SQL İLE SORGULAMA

SQL veritabanları ile iletişime geçmek için kullandığımız bir dilidir.

SQL bir programlama dili olmsa bile pek çok kişi tarafından bu şekilde kabul görür.

SQ'i bir sorgu dili olarak ifade etmek daha doğrudur.

Kolay Syntax yapısı nedeniyle programlamaya başlamak için en uygun dillerden biridir.

Bilgisayar ortamında saklanan, işlenmemiş her bilgiye veri denir.

Verinin çok önemli olduğu ve üretilen verinin artış hızının inanılmaz boyutlara ulaştığı günümüz dünyasında, verilerin depolanması neredeyse tüm kullanıcılar için bir zorunluluk haline gelmiştir.

Veri saklamak ve bu verileri istenildiği zaman ulaşma ihtiyacı olan her platform SQL kullanmak zorundadır.

Günümüz dünyasında verilerle ilgili bu hızda cevap verebilmek için; veri yığınlarını bir arada tutmak, düzenlemek, listelemek, silmek, güncellemek, istediğimiz analizleri yapabilmek için verileri sorgulama ihtiyacı doğmuştur.

Tüm bu ihtiyaçlar üzerine SQL veritabanı sistem yazılımı geliştirilmiştir.

Veri tabanına ihtiyaç duyulan her iş ve her sektörde kullanılabilir. Finans sektöründe, özellikle bankacılık, havacılık, perakende, internet ve ağ üzerine yapılan tüm uygulama geliştirmelerinde, tüm sosyal medya platformlarında.

SQL ile veri tabanında yer alan verileri kaydedebilir, güncelleyebilir, silebilir, yeni veri girebilir, yeni tablolar oluşturabilir, milyonlarca veri üzerinden analize konu olabilecek tüm veri sorgulamalarını yapabilir ve yeni bir veri tabanı oluşturabiliriz.

Data; bilgi, veri, girdi gibi anlamlar taşımaktadır. Her alanda istatistiksel çalışmaların kaynağında kullanılır.

Datalarımızı sakladığımız veri depoları Database olarak adlandırılır. Database'ler ile iletişime geçmek için Database Management System yani veritabanı yönetim sistemleri kullanılır. Database ile konuşmak için program/arayüze ihtiyaç vardır.

Veritabanı Yönetim Sistemleri temelde ilişkisel veritabanı ve ilişkisel olmayan veritabanları olmak üzere ikiye ayrılır. SQL kullanılan ilişkisel veritabanlarıdır. Burada dataları tablolar halinde depolarız. Buradaki tablolar birbirleriyle ilişkilidir.

SELECT * FROM tablo_adi; komutunu girerek istediğimiz tablonunun tamamını seçebiliriz. Buradaki select seç, * ise tüm satır ve sütunları anlamına gelir. From ile de hangi tablo olduğunu belirtiriz. Burada * yerine tablonun istediğimiz kolonlarının ismini yazarak onun getirilmesini de sağlayabiliriz.

WHERE kolon_adi = 'satırda görmek istediğimiz veri ismi'; komutuyla bir tablonun filtrelenerek ismini girdiğimiz tüm satırları listelemesini istiyoruz.

SQL'de yorum satırı yapmak için satırın başına "- -" ekleyebiliriz.

Örneğin bir isim tablosunda ismin ilk harfi m ikinci harfi a olan tüm isimleri getirmek istediğimizde girmemiz gereken komut satırları şu şekilde olmalıdır;

SELECT * FROM isimler
WHERE first_name LIKE 'MA%'; buradaki yüzde işareti m ve a harfinden sonra ne gelirse gelsin anlamını taşır. Kaç harf ve hangi harf olduğu önemsizdir.

SQL büyük küçük harf ayrımı yapmaz. Komutlarımızı büyük veya küçük harf kullanarak yazabiliriz.

USE veritabanı_adı; komutunu kullanarak istediğimiz veritabanını seçebiliriz.
Bu komutu girdikten sonra farklı bir database içindeki tabloya erişmek istersek SELECT * FROM veritabanı_adı.tablo_adı; komutunu girmeliyiz.

Bir tabloda istediğimiz sütunları getirmek için SELECT sütun_adi FROM tablo_adı komutunu kullanırız. Birden fazla sütun ismini yan yana yazarak daha fazla sütunu listeleyebiliriz.

SELECT *, sütun_adi FROM tablo_adi komutuyla öncelikle istediğimiz tablonun tümünü ardından seçtiğimiz tablonun istediğimiz sütununu getirmesini isteriz.

