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 NELERDIR**

**1) Yuksek miktarda bilgi depolanabilir**

**2) Olusturma, Okuma,Degistirme ve Silme kolayligi Create,Read,Update,Delete (CRUD)**

**3) Girisin kolay ve kontrollu olmasi**

**4) Dataya ulasim kolayligi**

**5) Guvenlik**

**API Application Programming Interface,**

bir uygulamaya ait yeteneklerin, başka bir uygulamada da kullanılabilmesi için, yeteneklerini paylaşan uygulamanın sağladığı arayüzdür

**END To END (E2E) Testing**

1 Eger datayi User Interface (UI) kullanarak yolladiysaniz

A) Datayi UI dan arama fonksiyonunu kullanarak dogrula (Selenium)

B) Datayi SQL kodlarini kullanarak dogrula (SQL + Selenium)

C) Datayi API kodlarini kullanarak dogrula (API + Selenium)

2) Eger datayi SQL kodlarini kullanarak yolladiysaniz

A) Datayi UI dan arama fonksiyonunu kullanarak dogrula (Selenium)

B) Datayi SQL kodlarini kullanarak dogrula (SQL + Selenium)

C) Datayi API kodlarini kullanarak dogrula (API + Selenium)

3) Eger datayi API kodlarini kullanarak yolladiysaniz

A) Datayi UI dan arama fonksiyonunu kullanarak dogrula (Selenium)

B) Datayi SQL kodlarini kullanarak dogrula (SQL + Selenium)

C) Datayi API kodlarini kullanarak dogrula (API + Selenium)

**Database Management System (DBMS) –**

Veritabanlarını yönetmek, kullanmak, geliştirmek ve bakımını yapmak için kullanılan yazılımlara denir. *Database’e erisimi duzenler*

*Create,Read,Update,Delete islemlerini duzenler*

*Data guvenligini saglar*

*Sorgular oluşturur ve sorgular iletilir,*

*Raporlar oluşturur ve raporları işletir,*

*Uygulamayı kontrol eder*

*Diger uygulamalarla (Application) iletisimi saglar.*

**Cok Kullanilan VTYS(Veri Tabani Yonetim Sistemleri)**

SQL Server : Microsoft tarafindan gelistirilmistir Negatif: Pahali – Kurumsal Kullanicilar icin binlerce dolar odenmesi gereklidir Pozitif : Zengin bir user interface’e sahip ve cok buyuk datalarin kullanilmasinda sorunsuz calisir.

MySQL Server : Isvecli MySQL firmasi tarafindan gelistirildi. 2010’da Oracle satin aldi Negatif: Eszamanli cok fazla islem girildiginde calismayi durdurabilir. Pozitif: Acik kaynak. Online destek ve ucretsiz cok fazla dokuman var

PostgreSQL Server : Created by a computer science professor Michael Stonebraker. Yeni nesil olarak ortaya cikti. Kisisellestirme mumkundur, zor gorevler icin ideal olabilir.

PL/SQL Oracle database sunuculari icinde depolanir PL/SQL SQL komutlarini ozellikle karsilamak uzere dizayn edilmistir. Pros: PL/SQL yuksek guvenlik seviyesi saglar ve Object-Oriented Programing’e uyumludur.

**RELATIONAL DATABASES (ILISKILI TABLOLAR)**

SQL tablolar datalari iliskili tablolarda depolar.

Tablolar arasi iliskiler net olmalidir.

Tablolar arasi gecis kolay olmalidir

Tablolarin ve iliskilerin butunune SCHEMA denir

Relational Databases, SQL Databases (Structured Query Language) olarak da adlandirilir.

**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.

**--DQL --> Data Query Lang.**

**--SELECT: Veritabanındaki verileri alır.**

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

select \* from student;

select st\_name from student;

**--SELECT KOMUTU WHERE KOŞULU**

select \* from student WHERE age>35;

**--TCL - Transaction Control Lang.**

**--Begin - Savepoint - rollback - commit**

**--Transaction veritabanı sistemlerinde bir işlem başladığında başlar ve işlem bitince sona erer**

**--Bu işlemler veri tabanı oluşturma, veri silme, veri güncelleme, veriyi geri getirme gibi işlemler olabilir**

CREATE TABLE ogrenciler2(

id serial,

isim VARCHAR(50),

veli\_isim VARCHAR(50),

yazılı\_notu real

);

Begin;

insert into ogrenciler2 VALUES (default,'Ali Can','Hasan',75.5);

insert into ogrenciler2 VALUES (default,'Canan Gül','Ayşe Şen',90.5);

savepoint x;

insert into ogrenciler2 VALUES (default,'Kemal Can','Ahmet Can',85.5);

insert into ogrenciler2 VALUES (default,'Ahmet Şen','Ayşe Can',65.5);

ROLLBACK TO x;

select \* from ogrenciler2

commit;

