***Day 1- 2***

--Personel isminde bir tablo oluşturalım

create table personel(

pers\_id int,

isim varchar(30),

sehir varchar(30),

maas int,

sirket varchar(20),

adres varchar(50)

);

--Varolan personel tablosundan pers\_id,sehir,adres fieldlarına sahip personel\_adres adında yeni tablo olusturalım

create table personel\_adres

as

select pers\_id, sehir, adres from personel;

--Tabloyu gormek icin

select \* from personel

**DML 🡪 Data Manupulation Language**

INSERT UPDATE DELETE

-- Tabloya veri ekleme, tablodan veri güncelleme ve silme işlemlerinde kullanılan komutlardir.

INSERT into student VALUES ('1001','Ali Can',25);

--Tabloya parçalı veri ekleme

insert into student(st\_name,age) values ('Murat Can',65);

DELETE FROM tablo\_adi --> Tablo'nun tüm içeriğini siler

-- Veriyi seçerek silmek için WHERE koşulu kullanılır

DELETE FROM tablo\_adı WHERE sutun\_adi = veri

Ogrenegin id nosu 124 olan ogrenciyi silmek icin;

DELETE FROM ogrenciler WHERE id = 124;

--Iki yada daha fazla silmek icin ‘or’ kullanilir.

delete from ogrenciler WHERE isim='Ali Can' or id=123;

-- Tablodaki tüm verileri silelim

Delete from ogrenciler

--Truncate komutuda tum verileri siler fakat parcali slime islemi yapamaz.

TRUNCATE TABLE ogrenciler

**DQL 🡪 Data Query Language**

--SELECT ile student tablosunun tamamini gormek icin.

select \* from student;

--Student tablosundaki st\_name column unu (sutun)unu gormek icin:

select st\_name from student;

--SELECT komutu WHERE kosulu;

Tablodaki 35 yasindan buyuk olanlarin tum verilerini gorme icin;

select \* from student WHERE age>35;

**TCL 🡪 Transaction Control Language.**

--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,

--Erasmus programiyla okulumuza yeni ogrenciler eklenip bir yil sonra ogrencilerin cikartilmasi gibi;

Begin; -- ile basliyoruz

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

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

savepoint x; --ile bundan onceki verileri kaydediyoruz

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; --ile ‘savepoint x’ ile belirttigimiz noktaya geri donuyoruz.

commit; --ile islemi sonlandiriyoruz.

--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.

*Ornegin;* 1 ve 2. Veriyi girip 3,4 ile yeni veri ekleriz ve rollback ile kaldirdigimiz 2v4 den sonra ekleme yaptigimizda 5,6 dan devam eder, buyuzden tavsiye edilmez.

-- 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 savepoint adi' kullanırız ve bu savepointe dönebilmek için 'ROLLBACK TO savepoint adi'

-- 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.

**DDL - Data Definition Language**

CREATE –en ustte ilk olarak table olusturmustuk.

-- ALTER TABLE tabloda ADD, TYPE, SET, RENAME veya DROP COLUMNS işlemleri için kullanılır

​

-- Personel tablosuna cinsiyet Varchar(20) ve yas int seklinde yeni sutunlar ekleyiniz.

alter table personel add cinsiyet varchar(20), add yas int;

​

--Personel tablosundan sirket field'ini siliniz

alter TABLE personel drop column sirket;

​

-- Personel tablosundaki sehir sutununun adını ulke olarak değiştirelim

alter TABLE personel RENAME COLUMN sehir to ulke;

​

-- Personel tablosunun adını isciler olarak değiştiriniz

alter table personel rename to isciler;

​

--Isciler tablosunun tamamini silemek icin;

DROP table isciler;

​

**CONSTRAINT (Kısıtlamalar)**

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

b-) FOREIGN 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.

{Bu komutlar tablo olusturulurken ilgili sutundan sonra belirtilir}

​

CREATE TABLE calisanlar --(Parent Tablo)

(

id CHAR(5) PRIMARY KEY, -- not null + unique

isim VARCHAR(50) UNIQUE,

maas int NOT NULL,

ise\_baslama DATE

);

​

CREATE TABLE calisanlar2(

id CHAR(5),

isim VARCHAR(50),

maas int NOT NULL,

ise\_baslama DATE,

CONSTRAINT pk\_id PRIMARY KEY(id),

CONSTRAINT ism\_unq UNIQUE(isim)

);

​​

b-) FOREIGN KEY (Child Tablo)

CREATE TABLE adresler ( --(Child Tablo)

adres\_id char(5),

sokak varchar(20),

cadde varchar(30),

sehir varchar(20),

CONSTRAINT id\_fk FOREIGN KEY (adres\_id) REFERENCES calisanlar2(id)

);

--adresler tablosunda son satirdaki komut, calisanlar2 tablosundaki id sutunu ile baglanti kuruluyor.

--Primary key ile tablo bir nevi Parent-Child iliskisindeki Parent haline geliyor ve dolayisiyla Foreign key ilede tablo Child tablo haline geliyor.

Parent tabloda olmayan id ile child tabloda ekleme yapilamaz ve Parent tabloyu silmemiz icin once Child tablo silinmesi gerekir.