**SQL komutları 4 ana gruba ayrılır:**

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

DQL içindeki SELECT komutu ile veri tabanında yer alan mevcut kayıtların bir kısmını veya

tamamını tanımlanan koşullara bağlı olarak alır.

**SELECT :** Veritabanındaki verileri alır.

1. Veri Kullanma Dili (Data Manipulation Language - DML)

DML komutları ile veri tabanlarında 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.

1. 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.

1. 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.

**CONSTRAINT**

**UNIQUE** - Bir sütundaki tüm değerlerin BENZERSİZ/TEKRARSIZ 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 sütünün NULL içermemesini ve sütundaki verilerin

BENZERSİZ olmasını sağlar. (NOT NULL ve UNIQUE)

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

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

PG Adminleri açalım

Sql\_prac

Örnek 1. Önce öğrenciler isimli bir tablo oluşturalım

CREATE TABLE ogrenciler (

    id INTEGER PRIMARY KEY,

    isim VARCHAR(50),

    adres VARCHAR(100),

    notu INTEGER

);

Tablonun içine öğrenci ekleyelim

INSERT INTO ogrenciler (id, isim, adres, notu) VALUES

(1, 'Ahmet Yılmaz', 'Ankara', 80),

(2, 'Burcu Öztürk', 'İstanbul', 90),

(3, 'Cemal Doğan', 'İzmir', 70),

(4, 'Deniz Arslan', 'Adana', 85),

(5, 'Ebru Kaya', 'Bursa', 95),

(6, 'Fatih Demir', 'Antalya', 65),

(7, 'Gizem Tekin', 'Konya', 75);

--Soru 1: Öğrenciler tablosundaki id ve isim sütununu getiriniz

select id,isim from ogrenciler;

--Soru 2: Sınav notu 80 den büyük olan öğrencilerin tüm bilgilerini listele

select \* from ogrenciler where notu>80;

--Soru 3:Adresi Ankara olan öğrencilerin tüm bilgilerini getir

select \* from ogrenciler where adres='Ankara'

--Soru 4: Sınav notu 80 ve adresi Ankara Olan öğrenci ismini getirsin

select isim from ogrenciler where notu='80' and adres='Ankara'

--Soru 5: Sınav notu 65 veya 80 olan öğrencilerin tüm bilgilerini listeleme

select \* from ogrenciler where notu=65 or notu=80

select \* from ogrenciler where notu in (65,80);

--Soru 6: Notu 65 ile 85 arasında olan öğrencilerin tüm bilgilerini listeleyin

select \* from ogrenciler where notu between 65 and 85;

--Soru 7: id si 2 ile 4 arasında olmayan öğrencilerin isim ve notunu listeleyiniz

select id,isim,notu from ogrenciler where id not between 2 and 4;

--Soru 8: notu 80 olan satırı siliniz

delete from ogrenciler where notu=80;

--Soru 9: Adresi Adana olan satırı siliniz

delete from ogrenciler where adres='Adana'

--Soru 10: ismi Burcu Öztürk veya Fatih Demir olanları sil

delete from ogrenciler where isim='Burcu Öztürk' or isim='Fatih Demir'

--Soru 11: notu 80 den küçük olanları siliniz

delete from ogrenciler where notu<80;

--Soru 12: Tablolardaki tüm satırları siliniz

delete from ogrenciler;

**PostgreSQL'de kullanabileceğiniz bazı farklı türde constraint'ler şunlardır:**

Primary Key Constraint: Bir tablodaki bir veya daha fazla sütunu tekil hale getirir. Örnek: CREATE TABLE ogrenciler (id serial PRIMARY KEY,

adi varchar(50),

soyadi varchar(50));

Foreign Key Constraint: İki tablo arasında bir ilişki kurar ve ilişkili tablodaki bir sütunun değerleri, diğer tablodaki belirli bir sütunun değerleriyle eşleşir. Örnek:

CREATE TABLE siniflar (id serial PRIMARY KEY,

adi varchar(50));

CREATE TABLE ogrenciler (id serial PRIMARY KEY,

adi varchar(50), soyadi varchar(50),

sinif\_id int REFERENCES siniflar(id));