**--Transaction kullanımında SERIAL data türü kullanımı tavsiye edilmez.**

**--savepointten sonra eklediğimiz veride sayaç mantığı ile çalıştığı için**

**--sayacta en son hangi sayıda kaldıysa ordan devam eder**

**-- NOT : PostgreSQL de transaction kullanımı için 'Begin' komutuyla başlarız sonrasında tekrar yanlış bir veriyi**

**-- düzeltmek veya bizim için önemli olan verilerden sonra ekleme yapabilmek için 'SAVEPOINT savepointadi'**

**-- kullanırız ve bu savepointe dönebilmek için 'ROLLBACK TO savepointismi'**

**-- komutunu kullanırız ve rollback çalıştırıldığında savepoint yazdığımız satırın üstündeki verileri tabloda bize**

**-- verir ve son olarak Transaction'ı sonlandırmak için mutlaka 'COMMIT' komutu kullanırız. MySQL de**

**-- transaction olmadanda kullanılır**

**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

**-- DELETE - TRUNCATE --**

**-- TRUNCATE komutu DELETE komutu gibi bir tablodaki verilerin tamamını siler.**

**-- Ancak, seçmeli silme yapamaz**

**3. Veri Tanimlama 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.

-- CONSTRANINT -- Kısıtlamalar

**-- Primary Key --> Bir sutunun NULL içermemesini ve sutundaki verilerin BENZERSİZ olmasını sağlar (NOT NULL - UNIQUE)**

**-- FOREGIN KEY --> Başka bir tablodaki PRİMARY KEY'i referans göstermek için kullanılır. Böylelikle, tablolar arasında ilişki kurmuş oluruz.**

**-- UNIQUE --> Bir sutundaki tüm değerlerin BENZERSİZ yani tek olmasını sağlar**

**-- NOT NULL --> Bir sutunun NULL içermemesini yani boş olmamasını sağlar**

**-- NOT NULL kısıtlaması için CONSTRAINT ismi tanımlanmaz. Bu kısıtlama veri türünden hemen sonra yerleştirilir**

**-- CHECK --> Bir sutuna yerleştirilebilecek değer aralığını sınırlamak için kullanılır**.

**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.

-----------------------------------------------------------------------------------------------------------------------------

**--FOREIGN KEY--**

CREATE TABLE adresler

(

adres\_id char(5) ,

sokak varchar(20),

cadde varchar(30),

sehir varchar(20),

CONSTRAINT fk FOREIGN KEY (adres\_id) REFERENCES calısanlar(id)

);

INSERT INTO adresler VALUES('10003','Mutlu Sok', '40.Cad.','IST');

INSERT INTO adresler VALUES('10003','Can Sok', '50.Cad.','Ankara');

INSERT INTO adresler VALUES('10002','Ağa Sok', '30.Cad.','Antep');

select \* from adresler;

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

**--Parent tabloda olmayan id ile child tabloya ekleme yapamayız**

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

**--Calısanlar id ile adresler tablosundaki adres\_id ile eşlesenlere bakmak için**

select \* from calısanlar,adresler WHERE calısanlar.id = adresler.adres\_id;

**DROP table calısanlar**

**--Parant tabloyu yani primary key olan tabloyu silmek istediğimizde tabloyu silmez**

**--Önce child tabloyu silmemiz gerekir**

delete from calısanlar where id=10002; --parent

delete from adresler WHERE adres\_id = '10002'; --child

**drop table calısanlar;**

**-- ON DELETE CASCADE --**

**--Her defasında önce child tablodaki verileri silmek yerine**

**--ON DELETE CASCADE silme özelliğini aktif hale getirebiliriz**

**--Bunun için FK olan satırın en sonuna ON DELETE CASCADE komutunu yazmamız yeterli**

CREATE TABLE talebeler

(

id CHAR(3) primary key,

isim VARCHAR(50),

veli\_isim VARCHAR(50),

yazılı\_notu int

);

select \* from talebeler

INSERT INTO talebeler VALUES(123, 'Ali Can', 'Hasan',75);

INSERT INTO talebeler VALUES(124, 'Merve Gul', 'Ayse',85);

INSERT INTO talebeler VALUES(125, 'Kemal Yasa', 'Hasan',85);

INSERT INTO talebeler VALUES(126, 'Nesibe Yılmaz', 'Ayse',95);

INSERT INTO talebeler VALUES(127, 'Mustafa Bak', 'Can',99);

CREATE TABLE notlar(

talebe\_id char(3),

ders\_adi varchar(30),

yazili\_notu int,

CONSTRAINT notlar\_fk FOREIGN KEY (talebe\_id) REFERENCES talebeler(id)

on delete cascade

);

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);

select \* from talebeler

select \* from notlar

delete from notlar where talebe\_id ='123' **-- child table**

delete from talebeler where id ='126' **-- on delete cascade kullandıgımız ıcın parent table'dan direkt silebildik**

**--Parent table'dan silebildigimiz icin chil table'dan da silinmis olur**

Delete from talebeler

DROP table talebeler CASCADE**;-- PARENT tablouy kaldırmak istersek Drop table tablo\_adı' ndan sonra CASCADE komutunu kullanırız**