SELECT DISTINCT sütun_adi FROM tablo_adi; komutunu kullanarak çoklayan (tekrarlayan) ifadeleri siler ve tekil olan verileri sıralar. Bu verilerin kaç tane olduğunu saydırmak ve kaç tane olduğunun sayısını yazdırmak için de SELECT COUNT(DISTINCT sütun_adi) FROM tablo_adi; komutunu kullanırız. Başlangıçta sildiğimiz çoklanan ifadelerin kaç tane olduğunu da görmek istersek SELECT COUNT (DISTINCT sütun_adi), COUNT (sütun_adi) FROM tablo_adi; komutunu yazarız.

Eğer tek satırda iki kolonun teklenmiş halini görmek istersek SELECT DISTINCT sütun_adi, sütun_adi2 FROM tablo_adi; komutunu kullanırız. Bu komut iki sütunun kombinasyonunu gösterir.

SELECT 2+6; gibi işlemleri yaptırarak sonucu yazdırabiliriz.

SELECT sütun_adi, sütun_adi + 1 FROM tablo_adi; komutunu kullanarak istediğimiz sütunun tüm verilerine 1 ekleyerek yeni bir klonda listelemiş olduk

İşlem yaptığımız bu sütuna isim vermek için de
SELECT sütun_adi, (sütun_adi + 1) AS "yeni kolon" FROM tablo_adi; komutunu yazmalıyız. Sütun adını yazarken arada boşluk bırakacaksak tırnak içine almalıyız fakat yeni_kolon gibi bir isim yazarsak tırnak kullanmaya gerek kalmaz.

Seçtiğimiz bir tabloda; *filitreleme yaparak istediğimiz verileri göstermesini söylemek için* WHERE tablo_adi = 'istenilen veri değeri'; komutunu kullanırız.

Seçilen tabloda istediğimiz veri sütunun değerini belli bi değer olarak belirleyip listelemesini isteyebiliriz bunun için WHERE sütun_adı = veri1; komutunu kullanırız.

ORDER BY length ASC; komutuyla verileri küçükten büyüğe sıralayarak listelemesini isteriz. Bu varsayılan bi komut olduğu için ASC yazmamış olukta küçükten büyüğe sıralama yapar. Büyükten küçüğe sıralama için ORDER BY length DESC komutunu kullanırız.

WHERE sütun_adi between 80 AND 90; komutuyla 80 ve 90 dahil bu aralıktaki verileri sıralamasını isteriz.

İstediğimiz sütunda ilk harfi A olan tüm verileri görmek istersek WHERE sütun_adi LIKE 'A%'; komutunu yazarız.

Bazı operatörler ve anlamlarına bakarsak;

= eşittir
> büyüktür
>= büyük eşittir
< küçüktür
<= küçük eşittir
<> eşit değildir

İki koşulun da sağlandığı mesela seçtiğimiz tabloda iki farklı sütundan istediğimiz değerleri getirmesini istersek WHERE sütun_adi = veri1 AND sütun_adii = veri2; komutunu kullanırız.

Seçilen tabloda aynı sütuna ait iki farklı veri değerini listelemek istersek WHERE sütun_adi = veri1 OR sütun_adi = veri2; komutunu kullanırız.

Farklı sütun için yaparsak alacağımız sonuç şu şekilde olur. Örneğin WHERE sütun_adi = veri1 OR sütun_adii = veri2; komutunu kullanırsak, birinci sütunu veri1 değerine sahip olan ve ikinci sütun değeri ne olursa olsun fark etmeksizin her değeri alabilen verilerle birlikte ikinci sütun değeri veri2 olan ve ilk sütun değeri herhangi bir veri alan sonuçları listeler.

OR kullanarak sorgumuzu genişletirken, AND ifadesiyle sorgumuzu daraltıyoruz.

Herhangi bir şart belirteceksek kolon adını mutlaka yazmamız gerekir örneğin bu kullanım hata verecektir. WHERE sütun_adi = veri1 OR veri2; veya
WHERE sütun_adi = veri1 , veri2;

Bu gösterimi kullanabilmek için IN operatörünü kullanırız.
WHERE sütun_adı IN (‘veri1’ , ‘veri2’); şekline kullanılır. Bu komutla
Aynı sütuna ait farklı veri değerlerini listelemiş oluruz.

NOT IN operatörüyle tabloda görmek istemediğimiz verileri yazarak haricinde kalan tüm verilere erişebiliriz. WHERE sütun_adi NOT IN (‘veri1’ , ‘veri2’); şeklinde kullanılır.