Unique Constraint: Belirli bir sütunda bulunan tüm değerleri benzersiz hale getirir. Örnek: CREATE TABLE ogrenciler (id serial PRIMARY KEY,

numara int UNIQUE, adi varchar(50), soyadi varchar(50));

Not Null Constraint: Bir sütunun boş olamayacağını belirtir. Örnek:

CREATE TABLE ogrenciler (id serial PRIMARY KEY,

adi varchar(50) NOT NULL,

soyadi varchar(50) NOT NULL);

Check Constraint: Bir sütunun değerinin belirli bir koşulu karşılaması gerektiğini belirtir. Örnek:

CREATE TABLE ogrenciler (id serial PRIMARY KEY,

adi varchar(50),

yas int CHECK (yas > 0));

Bu sadece birkaç örnek ve PostgreSQL'de kullanabileceğiniz daha birçok constraint türü vardır.

/\*SORU: memurlar isminde bir tablo oluşturunuz. id, isim, maas sutunları olsun.

id sutununun data tipi int olsun. PRIMARY KEY kısıtlaması olsun.

isim sutununun data tipi varchar (30) olsun. UNIQUE kısıtlaması olsun.

maas sutununun data tipi int olsun. maas 5000 buyuk olsun. NOT NULL kısıtlaması olsun.\*/

CREATE TABLE memurlar (

    id int PRIMARY KEY,

    isim varchar(30) UNIQUE,

    maas int NOT NULL CHECK (maas > 5000)

);

* Bu sorunun çözümünden sonra pg admin de tablolarda constraint kısmında kıstlamaları gör

/\*SORU: insanlar isminde bir tablo oluşturunuz. isim, soyisim sutunları olsun.

isim sutununun data tipi int olsun. PRIMARY KEY kısıtlaması olsun. Kısıtlamanın ismi pr\_ks olsun

soyisim sutununun data tipi varchar (30) olsun. UNIQUE kısıtlaması olsun. Kısıtlamanın ismi uni\_ks olsun\*/

CREATE TABLE insanlar (

    isim int CONSTRAINT pr\_ks PRIMARY KEY,

    soyisim varchar(30) CONSTRAINT uni\_ks UNIQUE

);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*1.gün sonu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Ufak bir bilgilendirme

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

**SELECT:** Veri Tabanındaki verileri alır.

1. Veri Kullanma Dili (Data Manipulation Language - DML)

**INSERT :** Veritabanına yeni veri ekler.

**UPDATE :** Veritabanındaki verileri günceller.

**DELETE :** Veritabanındaki verileri siler.

1. Veri Tanimlama Dili (Data Definition Language - DDL)

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

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

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

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

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

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

* **2 tablo create edelim**

CREATE TABLE parent

(

vergi\_no int PRIMARY KEY,

firma\_ismi VARCHAR(50),

irtibat\_ismi VARCHAR(50)

);

INSERT INTO parent VALUES (101, 'IBM', 'Kim Yon');

INSERT INTO parent VALUES (102, 'Huwaei', 'Çin Li');

INSERT INTO parent VALUES (103, 'Erikson', 'Maki Tammen');

INSERT INTO parent VALUES (104, 'Apple', 'Adam Eve');

select \* from parent;

CREATE TABLE child

(

ted\_vergino int,

urun\_id int,

urun\_isim VARCHAR (50),

musteri\_isim VARCHAR(50),

CONSTRAINT fk FOREIGN KEY (ted\_vergino) REFERENCES parent (vergi\_no)

);

INSERT INTO child VALUES (101, 1001, 'Laptop', 'Ayşe Can');

INSERT INTO child VALUES (102, 1002, 'Phone', 'Fatma Aka');

INSERT INTO child VALUES (102, 1003, 'TV', 'Ramazan Öz');

INSERT INTO child VALUES (102, 1004, 'Laptop', 'Veli Han');

INSERT INTO child VALUES (103, 1005, 'Phone', 'Canan Ak');

INSERT INTO child VALUES (104, 1006, 'TV', 'Ali Bak');

INSERT INTO child VALUES (104, 1007, 'Phone', 'Aslan Yılmaz');

select \* from child;

* child deki ted\_vergino foreign key parent taki vergi no ya bağlı
* vergi\_no da bulunmayan bir numara ile child a veri girilemez
* child silinmeden parent silinmez

\*\*\*\*\*\*\*\*Practice 1 Sonu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

--Soru 1: child tablosuna ted\_vergi nosu 101 olan bir veri girişi yapınız

->Önce parentta 101 vergi nosu var mı kontrol etmem lazım

INSERT INTO child VALUES (101, 1008, 'Araba', 'Ali Can');

--Soru 2: child tablosuna ted\_vergnosu 105 olan veri girişi yapınız.