**--Talebeler tablosubdaki isim sütununa NOT NULL kısıtlaması ekleyiniz ve veri tipini varchar(30) olarak değiştiriniz**

alter table talebeler

alter column isim TYPE VARCHAR(30),

ALTER COLUMN isim SET NOT NULL;

**--talebeler tablosundaki yazılı\_notu sutununa 60’dan buyuk rakam giriniz**

alter table talebeler

ADD CONSTRAINT sinir CHECK(yazılı\_notu>60)

**--CHECK komutu ile tablodaki istedigimz sutunu sınırlandırabiliriz**

INSERT INTO talebeler VALUES(128,'Mustafa CAN','Hasan',45);

**-- yukarda 60'i sinir olarak belirledigimiz icin bunu eklemedi asagıdaki eror verdi**

**--ERROR: new row for relation "talebeler" violates check constraint "sinir"**

**-- DETAIL: Failing row contains (128, Mustafa CAN, Hasan, 45).**

create table ogrenciler(

id int,

isim varchar(45),

adres varchar(100),

sinav\_notu int

);

Create table ogrenci\_adres

AS

SELECT id, adres from ogrenciler;

select \* from ogrenciler

select \* from ogrenci\_adres

**--Tablodaki bir sutuna PRIMARY KEY ekleme**

alter table ogrenciler

ADD PRIMARY KEY (id);

**--PRIMERY KEY olusturmada 2. yol:**

alter table ogrenciler

ADD CONSTRAINT pk\_id PRIMARY KEY (id);

**--PK'den sonra FK ataması;**

alter table ogrenci\_adres

ADD foreign key (id) REFERENCES ogrenciler ;

alter table ogrenciler DROP CONSTRAINT pk\_id;

alter table ogrenciler DROP CONSTRAINT ogrenci\_adres\_id\_fkey;

--Yazılı notu 85'ten buyuk olan talebe bilgileri:

Select \* from talebeler WHERE yazılı\_notu>85

--ismi Mustafa Bak olan talebenin tum bilgieri:

select \* from talebeler where isim='Mustafa Bak'

**--SELECT komutunda --BETWEEN kosulu**

create table personel

(

id char(4),

isim varchar(50),

maas int

);

insert into personel values('1001', 'Ali Can', 70000);

insert into personel values('1002', 'Veli Mert', 85000);

insert into personel values('1003', 'Ayşe Tan', 65000);

insert into personel values('1004', 'Derya Soylu', 95000);

insert into personel values('1005', 'Yavuz Bal', 80000);

insert into personel values('1006', 'Sena Beyaz', 100000);

**/\***

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

**Bir tanesi gerçekleşmezse listelemez**

**Select \* from matematik sinav1 > 50 AND sınav2 > 50**

**Hem sınav1 hemde sınav2 alanı, 50'den büyük olan kayıtları listeler**

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

**select \* From matematik sınav>50 OR sınav2>50**

**Hem sınav1 veya sınav2 alanı, 50 den büyük olan kayıtları listeler**

**\*/**

select \* from personel

select \* from personel WHERE id BETWEEN '1003' and '1005';

-- 2. yol;

select \* from personel WHERE id >= '1003' and id<='1005';

select \* from personel WHERE isim between 'Derya Soylu' AND 'Yavuz Bal'

select \* from personel WHERE maas=70000 or isim='Sena Beyaz'

**--IN : Birden fazla mantıksal ifade ile tanımlayabileceğimiz durumları tek komutta yazabilme imkanı verir**

**--Farklı sütunlar için IN kullanılamaz**

**--id'si 1001,1002 ve 1004 olan personelin bilgilerini listele**

**--1.yol:**

select \* from personel where id ='1001' or id='1002' or id ='1004';

**--2. yol:**

select \* from personel WHERE id IN ('1001','1002','1004');

select \* from personel WHERE maas in (70000,100000);

**/\***

**SELECT - LIKE koşulu**

**LIKE : Sorgulama yaparken belirli kalıp ifadeleri kullanabilmemizi sağlar**

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

**LIKE : ~~**

**ILIKE : ~~\***

**NOT LIKE : !~~**

**NOT ILIKE :!~~\***

**% --> 0 veya daha fazla karakteri belirtir.**

**\_ --> Tek bir karakteri belirtir**

**\*/**

**-- Ismi A harfi ile baslayan personeli listele**

select \* from personel WHERE isim LIKE 'A%';

**--Ismi t harfi ile biten personeli listele**

select \* from personel where isim like '%t';

**--Isminin 2. Harfi e olan personeli listeleyiniz**

select \* from personel where isim like '\_e%';

**-- Isminin 3.harfi r olan personeli listeleyiniz**

select \* from personel WHERE isim LIKE '\_\_r%'

**4.gun – 1.ders**

create table personel(

id char(4),

isim varchar(50),

maas int

);

insert into personel values('1001', 'Ali Can', 70000);

insert into personel values('1002', 'Veli Mert', 85000);

insert into personel values('1003', 'Ayşe Tan', 65000);

insert into personel values('1004', 'Derya Soylu', 95000);

insert into personel values('1005', 'Yavuz Bal', 80000);

insert into personel values('1006', 'Sena Beyaz', 100000);