AND ve OR operatörlerinin birlikte kullanımı için dikkat edilmesi gereken hangi sırayla kullanıldığıdır. Örneğin WHERE sütun_adi = veri1 OR sütun_adı = veri2 AND sütun_adı2 = veri3; komutunu kullanırsak son iki sorguyu beraber okuyarak veri2 ve veri3 değerine aynı anda sahip olanları ve bütün veri1 değerlerini listeler.

Ama WHERE (sütun_adi = veri1 OR sütun_adı = veri2)AND sütun_adı2 = veri3; komutunu kullansaydık hem veri1 ve veri3 değerine aynı anda sahip olanları hem de veri2 ve veri3 değerine aynı anda sahip olanları listelerdi. Daha dar bi veri kümesini hedef alır.

LIKE operatörünü kullanarak seçtiğimiz tabloda istediğimiz sütundan çekeceğimiz verinin belirli harflerini ihtiyacımıza göre ayarlayabiliriz. Örneğin WHERE sütun_adi LIKE 'NA%'; komutuyla ilk n ve a olan sonrasında gelen harf ve sayısı fark etmeksizin tüm verileri getirir.

İlk harfi a ikinci harfi n ve sonraki üç harfi ne olduğu önemsiz olan beş harfi verilere erişmek istersek WHERE sütun_adi LIKE 'AN___'; komutunu kullanırız.

Kullandığımız % ve _ ifadeleri wildcards karakterlerdir.

WHERE sütun_adi >=değer1 AND sütun_adi <=değer2; komutuyla değer1 ve değer2 dahil olmak üzere seçilen sütundaki bu aralıkta olan değerleri listeleriz.

Aynı işlemi WHERE sütun_adi BETWEEN değer1 AND değer2 komutuyla da yapabiliriz.

NOT BETWEEN kullanarak iki değer arasındaki veriler dışında kalan verilere erişiriz.

Özet bir bilgi istendiği zaman mesela bir sütundaki değerlerin toplamını görmek istersek Aggregate Fonksiyonlarını kullanırız.

SUM Fonksiyonunu kullanarak seçilen sütunda bulunan verilerin toplamını yazdırmış oluruz bunun için SELECT SUM(sütun_adi) FROM tablo_adi; şeklinde yazmalıyız.

Belli bir aralıkta bulunan verilerin toplamını bulmak istersek. Örneğin bir müşteri tablomuz olduğunu ve 5. Ayda yapılan alışveriş tutarlarının toplamını görmek istediğimizi düşünelim bunun için SELECT SUM(miktar) FROM ödemeler
WHERE MONTH(ödeme_tarihi)=5; komutunu kullanırız.

Örneğin müşteri numarası 2 olan müşterinin 4. Ve 6. Aylar arasında yaptığı alışveriş tutarını görmek istersek bunun için
SELECT SUM(miktar) FROM ödemeler
WHERE MONTH(ödeme_tarihi) BETWEEN 4 AND 6 AND müşteri_id = 2;
komutunu yazmalıyız.

Bir sütundaki belli bir aralıkta bulunan verilerden en küçük olananına erişmek için SELECT MIN(sütun_adı) FROM tablo_adi; fonksiyonunu kullanırız.

Bir sütundaki belli bir aralıkta bulunan verilerden en büyük olananına erişmek için SELECT MAX(sütun_adı) FROM tablo_adi; fonksiyonunu kullanırız.

Bit tabloda 2025-05-25 11:30:37 şeklinde tarih ve saat olduğunu düşünelim bu sütundan sadece yıl veya ay ya da istediğimiz veriyi ayrı bir sütunda yazdırmak istersek SELECT sütun_adi, YEAR(sütun_adi) FROM tablo_adı; şeklinde kullanırız.

AVG fonksiyonu seçtiğimiz sütundaki verilerin ortalamasını alarak bize döndürür. SELECET AVG(sütun_adi) FROM tablo_adi; şeklinde kullanılır.

COUNT fonksiyonu temel olarak satırları sayar. Üç temel alt işlevi vardır.

SELECT COUNT(*) FROM tablo_adi; komutunu kullanarak toplamda tüm tablodaki satır sayısını verir. * yerine sütun adı girerek seçtiğimiz sütunda kaç satır olduğuna da bakabiliriz.

Sütun adı girerek yaptığımız sorgularda NULL olan değerleri satır olarak saymaz.

SELECT COUNT(DISTINCT sütun_adı); FROM tablo_adi; yazarsak çoklamayan yani tekrar etmeyen değerlere sahip satır sayısını verir.