Bu soru çözülemez. Parent tabloda 105 olan bir veri yok. Kod çalışmayacak bir bakalım sonuç ne olacak

insert into child values(105, 1009,’Phone’,’Tech Pro’);

--Soru 3: child tablosuna ted\_vergino null veri girişi olabilir mi

insert into child values(null,1010,’Bisiklet’,’Fatih B’);

istisna olarak null ı kaydediyor childa. Parenta kaydetmeye çalışsanız olmaz

--Soru 4: parent tablosundaki vergi\_no 101 olan veriyi siliniz

delete from parent where vergi\_n0=101;

silinemez

--Soru 5:child tablosunda ted\_vergino 101 olan ürünleri siliniz.

delete from child where ted\_vergino=101;

--Soru 6: parent tablosundaki vergi\_no 101 olan veriyi siliniz

delete from parent where vergi\_no=101;

--Soru 7: parent tablosunu siliniz.

child tablosunu silmeden parent silinmez

--Soru 8: child tablosunu siliniz

drop table child;

--Soru 9: parent tabloyu siliniz

drop table parent;

\* Tekrar tablo üretelim parent2 ve child 2 olsun

CREATE TABLE parent2

(

vergi\_no int PRIMARY KEY,

firma\_ismi VARCHAR(50),

irtibat\_ismi VARCHAR(50)

);

INSERT INTO parent2 VALUES (101, 'IBM', 'Kim Yon');

INSERT INTO parent2 VALUES (102, 'Huwaei', 'Çin Li');

INSERT INTO parent2 VALUES (103, 'Erikson', 'Maki Tammen');

INSERT INTO parent2 VALUES (104, 'Apple', 'Adam Eve');

select \* from parent2;

CREATE TABLE child2

(

ted\_vergino int,

urun\_id int,

urun\_isim VARCHAR (50),

musteri\_isim VARCHAR(50),

CONSTRAINT fk FOREIGN KEY (ted\_vergino) REFERENCES parent2 (vergi\_no)

ON DELETE CASCADE

);

INSERT INTO child2 VALUES (101, 1001, 'Laptop', 'Ayşe Can');

INSERT INTO child2 VALUES (102, 1002, 'Phone', 'Fatma Aka');

INSERT INTO child2 VALUES (102, 1003, 'TV', 'Ramazan Öz');

INSERT INTO child2 VALUES (102, 1004, 'Laptop', 'Veli Han');

INSERT INTO child2 VALUES (103, 1005, 'Phone', 'Canan Ak');

INSERT INTO child2 VALUES (104, 1006, 'TV', 'Ali Bak');

INSERT INTO child2 VALUES (104, 1007, 'Phone', 'Aslan Yılmaz');

select \* from child2;

ON DELETE CASCADE ile bağımsız şekilde silebiliyoruz. Child daki verileri silmeden parenttakileri silebiliyoruz, fakat parentta silme işlemi yapınca child taki bağlı veriler de siliniyor

--Soru 1: parent2 tablosundaki tüm verileri siliniz

delete from parent2;

child tablodaki veriyi silmeden parentı silebildik on delete cascade sayesinde

parent2 ve child tablolarına tekrar bak duruyormu içi

--Soru 2:parent2 tablosunu siliniz

drop table parent2 cascade;

parent2 silindi ama child2 duruyor.