-- Ismi A harfi ile baslayan personeli listele

select \* from personel where isim like 'A%';

-- Isminin 2. harfi e olan personeli listeleyiniz

select \* from personel where isim like '\_e%';

-- Ismi t harfi ile biten personeli listele

select \* from personel where isim like '%t';

-- 'a' ile başlayıp 'n' ile biten personel isimlerini listeleyiniz

select isim from personel where isim ~~\* 'a%n';

-- ikinci karakteri 'a' ve dördüncü karakteri 'u' olan personel isimlerini listeleyiniz

select isim from personel where isim ~~\* '\_a\_u%';

-- İçinde 'e' ve 'r' bulunan personel isimlerini listeleyiniz

select \* from personel where isim ILIKE '%e%' and isim ilike '%r%';

--select isim from personel where isim ~~\* '%e%' and isim ~~\* '%r%';

-- İçinde 'e' ve 'r' bulunan personel isimlerini listeleyiniz

select \* from personel where isim ILIKE '%e%' or isim ilike '%r%';

-- 2. harfi e olup diğer harflerinde y olan personeli listeleyiniz

select \* from personel where isim ~~\* '\_e%y%';

-- a harfi olmayan personeli listeleyin

--// select \* from personel where isim not like '%a%';

select \* from personel where isim !~~ '%a%';

-- 1. harfi A ve 7. harfi a olan personeli listeleyin

select \* from personel where isim like 'A\_\_\_\_\_a%';

--//select \* from personel where isim ~~\* 'A\_\_\_\_\_a%';

--Sondan ikinci karakteri 'r' olan "isim" değerlerini listeleyiniz.

select \* from personel where isim ~~\* '%r\_';

**4.gun-2.ders**

-- SELECT - SIMILAR TO - REGEX(Regular Expressions) --

/\*

SIMILAR TO : Daha karmaşık pattern(kalıp) ile sorgulama işlemi için SIMILAR TO kullanılabilir.

Sadece PostgreSQL de kullanılır. Büyük Küçük harf önemlidir

REGEX : Herhangi bir kod, metin içerisinde istenen yazı veya kod parçasının aranıp bulunmasını sağlayan

kendine ait bir söz dizimi olan bir yapıdır.MySQL de(REGEXP\_LİKE) olarak kullanılır

PostgreSQL'de "~" karakteri ile kullanılır.

\*/

CREATE TABLE kelimeler(

id int,

kelime VARCHAR(50),

harf\_sayısı int

);

INSERT INTO kelimeler VALUES (1001, 'hot', 3);

INSERT INTO kelimeler VALUES (1002, 'hat', 3);

INSERT INTO kelimeler VALUES (1003, 'hit', 3);

INSERT INTO kelimeler VALUES (1004, 'hbt', 3);

INSERT INTO kelimeler VALUES (1005, 'hct', 3);

INSERT INTO kelimeler VALUES (1006, 'adem', 4);

INSERT INTO kelimeler VALUES (1007, 'selim', 5);

INSERT INTO kelimeler VALUES (1008, 'yusuf', 5);

INSERT INTO kelimeler VALUES (1009, 'hip', 3);

INSERT INTO kelimeler VALUES (1010, 'HOT', 3);

INSERT INTO kelimeler VALUES (1011, 'hOt', 3);

INSERT INTO kelimeler VALUES (1012, 'h9t', 3);

INSERT INTO kelimeler VALUES (1013, 'hoot', 4);

INSERT INTO kelimeler VALUES (1014, 'haaat', 5);

INSERT INTO kelimeler VALUES (1015, 'hooooot', 5);

INSERT INTO kelimeler VALUES (1016, 'booooot', 5);

INSERT INTO kelimeler VALUES (1017, 'bolooot', 5);

-- İçerisinde 'ot' veya 'at' bulunan kelimeleri listeleyiniz

select \* from kelimeler where kelime SIMILAR TO '%(at|ot)';

--// select \* from kelimeler where ILIKE kelime '%at%' or kelime ILIKE '%ot%';

select \* from kelimeler where kelime SIMILAR TO '%(at|ot|Ot|oT|At|aT|OT)';

select \* from kelimeler WHERE kelime ~~\* '%at%' or kelime ~~\* '%ot%';

--REGEX

select \* from kelimeler WHERE kelime ~\* 'ot' or kelime ~\* 'at';

-- : 'ho' veya 'hi' ile başlayan kelimeleri listeleyeniz

select \* from kelimeler WHERE kelime ~~\* 'ho%' or kelime ~~\*'hi%';

--select \* from kelimeler WHERE kelime similar to 'ho%|hi%';

--REGEX ile;

select \* from kelimeler where kelime ~\* 'h[oi](.\*)' --Regex'te ".(nokta) bir karakteri temsil eder"

