**Veri Nedir?**

Ölçüm,sayım gözlem veya araştırma sonucu elde edilen bilgiler.

**Veri Tabanı Nedir?**

Verileri organize şekilde depolandığı ve modifiye edilebildiği sistemler.

**Neden veri tabanı kullanırız?**

Tutarlı veriler elde etmek, çoklu kullanıcı ve diğer yazılımlarla daha uyumlu çalışması.

**DBMS Nedir?**

Verileri organize bir şekilde saklayabileceğimiz veri tabanlarını oluşturmamıza ve yönetebilmemize imkan sağlayan araçlara DBMS(Data Base Management System)denir.

**SQL Nedir?**

Yapılandırılmış sorgu dili olarak tercüme edilen SQL, veri tabanı işlemlerini gerçekleştirirken tablolarla etkileşime geçmemizi sağlayan declarative programlama dildir.

**PSQL Nedir?**

Psql postgresql’in terminal ekranıdır. PSql’i kullanabilmek için, cmd ile cd C:\Program Files\PostgreSQL\15\bin komutu ile klasörüne gidip orada psql komutunu kullnarak psql’i çalıştırabiliriz. Ancak bu şekilde işletim sistemini kullanırken giriş yaptığımız kullanıcı ile veri tabanına bağlanmaya çalışacaktır, bunun yerine farklı bir kullanıcı ile giriş yapmak için psql –U ‘kullanıcıAdı’ şeklinde kullanılması gerekir. Örn: psql –U postgres şeklinde, ardından bizden parola isteyecektir, bu işletim sistemi kullanıcısı için mevcut bilgisayarı açarken kullandığımız paroladır, diğer kullanıcıların parolalarını uygun şekilde girmek gerekir. **psqlden çıkmak için ctrl+c**

**PSQL’e komut ekranında dosya yolu olmadan erişebilmek için ayarlar**

İlk adım olarak, başlat>ortam değişkenlerini düzenleyin ayarı açılır.

Daha sonra açılan pencereden > ortam değişkenleri seçeneği seçilir.

Gelen ekranda alt kısımda sistem değişkenleri başlığı altında bulunan Path değişkeni seçilerek düzenle butonuna basılır.  
gelen ekrandan sağ tarafta bulunan yeni butonuna tıklanır ve listenin en altında oluşan boş satıra psql’in dosya yolu yapıştırıldıktan sonra tem pencereler tamam butonuna basılarak kapatılır.

**PostgreSql import Database**

Pgadmin arayüzünde sol tarafta PostgrSQL ‘sürüm’ üzerinde sağ tıklanır, ardından create>database seçilir. Ve database adı belirlenir. Ve database oluşturulur, ardından oluşturulan database üzerinde sağ tık yapıp>restore seçeneği seçilir. File name kısmından select file kısmı seçilir. Böylece veri tabanını eklemiş oluruz.

**NOT:** Eğer database i oluşturduktan sonra restore etmeye çalışınca “Utility file not found. Please correct the Binary Path in the Preferences dialog postgresql error” Hatasını alırsanız, pgadmin arayüzünde sağ üst köşeden file>preferences> ardından sol tarafta bulunan Binary paths seçildikten sonra aşağı kaydırıp PostgreSQL Binary Path başlığı altında ilgili postgresql sürümünün binary path kısmına, postgresql’in bilgisayarınızda kurulu olduğu yolu örn: C:\Program Files\PostgreSQL\15\bin şeklinde kopyalayıp yapıştırdıktan sonra kaydederseniz sorun düzelir.

**BETWEEN :**

Between kullanım örneği:

SELECT title FROM film

WHERE length BETWEEN 90 AND 120 // 90 ile 120 arasındaki değerler için kullanılır, 90 ve 120 dışındaki değerler için ise NOT BETWEEN yapısı kullanılır.  
**IN:**

SELECT title FROM film

WHERE length IN(40, 50, 60) **//buradaki kullanımda IN bir dizi olarak düşünülebilir ve dizinin elemanlarından herhangi birine eşit olan değerleri getirmeye yarar, belirli aralık haricinde spesifik olarak değer aradığımız durumlarda kullanılabilir.**

LIKE:

SELECT \* FROM customer

WHERE first\_name LIKE ‘Ma%’

Buradaki % ya herhangi bir karakter olmadığı, ya da birden fazla karakter için yer tutucu anlamına gelir. % nin konumunun bir önemi yoktur başta, sonda veya ortada olabileceği gibi, önünde veya arkasında harf bulunabilir.

SELECT \* FROM customer

WHERE first\_name NOT LIKE ‘Ma%’

Bu kullanım da yukarıdakinin tam tersi anlamdadır yukarıdaki Mai ilke başlayanları, NOT LIKE ise Ma ile başlamayanları getirir.