--Soru 3 : child tablosunu sil

drop table child2;

Biraz da update çalışması yapalım. Yeni 2 tablo veriyorum

CREATE TABLE toptancilar

(

vergi\_no int PRIMARY KEY,

sirket\_ismi VARCHAR(40),

irtibat\_ismi VARCHAR(30)

);

INSERT INTO toptancilar VALUES (201, 'IBM', 'Kadir Şen');

INSERT INTO toptancilar VALUES (202, 'Huawei', 'Çetin Hoş');

INSERT INTO toptancilar VALUES (203,'Erikson', 'Mehmet Gör');

INSERT INTO toptancilar VALUES (204, 'Apple', 'Adem Coş');

CREATE TABLE malzemeler --> child

(

ted\_vergino int,

malzeme\_id int,

malzeme\_isim VARCHAR(20),

musteri\_isim VARCHAR(25),

CONSTRAINT FK FOREIGN KEY (ted\_vergino) REFERENCES toptancilar (vergi\_no) on delete cascade

);

INSERT INTO malzemeler VALUES (201, 1001, 'Laptop', 'Asli Can');

INSERT INTO malzemeler VALUES (202, 1002, 'Telefon', 'Fatih Ak');

INSERT INTO malzemeler VALUES (202, 1003, 'TV', 'Ramiz Özmen');

INSERT INTO malzemeler VALUES (202, 1004, 'Laptop', 'Veli Tan');

Soru 1: vergi\_no su 202 olan toptancını şirket\_ismini ‘VivoBook’ olarak güncelleyin

update toptancilar set sirket\_ismi='VivoBook' where vergi\_no=202;

Soru 2: toptancilar tablosundaki tüm şirket isimlerini ‘NOKIA’ olarak güncelleyiniz

update toptancilar set sirket\_ismi='NOKIA';

3.bölüm practice

"ON DELETE CASCADE" kullanmanın avantajları şunlardır:

1. Veritabanı bütünlüğünü sağlar: "ON DELETE CASCADE", bağımlı tablolarda bir kayıt silindiğinde, bağımlı tabloların diğer kayıtlarını da otomatik olarak silerek, veritabanı bütünlüğünü sağlar. Bu, veri bütünlüğünü koruyarak veritabanı hatalarını önlemeye yardımcı olur.
2. Veritabanı tasarımı basitleştirir: "ON DELETE CASCADE" kullanmak, veritabanı tasarımını basitleştirebilir. Birçok bağımlı tablo varsa, her bir tablonun kayıtlarını tek tek silmek yerine, ana tabloda bir kayıt silinmesi yeterlidir.

Ancak "ON DELETE CASCADE" kullanmanın bazı dezavantajları da vardır:

1. Beklenmeyen sonuçlar: "ON DELETE CASCADE" kullanarak veri kayıplarına yol açabilecek beklenmeyen sonuçlarla karşılaşabilirsiniz. Örneğin, yanlışlıkla bir kaydı sildiğinizde, bu kayda bağlı olan diğer tüm kayıtlar da silinebilir.
2. Performans etkisi: "ON DELETE CASCADE" kullanmanın bir dezavantajı, performansı etkileyebilecek olmasıdır. Özellikle büyük veritabanlarında, silme işlemi çok zaman alabilir ve veritabanı performansını düşürebilir.
3. İşlem geri alma sorunları: "ON DELETE CASCADE" kullanarak silinen kayıtların geri alınması zordur. Bağımlı tablolardaki kayıtların tümü silindiğinden, bu kayıtların geri yüklenmesi mümkün olmayabilir.

Bu nedenle, "ON DELETE CASCADE" kullanmadan önce, veritabanınızı dikkatli bir şekilde tasarlamalı ve bu yöntemi kullanmadan önce avantaj ve dezavantajlarını iyice düşünmelisiniz. Ayrıca, verilerinizi yedekleyerek, beklenmedik sonuçlara karşı kendinizi korumanız önerilir.

Nerede kullanmak faydalı olur

Örneğin, bir "Customers" tablosu ve bu tabloya bağlı olan "Orders" tablosu varsa, "Customers" tablosundaki bir müşteri kaydı silindiğinde, "Orders" tablosundaki ilgili tüm siparişlerin de silinmesi istenebilir.

bir "Departments" tablosu ve bu tabloya bağlı olan "Employees" tablosu da örnek olarak verilebilir. "Departments" tablosundaki bir departman kaydı silindiğinde, "Employees" tablosundaki o departmana ait tüm çalışan kayıtlarının da silinmesi istenebilir.