--Regex'de ikinci karakter için köşeli parantez kullanılır. \* hepsi anlamında kullanılır

--SIMILAR TO ile

--Sonu 't' veya 'm' ile bitenleri listeleyeniz

select \* from kelimeler where kelime similar to '%t|%m';

--REGEX

select \* from kelimeler WHERE kelime ~\* '(.\*)[tm]$';-- $ karakteri bitişi gösterir

-- h ile başlayıp t ile biten 3 harfli kelimeleri listeleyeniz

select \* from kelimeler where kelime similar to 'h[a-z,A-Z,0-9]t';

--LIKE ile;

select \* from kelimeler WHERE kelime ~~\* 'h\_t';

--REGEX

select \* from kelimeler WHERE kelime ~\* 'h[a-z,A-Z,0-9]t';

--İlk karakteri 'h', son karakteri 't' ve ikinci karakteri 'a'dan 'e'ye herhangi bir karakter

--olan “kelime" değerlerini çağırın.

-- SIMILAR TO ile

select \* from kelimeler where kelime similar to 'h[a-e](.\*)t';

select kelime from kelimeler where kelime ~~\* 'h[a-e])(.\*)t';

--İlk karakteri 's', 'a' veya 'y' olan "kelime" değerlerini çağırın.

select \* from kelimeler where kelime ~ '^[say](.\*)';

⇒ ‘ ^ ’ baslangici temsil eder

--Son karakteri 'm', 'a' veya 'f' olan "kelime" değerlerini çağırın.

select \* from kelimeler where kelime ~ '(.\*)[maf]$'

--regex ile

select \* from kelimeler where kelime ~ '(.\*)[maf]';

--similar to ile

select \* from kelimeler where kelime similar to '%[maf]';

--İlk harfi h, son harfi t olup 2.harfi a veya i olan 3 harfli kelimelerin tüm bilgilerini sorgulayalım.

-- similar to ile ;

select \* from kelimeler where kelime similar to 'h[a|i]t';

--regex ile ;

select \* from kelimeler where kelime ~ '^h[a|i]t$';

--ilike ile ;

select \* from kelimeler WHERE kelime ILIKE 'h%t' and ('\_a%' or '\_i%') and length(kelime)=3

--İlk harfi 'b' dan ‘s' ye bir karakter olan ve ikinci harfi herhangi bir karakter olup

--üçüncü harfi ‘l' olan “kelime" değerlerini çağırın.

select \* from kelimeler WHERE kelime similar to '[b-s].l(.\*)';

select \* from kelimeler WHERE kelime ~ '^[b-s].l(.\*)';

--içerisinde en az 2 adet oo barıdıran kelimelerin tüm bilgilerini listeleyiniz.

select \* from kelimeler where kelime similar to '%[o][o]%';

select \* from kelimeler where kelime similar to '%[o]{2}%';

--⇒ Süslü parantez içinde belirttiğimiz rakam bir önceki köşeli parantez içinde kaçtane olduğunu belirtir

--içerisinde en az 4 adet oooo barıdıran kelimelerin tüm bilgilerini listeleyiniz.

select \* from kelimeler where kelime similar to '%[o]{4}%';

--a', 's' yada 'y' ile başlayan VE 'm' yada 'f' ile biten "kelime" değerlerini çağırın

⇒ select \* from kelimeler where kelime similar to '^[a|s|y](.\*)[m|f]$';

//select \* from kelimeler where kelime similar to '[a|s|y](.\*)%[m|f]';

CREATE TABLE calisanlar2

(

id int,

isim VARCHAR(50),

sehir VARCHAR(50),

maas int,

isyeri VARCHAR(20)

);

INSERT INTO calisanlar2 VALUES(123456789, 'Ali Seker', 'Istanbul', 2500, 'Vakko');

INSERT INTO calisanlar2 VALUES(234567890, 'Ayse Gul', 'Istanbul', 1500, 'LCWaikiki');

INSERT INTO calisanlar2 VALUES(345678901, 'Veli Yilmaz', 'Ankara', 3000, 'Vakko');

INSERT INTO calisanlar2 VALUES(456789012, 'Veli Yilmaz', 'Izmir', 1000, 'Pierre Cardin');

INSERT INTO calisanlar2 VALUES(567890123, 'Veli Yilmaz', 'Ankara', 7000, 'Adidas');

INSERT INTO calisanlar2 VALUES(456789012, 'Ayse Gul', 'Ankara', 1500, 'Pierre Cardin');

INSERT INTO calisanlar2 VALUES(123456710, 'Fatma Yasa', 'Bursa', 2500, 'Vakko');

CREATE TABLE markalar

(

marka\_id int,

marka\_isim VARCHAR(20),

calisan\_sayisi int

);

INSERT INTO markalar VALUES(100, 'Vakko', 12000);

INSERT INTO markalar VALUES(101, 'Pierre Cardin', 18000);

INSERT INTO markalar VALUES(102, 'Adidas', 10000);

INSERT INTO markalar VALUES(103, 'LCWaikiki', 21000);

select \* from calisanlar2

select \* from markalar

-- calisan sayisi 15.000’den cok olan markalarin isimlerini --->

-- --> ve bu markada calisanlarin isimlerini ve maaşlarini listeleyin.