SELECT \* FROM customer

WHERE first\_name LIKE ‘T\_m’

Alt çizgili LIKE kullanımı, alt çizgi tek bir karakteri temsel ede yani bu sorguda T ile başlayıp m ile biten ve T ile m arasında yalnızca tek bir karakter bulunan kayıtları getirir.

~~ 2 adet Tilda işareti de LIKE yerine geçer.

ILIKE:

LIKE’ın büyük küçük harf duyarsız halidei, LIKE sorgusunda belirttiğimiz harfin büyük küçük olmasına göre dönüt alırken, ILIKE sorgusunda harfin BÜYÜK veya küçük olmasının bir önemi yoktur.

~~\* ILIKE yerine geçer.

**DISTINCT:**

Tablodaki sütunlardan benzersiz(tekrar eden veriler veriler 1 kez görülecek) şekilde görüntülemeye yarar.

SELECT DISTINCT replacement\_cost FROM film.

**COUNT:**

İlgili sütundaki veri sayısını sayar(satır sayısı)

SELECT COUNT(\*) FROM actor

WHERE first\_name=’Penelope’

İsmi first\_name sütunu Penelope olan değer sayısını getirdi.

**PSQL KOMUTLARI:**

**\h**

sql komutları

**\?**

Psql komutları hakkında bilgi verir.

**\list Veya \l:**

Veri tabanlarının bir listesini getirir.

**\connect ‘veritabanıAdı’:**

Veri tabanına bağlanmak için kullanılan komut.

**\dt:**

Bağlandığımız veri tabanındaki tabloları görmemizi sağlar.

**\d:**

Tablo detaylarını görmeye yarayan komut.

**PSQL’de Sorgu yazımı:**

Terminalde sorgu yazarken, sorgunun sonuna “;” işareti koymamız gerekir, Eğer koymazsak enter’a bastığımızda sorgu yazmaya devam edecekmişiz gibi alt satıra gidip oradan devam eder. Bu aynı zamanda sorgumuzu tek satırda değilde birden çok satırda yazmamıza olanak sağlar ve okunurluğu artırır.

**Sorgu yazımı:**

Sorgular pgadmin’de yazıldı şekilde yazılı.

**\q:**

Psql terminalinden çıkmak için kullanılır.

**ORDER BY:**

SELECT \* FROM film

ORDER BY title

Not: eğer sıralanacak sütundan sonra DESC yazmazsak küçükten büyüğe sıralama yapar(ASC), eğer DESC yazarsak büyükten küçüğe göre sıralar. **Birden fazla sütunda farklı sıralamalar yapılabilir.**

SELECT title, rental\_rate, length FROM film

ORDER BY rental\_rate ASC, length DESC

**LIMIT:**

Çok fazla veri bulunan tablolarda sorgu yazarken koşula uyan tüm veriler değil de belirli bir sayıda verinin gelmesini istediğimiz durumlarda kullanılabilir.

SLECT \* FROM film

WHERE rental\_rate = 4.99

ORDER BY length

LIMIT 10;

**OFFSET:**

Ofsett ile belirli sayıda veriyi geçmek için kullanılır, örneğin ülke tablosundaki verileri listelerken baştan 6. Veriden başlayıp 4 veri gösteren sorgu(6,7,8,9,10.) satırdaki veriler:

SELECT \* FROM country

OFFSET 6

LIMIT 4;

**Not: offset pagination işlemi için kullanılır, örneğin ilk sayfada 5 veri gösterildi ikinci sayfada offset kullanılarak ilk 5 veri geçilir ve sonraki 5 veri gösterilir.**

Aggregate Fonksiyonlar –MIN, MAX, SUM, AVG

Aggregate fonksiyonları yardımıyla bizler veri kümelerimizden sonuçlar çıkarabiliriz.

**AVG:**

AVG fonksiyonu kullandığımız sayısal değerlerden oluşan sütununun ortalama değerini kullanırız

SELECT AVG(length)

FROM film;

**SUM:**

SUM fonksiyonunu kullandığımız sayısal değerlerden oluşan sütunun toplam değerini alırız.

SELECT SUM(length)

FROM film;

**MAX:**

MAX fonksiyonunu kullandığımız sayısal değerlerden oluşan sütunun en yüksek değerini alırız

SELECT MAX(length)

FROM film;

**MIN:**

MIN fonksiyonunu kullandığımız sayısal değerlerden oluşan sütunun en düşük değerini alırız.

SELECT MIN(length)

FROM film;

**GROUP BY:**

Group by komutu kullanımı, bir sütundaki benzersiz değerleri referans alarak diğer sütunlardaki veriler üzerinde filtreleme yapmayı sağlar. GROUP BY komutu kullanırken SELECT ifadesinden sonra gelen sütun adlarının tümüyle beraber kullanmak gerekir. Ancak Select ifadesinde bulunmayan sütunlar da GROUP BY komutuyla beraber kullanılıp gruplama yapılabilir.