--Soru 3: vergi\_no su 201 olan toptancının şirket ismini nokia ve irtibat\_ismini Canan Can olarak güncelleyiniz

update toptancilar set sirket\_ismi='nokia',irtibat\_ismi='Canan Can' where vergi\_no=201;

--Soru 4: toptancılar tablosunda Şirket ismi nokia olan firmanın irtibat ismini Bilal Ayhan diye güncelle

update toptancilar set irtibat\_ismi='Bilal Ayhan' where sirket\_ismi='nokia';

--Soru 5 : malzemeler tablosundaki telefon değerlerini phone olarak güncelleyiniz

update malzemeler set malzeme\_isim='Phone' where malzeme\_isim='Telefon';

--Soru 6: Malzemeler tablosundaki malzeme\_id değeri 1002 den büyük olanların malzeme id lerini 1 arttır

update malzemeler set malzeme\_id=malzeme\_id+1 where malzeme\_id>1002;

--Soru 7: malzemeler tablosundaki tüm malzemelerin malzeme\_id değerlerini tedarik vergi no ile toplayarak güncelleyiniz

update malzemeler set malzeme\_id=malzeme\_id+ted\_vergino;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*2.Gün Sonu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tablolarımızı silip tekrar tabloları oluşturalım.

CREATE TABLE toptancilar

(

vergi\_no int PRIMARY KEY,

sirket\_ismi VARCHAR(40),

irtibat\_ismi VARCHAR(30)

);

INSERT INTO toptancilar VALUES (201, 'IBM', 'Kadir Şen');

INSERT INTO toptancilar VALUES (202, 'Huawei', 'Çetin Hoş');

INSERT INTO toptancilar VALUES (203,'Erikson', 'Mehmet Gör');

INSERT INTO toptancilar VALUES (204, 'Apple', 'Adem Coş');

CREATE TABLE malzemeler --> child

(

ted\_vergino int,

malzeme\_id int,

malzeme\_isim VARCHAR(20),

musteri\_isim VARCHAR(25),

CONSTRAINT FK FOREIGN KEY (ted\_vergino) REFERENCES toptancilar (vergi\_no) on delete cascade

);

INSERT INTO malzemeler VALUES (201, 1001, 'Laptop', 'Asli Can');

INSERT INTO malzemeler VALUES (202, 1002, 'Telefon', 'Fatih Ak');

INSERT INTO malzemeler VALUES (202, 1003, 'TV', 'Ramiz Özmen');

INSERT INTO malzemeler VALUES (202, 1004, 'Laptop', 'Veli Tan');

SORU: Malzemeler tablosundaki musteri\_isim'i Ali Can olan malzeme\_isim'ini, --toptancılar

tablosunda irtibat\_ismi 'Adem Coş' olan sirket\_ismi ile güncelleyiniz.

1.Adım ayrı satırlarda düşünelim

update malzemeler set malzeme\_isim=()

where musteri\_isim=’Ali Can’

select sirket\_ismi from toptancilar where irtibat\_ismi=’Adem Coş’

sonra en alttaki satırı parantez içine al

update malzemeler set malzeme\_isim=( select sirket\_ismi from toptancilar where irtibat\_ismi=’Adem Coş’)

where musteri\_isim=’Ali Can’;

SORU: malzeme\_ismi Laptop olan musteri\_isim'ini, sirket\_ismi Apple'in irtibat\_isim'i ile güncelleyiniz.

update malzemeler

set musteri\_isim= ( select irtibat\_ismi from toptancilar where sirket\_ismi='Apple') where malzeme\_isim= 'Laptop';

select irtibat\_ismi from toptancılar where sirket\_ismi='Apple'

create table arac (

id int,

marka varchar (30),

model varchar(30),

fiyat int,

kilometre int,

vites varchar(20)

)

insert into arac values (100, 'Citroen', 'C3', 500000, 5000, 'Otomatik' );

insert into arac values (101, 'Mercedes', 'C180', 900000, 10000, 'Otomatik' );

insert into arac values (102, 'Honda', 'Civic', 400000, 15000, 'Manuel' );

insert into arac values (103, 'Wolkswagen', 'Golf', 350000, 20000, 'Manuel' );

insert into arac values (104, 'Ford', 'Mustang', 750000, 5000, 'Otomatik' );

insert into arac values (105, 'Porsche', 'Panamera', 850000, 5000, 'Otomatik' );

insert into arac values (106, 'Bugatti', 'Veyron', 950008, 5000, 'Otomatik' );

**AGGREGATE FONKSİYON**

birden çok değer üzerinde bir hesaplama gerçekleştirir

ve tek bir değer döndürür.