SELECT isim,maas,isyeri from calisanlar2

WHERE isyeri IN(SELECT marka\_isim from markalar WHERE calisan\_sayisi>15000);

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

SELECT isim,maas,sehir from calisanlar2

where isyeri IN(select marka\_isim from markalar where marka\_id>101)

-- Ankara’da calisani olan markalarin marka id'lerini ve calisan sayilarini listeleyiniz.

select marka\_id,calisan\_sayisi from markalar

where marka\_isim IN(SELECT isyeri from calisanlar2 where sehir='Ankara' )

-- calisan sayisi 3.000’den cok olan markalarin isimlerini --->

-- --> ve bu markada calisanlarin isimlerini ve maaşlarini listeleyin.

select isyeri,isim,maas from calisanlar2

where isyeri IN(select marka\_isim from markalar where calisan\_sayisi>3000);

-- marka\_id’si 101’den kucuk olan marka çalişanlarinin isim, maaş ve şehirlerini listeleyiniz.

select isim,maas,sehir from calisanlar2

where isyeri IN(select marka\_isim from markalar where marka\_id<101);

-- Bursa’da calisani olan markalarin marka id'lerini ve calisan sayilarini listeleyiniz.

select marka\_id,calisan\_sayisi from markalar

where marka\_isim IN(select isyeri from calisanlar2 where sehir='Bursa');

-- İzmir'de calisani olan markalarin marka id'lerini ve calisan sayilarini listeleyiniz.

select marka\_id,calisan\_sayisi from markalar

where marka\_isim IN(select isyeri from calisanlar2 where sehir='Izmir');

**5.gun-1.ders**

**-- EXISTS , NOT EXISTS --**

**-- EXISTS condition subquerylerde kullanilir**

**-- IN ifadesinin kullanimina benzer**

**-- EXISTS , NOT EXISTS ifadeleri de altsorgudan getirilen degerlerin icerisinde**

**--bir degerin olmasi veya olmamasi durumunda islem yapilmasini saglar**

**-- Exists true false gibidir --> Boolean**

CREATE TABLE mart(

urun\_id int,

musteri\_isim varchar(50),

urun\_isim varchar(50)

);

CREATE TABLE nisan(

urun\_id int ,

musteri\_isim varchar(50),

urun\_isim varchar(50)

);

INSERT INTO mart VALUES (10, 'Mark', 'Honda');

INSERT INTO mart VALUES (20, 'John', 'Toyota');

INSERT INTO mart VALUES (30, 'Amy', 'Ford');

INSERT INTO mart VALUES (20, 'Mark', 'Toyota');

INSERT INTO mart VALUES (10, 'Adam', 'Honda');

INSERT INTO mart VALUES (40, 'John', 'Hyundai');

INSERT INTO mart VALUES (20, 'Eddie', 'Toyota');

INSERT INTO nisan VALUES (10, 'Hasan', 'Honda');

INSERT INTO nisan VALUES (10, 'Kemal', 'Honda');

INSERT INTO nisan VALUES (20, 'Ayse', 'Toyota');

INSERT INTO nisan VALUES (50, 'Yasar', 'Volvo');

INSERT INTO nisan VALUES (20, 'Mine', 'Toyota');

select \* from mart

select \* from nisan

-- MART VE NİSAN aylarında aynı URUN\_ID ile satılan ürünlerin

--URUN\_ID’lerini listeleyen ve aynı zamanda bu ürünleri MART ayında alan

--MUSTERI\_ISIM 'lerini listeleyen bir sorgu yazınız.

SELECT urun\_id,muteri\_isim from mart

WHERE EXISTS (SELECT urun\_id from nisan WHERE mart.urun\_id=nisan.urun\_id)

--Her iki ayda birden satılan ürünlerin URUN\_ISIM'lerini ve bu ürünleri

--NİSAN ayında satın alan MUSTERI\_ISIM'lerini listeleyen bir sorgu yazınız.

SELECT urun\_isim,musteri\_isim FROM nisan

WHERE exists (SELECT urun\_isim from mart WHERE nisan.urun\_isim=mart.urun\_isim)

--Her iki ayda ortak satilmayan ürünlerin URUN\_ISIM'lerini ve bu ürünleri

--NİSAN ayında satın alan MUSTERI\_ISIM'lerini listeleyen bir sorgu yazınız.

SELECT urun\_isim,musteri\_isim from nisan

WHERE not exists (SELECT urun\_isim from mart WHERE mart.urun\_isim=nisan.urun\_isim)

--Her iki ayda ortak satilmayan ürünlerin urun\_id'lerini ve bu ürünleri

--Mart ayında satın alan MUSTERI\_ISIM'lerini listeleyen bir sorgu yazınız.