Ör:

SELECT rental\_rate, MAX(length) FROM film

GROUP BY rental\_rate;

Bu örnekte benzersiz rental\_rate değerleri referans alınarak length sütunundaki veriler filtrelenmiştir ve her rental\_rate değerine karşılık gelen en büyük length değeri elde edilmiştir, burada rental\_ratele beraber farklı sütunlar da GROUP BY ifadesi yanına eklenebilir SELECT edilmeleri gerekmez, fakat SELECT ifadesi yanına gelen aggregate function olmayan tüm ifadeler(sütunlar) GROUP BY komutu yanına eklenmelidir.

**HAVING:**

HAVING komutu gruplanmış ifadeler için koşullar belirlememizi sağlar, eğer GROUP BY ifadesi kullanmışsak. Bu gruplanmış ifadelerde koşul belirlemek için WHERE kullanamayız, GROUP BY için koşul belirleme komutu HAVING dir.

SELECT rental\_rate, COUNT(\*) FROM film

GROUP BY rental\_rate HAVING COUNT(\*) >325

Aynı işin WHERE ile yapılması

SELECT rental\_rate, COUNT(\*) FROM film

WHERE rental\_rate !=2.99

GROUP BY rental\_rate

**NOT:** WHERE satır bazlı filtre uygular, HAVING grup bazlı filtre uygular.

**AS:**

Sorgular sonucu gelen sonuçlardaki sütunların adlarını değiştirmeye yarayan komuttur. AS kullanmak veya sütun adından sonra AS kullanmadan isimlendirme yapılabilir fakat isimlendirme birden fazla kelimeden oluşuyorsa ve arada boşluk varsa isim tırnak içine alınmalıdır. AS kullanılmasa da sorgu beklendiği şekilde çalışır.

CONCAT:

Sorgu sonucunda gelen verilerin olduğu birden fazla sütunun verisini birleştirip tek bir sütun gibi göstermeye yarar.

SELECT CONCAT(first\_name, ‘ ’,last\_name) AS “isim ve soyisim”

FROM actor

**CREATE TABLE:**

Veri tabanına tablo eklemek için kullanılan komuttur. Grafik arayüz ile tablo oluşturma işlemlerini sorgu ile yapabilmemizi sağlar, CREATE TABLE komutundan sonra tablo adı sonrasında sütun adları veri tipleri ve kısıtlamaları yazılır. Ör:

CREATE TABLE author(

İd SERIAL PRIMARY KEY,

First\_name VARCHAR(50) NOT NULL,

Last\_name VARCHAR(50) NOT NULL,

Email VARCHAR(100),

Birthday DATE

);

**INSERT INTO:**

Tablolara sorgu ile veri ekleme komutudur,

INSERT INTO <tablo\_adı>(sütun1,sütun2…)

VALUES (sütun1\_degeri,sütun2\_degeri…)

Şeklinde her sütun için uygun veri tipinde değerler girilmelidir.ör:

INSERT INTO author(first\_name,last\_name,email,birthday)

VALUES ('Sabahattin','Ali','sabahattin@ali.com','1948-11-07')

**Tablo Şablonu kopyalamak:**

Var olan bir tablonun sütun özelliklerini kullanarak bir kopyasını oluşturmak istediğimizde kullanabileceğimiz komuttur, sütun özelliklerini birebir kopyaladığımız boş bir tablo oluşturmayı sağlar. Ör:

CREATE TABLE author2 (LIKE author) LIKE’dan sonra belirttiğimiz tablonun özelliklerini kopyalar.

**Tablolar arası veri kopyalama**

Var olan bir tabloyla aynı sütun özelliklerine sahip başka bir tabloya veri kopyalamak için bu şekilde kullanmamız gerekir:

INSERT INTO author2

SELECT \* FROM author

BU şekilde tüm verileri koyalarız, eğer belirli bir satırı eklemek istersek ör:

INSERT INTO author2

SELECT \* FROM author

WHERE first\_name='Sabahattin';

**Tablonun kopyasını yaratmak:**

Bir tablonun içindeki verilerle beraber kopyasını oluşturmak için örnek kullanım:

CREATE TABLE author3 AS

SELECT \* FROM author;

DROP TABLE:

Bir tabloyu silmek için kullanılan komuttur. Kullanım ör:

DROP TABLE author4

bu şekilde kullandığımızda eğer silmek istediğimiz tablo yoksa not exist hatası verir. Eğer bu hatayı almak istemiyorsak:

DROP TABLE IF EXISTS author4

Şeklinde kullanabiliriz.