SQL; avg, count, sum, min, max, vb. içeren birçok toplama işlevi sağlar.

--SORU1: arac tablosundaki en yüksek fiyat'ı listele

Select max(fiyat) from arac;

--Soru 2:arac tablosundaki araçları toplam fiyatını bulunuz

Select sum(fiyat) from arac;

--Soru 3:arac tablosundaki fiyat ortalamalarını bulunuz

Select avg(fiyat) from arac;

Select round(avg(fiyat)) from arac;

Select round(avg(fiyat),2) from arac;

--Soru 4:arac tablosunda kaç tane araç olduğunu bulunuz

Select count(id) from arac

Yeni bir tablo oluşturalım

CREATE TABLE meslekler

(

id int PRIMARY KEY,

isim VARCHAR(50),

soyisim VARCHAR(50),

meslek CHAR(9),

maas int

);

INSERT INTO meslekler VALUES (1, 'Ali', 'Can', 'Doktor', '20000' );

INSERT INTO meslekler VALUES (2, 'Veli', 'Cem', 'Mühendis', '18000');

INSERT INTO meslekler VALUES (3, 'Mine', 'Bulut', 'Avukat', '17008');

INSERT INTO meslekler VALUES (4, 'Mahmut', 'Bulut', 'Ögretmen', '15000');

INSERT INTO meslekler VALUES (5, 'Mine', 'Yasa', 'Teknisyen', '13008');

INSERT INTO meslekler VALUES (6, 'Veli', 'Yilmaz', 'Hemşire', '12000');

INSERT INTO meslekler VALUES (7, 'Ali', 'Can', 'Marangoz', '10000' );

INSERT INTO meslekler VALUES (8, 'Veli', 'Cem', 'Tekniker', '14000');

--Soru 1: meslekler tablosunu isime göre sıralayınız

Select \* from meslekler order by isim;

--Soru 2: meslekler tablosunda maaşı büyükten küçüğe doğru sıralayınız

Select \* from meslekler order by maas desc;

--Soru 3: meslekler tablosunda ismi Ali olanları maaşı büyükten küçüğe doğru sıralayınız

Slect \* from meslekler where isim=’Ali’ order by maas desc;

--Soru 4: meslekler tablosunda soyisimi Bulut olanları, maaşı sıralı olarak sorgulayınız

Select \* from meslekler where soyisim=’Bulut’ order by maas;

--Soru 5: meslekler tablosunda id değeri 5 ten büyük olan ilk iki veriyi listeleyiniz

Select \* from meslekler where id>5 limit2;

--Soru 6: meslekler tablosunda maaşı en yüksek 3 kişinin bilgilerini getiriniz

Select \* from meslekler order by maas desc limit 3;

(offset leri görmemişler)

--Soru 7: meslekler tablosunda ilk 2 den sonra 5 veriyi getiriniz

Select \* from meslekler offset 2 limit 5;

--Soru 8 : meslekler tablosunda en yüksek maaşı alan 4,5,6. Kişilerin bilgilerini getiriniz

Select \* from meslekler order by maas desc offset 3 limit 3;

INSERT INTO meslekler VALUES (9, 'Veli', 'Cem', NULL, '');

INSERT INTO meslekler VALUES (10, 'Görür', 'Görmez', NULL, '');

-Soru 9: meslekler tablosunda mesleği null olanları getiriniz

Select \* from meslekler where meslek is null;

--Soru 10: meslekler tablosundan mesleği null olmayanları getiriniz

Select \* from meslekler where meslek is not null;

\*\*\*\*\*\*Practice 3 Sonu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Join lere geçiyoruz 2 tabloyu yeniden oluşturuyoruz

Left join : Left join işlemi, ilk tablodaki tüm kayıtları ve ikinci tablodaki eşleşen kayıtları getirir. Eğer ikinci tabloda eşleşen kayıt yoksa, sonuç olarak NULL değeri döndürür.

Right join : Right join işlemi, ilk tablodaki tüm kayıtları ve ikinci tablodaki eşleşen kayıtları getirir. Eğer ikinci tabloda eşleşen kayıt yoksa, sonuç olarak NULL değeri döndürür.