SELECT urun\_id,musteri\_isim from mart

where not exists (SELECT urun\_id from nisan WHERE nisan.urun\_id=mart.urun\_id)

--Her iki ayda birden satılan ürünlerin urun\_id'lerini ve bu ürünleri

--NİSAN ayında satın alan URUN\_ISIM'lerini listeleyen bir sorgu yazınız.

SELECT urun\_id,urun\_isim from nisan

WHERE exists (SELECT urun\_id from mart WHERE mart.urun\_id=nisan.urun\_id)

**-- EXISTS , NOT EXISTS --**

**-- EXISTS condition subquerylerde kullanilir**

**-- IN ifadesinin kullanimina benzer**

**-- EXISTS , NOT EXISTS ifadeleri de altsorgudan getirilen degerlerin icerisinde**

**--bir degerin olmasi veya olmamasi durumunda islem yapilmasini saglar**

**-- Exists true false gibidir**

CREATE TABLE mart

(

urun\_id int,

musteri\_isim varchar(50),

urun\_isim varchar(50)

);

CREATE TABLE nisan

(

urun\_id int ,

musteri\_isim varchar(50),

urun\_isim varchar(50)

);

INSERT INTO mart VALUES (10, 'Mark', 'Honda');

INSERT INTO mart VALUES (20, 'John', 'Toyota');

INSERT INTO mart VALUES (30, 'Amy', 'Ford');

INSERT INTO mart VALUES (20, 'Mark', 'Toyota');

INSERT INTO mart VALUES (10, 'Adam', 'Honda');

INSERT INTO mart VALUES (40, 'John', 'Hyundai');

INSERT INTO mart VALUES (20, 'Eddie', 'Toyota');

INSERT INTO nisan VALUES (10, 'Hasan', 'Honda');

INSERT INTO nisan VALUES (10, 'Kemal', 'Honda');

INSERT INTO nisan VALUES (20, 'Ayse', 'Toyota');

INSERT INTO nisan VALUES (50, 'Yasar', 'Volvo');

INSERT INTO nisan VALUES (20, 'Mine', 'Toyota');

select \* from mart

select \* from nisan

-- MART VE NİSAN aylarında aynı URUN\_ID ile satılan ürünlerin

--URUN\_ID’lerini listeleyen ve aynı zamanda bu ürünleri MART ayında alan

--MUSTERI\_ISIM 'lerini listeleyen bir sorgu yazınız.

select urun\_id,musteri\_isim FROM mart

WHERE exists (SELECT urun\_id FROM nisan WHERE mart.urun\_id=nisan.urun\_id)

--Her iki ayda birden satılan ürünlerin URUN\_ISIM'lerini ve bu ürünleri

--NİSAN ayında satın alan MUSTERI\_ISIM'lerini listeleyen bir sorgu yazınız.

SELECT urun\_isim,musteri\_isim from nisan

where exists (select urun\_isim from mart where nisan.urun\_isim=mart.urun\_isim);

--Her iki ayda ortak satilmayan ürünlerin URUN\_ISIM'lerini ve bu ürünleri

--NİSAN ayında satın alan MUSTERI\_ISIM'lerini listeleyen bir sorgu yazınız.

SELECT urun\_isim,musteri\_isim from nisan

where not exists (select urun\_isim from mart where mart.urun\_isim=nisan.urun\_isim);

**--AGGREGATE METHOD KULLANIMI --**

**/\***

**-Aggregate Methodları (SUM(Topla),COUNT(Say),MIN(EN küçük değer),MAX(En büyük değer),AVG(Ortalama))**

**-Subquery içinde de kullanılır**

**-Ancak, sorgu tek bir değer döndürüyor olmalıdır**

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

**\*/**

select \* from calisanlar2

-- Çalısanlar2 tablosundaki en yüksek maaş değerini listeleyiniz

SELECT MAX(maas) from calisanlar2

--Calisanlar tablosundaki maaşların toplamını listeleyiniz

select SUM(maas) from calisanlar2

--Calıcanlar tablosundaki maas ortalamalını listeleyiniz

SELECT AVG(maas) from calisanlar2--(2730.7744)

select round(avg(maas)) from calisanlar2 --noktadan sonrasını(2730)yani ondalık kısmı kaldırır

SELECT round(AVG(maas),2) from calisanlar2--noktadan sonrasını(2730.77)yani ondalık kısmı 2 rakam olarak kalır

----Calisanlar toplasundan en düşük maası listeleyiniz;

select min(maas) from calisanlar2

----Calisanlar tablosundaki kaç kişinin maaş aldığını listeleyiniz;

select count(maas) from calisanlar2

-- Her markanin ismini, calisan sayisini ve o markaya ait calisanlarin toplam maaşini listeleyiniz

select marka\_isim,calisan\_sayisi,

(SELECT SUM(maas) from calisanlar2 WHERE marka\_isim=isyeri)

AS toplam\_maas from markalar

-- Her markanin ismini, calisan sayisini ve o markaya ait calisanlarin maksimum ve minumum maaşini

--listeleyen bir Sorgu yaziniz**. \*\*\*\*\*\*-- tekrar bakılacak!!!..\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

SELECT marka\_isim,calisan\_sayisi,

(select max(maas) from calisanlar2 WHERE marka\_isim=isyeri) AS maximum\_maas,

(select min(maas) from calisanlar2 where marka\_isim=isyeri) AS minumum\_maas

from markalar;

--Her markanin id’sini, ismini ve toplam kaç şehirde bulunduğunu listeleyen bir SORGU yaziniz.

