralı olarak listeleyin

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

SSN	TSTM	SOYTSTM	ADRES
123456789	Allı	Can	Istanbul
296789012	Mahmut	Bulut	Istanbul
344578901	Mine	Yasa	Ankara
345678901	Mine	Bulut	Ankara
345678901	Veli	Vilmaz	Istanbul
234567899	Veli	Cem	Ankara

# UNION

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

```
CREATE TABLE personel
```

```
(  
    id number(9),  
    isim varchar2(50),  
    sehir varchar2(50),  
    maas number(20),  
    sirket varchar2(20)  
)
```

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

- 1) Maasi 4000'den cok olanisci isimlerini ve 5000 liradan fazla maas alınan sehirleri 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
Istanbul	5500
Izmir	6000
Mehmet Ozturk	5000
Mehmet Ozturk	7000
Veli Sahin	4500
Veli Sahin	4500

# UNION

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

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

**NOT :** UNION ALL ile birlestirmelerde de

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

# UNION

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	SERIYET
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
456715812	Veli Sahin	Ankara	4500	Ford
123456718	Hatice Sahin	Bursa	4500	Honda

# LIKE

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

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

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

# NOT LIKE

**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	aden	4
1006	salim	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	hit	3
1006	hct	3
1007	salim	5
1007	yusuf	5