İnner join : Inner join işlemi, her iki tablodaki eşleşen kayıtları getirir ve eşleşmeyen kayıtları dışarıda bırakır.

Full Join : Full join işlemi, her iki tablodaki tüm kayıtları getirir ve eşleşen kayıtları birleştirir. Eğer bir tabloda eşleşen kayıt yoksa, o tablodaki diğer kayıtlar NULL değeri ile gösterilir.

Self Join : Aynı tablo içinde farklı satırları birleştirmek için kullanılan bir join işlemidir.

2 tablo create edelim

create table filmler (

film\_id int,

film\_name varchar(30),

category varchar(30) );

insert into filmler values (1, 'Eyvah Eyvah', 'Komedi');

insert into filmler values (2, 'Kurtlar Vadisi', 'Aksiyon');

insert into filmler values (3, 'Eltilerin Savasi', 'Komedi');

insert into filmler values (4, 'Aile Arasinda', 'Komedi');

insert into filmler values (5, 'GORA', 'Bilim Kurgu');

insert into filmler values (6, 'Organize Isler', 'Komedi');

insert into filmler values (7, 'Babam ve Oglum', 'Dram');

create table aktorler (

id int,

actor\_name varchar(30),

film\_id int

);

insert into aktorler values (1, 'Ata Demirer', 1);

insert into aktorler values (2, 'Necati Sasmaz', 2);

insert into aktorler values (3, 'Gupse Ozay', 3);

insert into aktorler values (4, 'Engin Gunaydin', 4);

insert into aktorler values (5, 'Cem Yilmaz', 5);

\*\*\*2 ayrı sayfada açıp anlat

--Soru1 Tüm film\_namel leri, kategorilerini ve filmlerde oynayan aktor\_name leri listeleyin

\*\*\*tüm film name dediği için tüm film name leri olacak

Select film\_name,category,actor\_name from filmler as A

left join(esas tablo sol da kalıyor

İster sol ister sağ seçebilirim) aktorler as B

(her iki tabloya bağlamam lazım)

on A.film\_id=B.film\_id;

bunu çalıştırmadan çizerek anlatabilirsin

id si olmayan yere aktör kısmını null veriyor

istersek aynı soruyu right ile de çözebiliriz

bir şey farketmez

2.yol olarak ta çözelim-esas tablo filmler joinin riht tarafında olacak

Select film\_name,category,actor\_name from aktorler as B

right join filmler as A

on A.film\_id=B.film\_id;

aynı sonuç olacak

--Soru 2: tüm actor\_name leri ve film name leri listeleyiniz

Select actor\_name,film\_name from aktorler as A

Left join filmler as B

on A.film\_id=B.film\_id;

bunu da çizerek göster null değeri yok

birde riht joinle yapalım bu d 2.yol olsu

Select actor\_name, film\_name from filmler as B

right join aktorler as A

on A.film\_id=B.film\_id;

Yeni tablolar atalım

CREATE TABLE Customers (

customer\_id INT PRIMARY KEY,

name VARCHAR(50),

email VARCHAR(50)

);

INSERT INTO Customers (customer\_id, name, email)

VALUES (1, 'Ali', 'ali@mail.com'),

(2, 'Ayşe', 'ayse@mail.com'),

(3, 'Fatih', 'fatih@mail.com');

CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

customer\_id INT,

product VARCHAR(50),

price DECIMAL(8,2),

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

INSERT INTO Orders (order\_id, customer\_id, product, price)

VALUES (101, 1, 'Laptop', 2500.00),

(102, 1, 'Printer', 500.00),

(103, 2, 'Tablet', 1200.00),

(104, 3, 'Phone', 1500.00),

(105, 3, 'Camera', 2000.00);

--Soru customer name o isme ait product ve price ı inner join kullanarak birleştiriniz

SELECT name, product, price

FROM Customers as A

INNER JOIN Orders as B ON A.customer\_id = B.customer\_id;

Yeni bir tablo create edelim

CREATE TABLE Employees (

employee\_id INT PRIMARY KEY,

name VARCHAR(50),

department VARCHAR(50),

manager\_id INT

);

INSERT INTO Employees (employee\_id, name, department, manager\_id)