SELECT marka\_id,marka\_isim,

(SELECT count(sehir) from calisanlar2 WHERE marka\_isim=isyeri) AS maxi\_sehir from markalar

--Interview Question: En yüksek ikinci maas değerini çağırın.

SELECT max(maas) AS enYuksek\_ikinci\_maas from calisanlar2

WHERE maas < (select max(maas) from calisanlar2 )

----Interview Question: En düşük ikinci maas değerini çağırın.

SELECT min(maas) AS en\_düşük\_ikinci\_maas from calisanlar2

WHERE maas >(select min(maas) from calisanlar2 ) ;

--En yüksek üçüncü maas değerini bulun

select MAX(maas) AS enYuksek\_ucuncu\_maas from calisanlar2

WHERE maas<(select max(maas) from calisanlar2 WHERE maas< (select max(maas) from calisanlar2))

--En dusuk ucuncu maas degerini bulunuz(ODEV) --> 2500 ?

SELECT min(maas) AS endusuk\_ucuncu\_maas from calisanlar2

WHERE maas> (select min (maas) from calisanlar2 WHERE maas >(SELECT min(maas) from calisanlar2));

CREATE TABLE calisanlar2

(

id int,

isim VARCHAR(50),

sehir VARCHAR(50),

maas int,

isyeri VARCHAR(20)

);

INSERT INTO calisanlar2 VALUES(123456789, 'Ali Seker', 'Istanbul', 2500, 'Vakko');

INSERT INTO calisanlar2 VALUES(234567890, 'Ayse Gul', 'Istanbul', 1500, 'LCWaikiki');

INSERT INTO calisanlar2 VALUES(345678901, 'Veli Yilmaz', 'Ankara', 3000, 'Vakko');

INSERT INTO calisanlar2 VALUES(456789012, 'Veli Yilmaz', 'Izmir', 1000, 'Pierre Cardin');

INSERT INTO calisanlar2 VALUES(567890123, 'Veli Yilmaz', 'Ankara', 7000, 'Adidas');

INSERT INTO calisanlar2 VALUES(456789012, 'Ayse Gul', 'Ankara', 1500, 'Pierre Cardin');

INSERT INTO calisanlar2 VALUES(123456710, 'Fatma Yasa', 'Bursa', 2500, 'Vakko');

CREATE TABLE markalar

(

marka\_id int,

marka\_isim VARCHAR(20),

calisan\_sayisi int

);

INSERT INTO markalar VALUES(100, 'Vakko', 12000);

INSERT INTO markalar VALUES(101, 'Pierre Cardin', 18000);

INSERT INTO markalar VALUES(102, 'Adidas', 10000);

INSERT INTO markalar VALUES(103, 'LCWaikiki', 21000);

select \* from calisanlar2

select \* from markalar

-- calisan sayisi 15.000’den cok olan markalarin isimlerini --->

-- --> ve bu markada calisanlarin isimlerini ve maaşlarini listeleyin.

SELECT isim,maas,isyeri from calisanlar2

WHERE isyeri IN(SELECT marka\_isim from markalar WHERE calisan\_sayisi>15000);

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

SELECT isim,maas,sehir from calisanlar2

where isyeri IN(select marka\_isim from markalar where marka\_id>101)

-- Ankara’da calisani olan markalarin marka id'lerini ve calisan sayilarini listeleyiniz.

select marka\_id,calisan\_sayisi from markalar

where marka\_isim IN(SELECT isyeri from calisanlar2 where sehir='Ankara' )

-- calisan sayisi 3.000’den cok olan markalarin isimlerini --->

-- --> ve bu markada calisanlarin isimlerini ve maaşlarini listeleyin.

select isyeri,isim,maas from calisanlar2

where isyeri IN(select marka\_isim from markalar where calisan\_sayisi>3000);

-- marka\_id’si 101’den kucuk olan marka çalişanlarinin isim, maaş ve şehirlerini listeleyiniz.

select isim,maas,sehir from calisanlar2

where isyeri IN(select marka\_isim from markalar where marka\_id<101);

-- Bursa’da calisani olan markalarin marka id'lerini ve calisan sayilarini listeleyiniz.

select marka\_id,calisan\_sayisi from markalar

where marka\_isim IN(select isyeri from calisanlar2 where sehir='Bursa');

-- İzmir'de calisani olan markalarin marka id'lerini ve calisan sayilarini listeleyiniz.

select marka\_id,calisan\_sayisi from markalar

where marka\_isim IN(select isyeri from calisanlar2 where sehir='Izmir');