VALUES (1, 'Ali', 'IT', 2),

(2, 'Ayşe', 'IT', NULL),

(3, 'Fatih', 'Marketing', 4),

(4, 'Mehmet', 'Marketing', NULL),

(5, 'Elif', 'DEV', 1);

Soru : inner join kullanarak emloyee\_name ve manager\_name leri gösteriniz

SELECT A.name AS employee\_name, B.name AS manager\_name

FROM Employees as A

INNER JOIN Employees as B ON A.manager\_id = B.employee\_id;

Orders ın içini güncelleyelim

delete from Orders

select \* from Orders

INSERT INTO Orders (order\_id, customer\_id, product, price)

VALUES (101, 1, 'Laptop', 2500.00),

(102, 1, 'Printer', 500.00),

(103, 2, 'Tablet', 1200.00),

(104, NULL, 'Phone', 1500.00),

(105, NULL, 'Camera', 2000.00);

--Soru : customer name product ve price ların hepsini getiriniz

Full join kullan

SELECT A.name, B.product, B.price

FROM Customers A

FULL JOIN Orders B ON A.customer\_id = B.customer\_id;

\*Like

Yeni bir tablo create edelim

CREATE TABLE ogrenciler

(

id int,

isim varchar(50),

sehir varchar(50),

puan int,

bolum varchar(20)

);

​

INSERT INTO ogrenciler VALUES(100, 'Ahmet Ümit', 'İstanbul', 546, 'Bilgisayar Müh.');

INSERT INTO ogrenciler VALUES(101, 'R.Nuri Güntekin', 'Antalya', 486, 'İşletme');

INSERT INTO ogrenciler VALUES(102, 'S.Faik Abasıyanık', 'Bursa', 554, 'Bilgisayar Müh.');

INSERT INTO ogrenciler VALUES(103, 'Yaşar Kemal', 'İstanbul', 501, 'Matematik');

INSERT INTO ogrenciler VALUES(104, 'Halide E. Adıvar', 'İzmir', 473, 'Psikoloji');

INSERT INTO ogrenciler VALUES(105, 'Nazan Bekiroğlu', 'İzmir', 432, 'Edebiyat');

INSERT INTO ogrenciler VALUES(106, 'Peyami Safa', 'Antalya', 535, 'Bilgisayar Müh.');

INSERT INTO ogrenciler VALUES(107, 'Sabahattin Ali', 'İstanbul', 492, 'Matematik');

--Soru 1 : İsimleri 'A' harfi ile başlayan öğrencilerin bilgilerini getirin.

SELECT \* FROM ogrenciler WHERE isim LIKE 'A%';

--Soru 2:İsimleri 'i' harfi ile biten öğrencilerin bilgilerini getirin.

SELECT \* FROM ogrenciler WHERE isim LIKE '%i';

--Soru 3:İsimleri 'A' harfi ile başlayan ve 'Bilgisayar' bölümünde okuyan öğrencilerin bilgilerini getirin.

SELECT \* FROM ogrenciler WHERE isim LIKE 'A%'

AND bolum = 'Bilgisayar Müh.';

--Soru 4: İsimleri 'n' harfi içeren öğrencilerin bilgilerini getirin.

SELECT \* FROM ogrenciler WHERE isim LIKE '%n%';

--Soru 5 :İsimleri 'A' harfi ile başlamayan öğrencilerin bilgilerini getirin.

SELECT \* FROM ogrenciler WHERE isim NOT LIKE 'A%';

--Soru 6 : İsimleri 5 karakterden daha uzun olan öğrencilerin bilgilerini getirin.

SELECT \* FROM ogrenciler WHERE LENGTH(isim) > 5;

--Soru 7:4.harfi a olan öğrenci bilgilerini getirin

SELECT \* FROM ogrenciler WHERE isim LIKE '\_\_\_a%';

--Soru 8: İsminin içinde z veya k harfi geçen ogrenci bilgilerini getirin

SELECT \* FROM ogrenciler WHERE isim ~ '[zk]';

~alt ü enter

--Soru 9: ismi Y veya N ile başlayan ogrenci bilgilerini getirin

SELECT \* FROM ogrenciler WHERE isim ~ '^[YN]';

^shift 3 sonra space