Çoklayan verileri tekil olarak gruplayıp listelemek için kullanılır. Dıstınct komutunun işlevini yapar gerçekleştirir. Farkı dıstınct'in kullanıldığı yerler daha kısıtlıdır.

```
SELECT sütun_adı FROM tablo_adi
group by sütun_adi;
```

Şeklinde kullanılır.

Bu çoklayan verilerden kaçar tane olduğunu da group by komutunu kullanarak bulabiliriz. Bunun için

```
SELECT sütun_adı, COUNT(sütun_adi) FROM tablo_adi
group by sütun_adi; şeklinde yazarız.
```

Ödemeler isimli bir tablomuz olduğunu düşünelim. Bir müşterinin defalarca kez alışveriş yaptığını ve bu tutarların toplamını listelememiz gerektiğiniz farz edelim. Bunun için sum komutuyla birlikte group by komutunu beraber kullanmamız gerekir.

SELECT sütun_adi, sum(sütun_adi2) FROM tablo_adi
WHERE sütun_adi=değer1
GROUP BY sütun_adi; şeklinde yazarız.

Eğer bütün müşteriler için listelemek istersen WHERE satırını yazmamalıyız.

Group by kullanmadan şu şekilde de yapabiliriz;

SELECET COUNT(sütun_adi) FROM tablo_adi
WHERE sütun_adi=değer1;

SUM komutundan önce yani aggregate fonksiyonlarından önce yazdığımız sütun isimlerini group by komutundan sonra yazmak zorundayız.

Ödemeler tablosundan bir müşteriye ait en düşük miktardaki alışverişini görmek istersek bunun için müşterilerin bi sütunda ve alışveriş tutarlarının başka bir sütunda yer aldığını düşünürsek şu komutları kullanmalıyız;

SELECT sütun_adi, MIN(sütun_adi2) FROM tablo_adi
WHERE sütun_adi=değer1
GROUP BY sütun_adi

MAX komutunu GROUP BY ile kullanım örneğine bakarsak,

SELECT sütun_adi, sütun_adi2 MAX(sütun_adi3) FROM tablo_adi
WHERE sütun_adi=değer1
GROUP BY sütun_adi, sütun_adi2

Örneğin bir kolonda bulunan müşterilerden belli bir müşterinin yaptığı alışveriş tutarlarının ortalamasını listelemek istersek ARG fonksiyonunu kullanırız.

SELECT sütun_adi AVG(sütun_adi2) FROM tablo_adi
WHERE sütun_adi=değer1
GROUP BY sütun_adi

Aynı işlemi Group by kullanmadan şu şekilde yapabilirdik,

SELECT SUM(sütun_adi)/COUNT(*) FROM tablo_adi
WHERE sütun_adi2=değer1

Buradaki count(*) satır sayısı anlamına geliyor.

GROUP BY ile COUNT kullanımı aşağıdaki gibidir.

SELECT sütun_adi, COUNT(sütun_adi2) FROM tablo_adi
GROUP BY sütun_adi;

Order by kullanım amacı tabloda yapmak istediğimiz tüm işlemleri yaptıktan sonra tabloyu ne şekilde görmek istiyorsak nasıl sıralamak istiyorsak kullandığımız komuttur. Gösterim amaçlı kullanırız.

ORDER BY sütun_adi ASC komutu verileri küçükten büyüğe ve a'dan z'ye sıralamamızı sağlar. ASC yazmasak bile varsayılan olarak sıralama bu şekilde karşımıza çıkar.

ORDER BY sütun_adi DESC komutu ise verileri büyükten küçüğe ve a'dan z'ye sıralamamızı sağlar.

ORDER BY sütun_adi DESC, sütun_adi2 ASC; şeklinde bir komut yazarsak
İlk şartı yerine getirir, ikinci şartı ise ilk şartın her bir kategorisi için gerçekleştirir. Yani örneğin ilk sütunda 10 adet 5 değeri olduğunu düşünürsek, her 5 değeri için ikinci sütundaki değerleri kendi içinde küçükten büyüğe sıralar. İlk sütunun her değeri için bunu baştan küçükten büyüğe sıralar.

Order by kullanımının diğer bir şekli de aşağıdaki gibidir.

SELECT sütun_adi, sütun_adi2, sütun_adi3 FROM tablo_adi
ORDER BY 1 DESC;

Yazarsak ilk satırda sırayla yazdığımız sütunlardan 1. Sırada olanı getirir.

SELECT sütun_adi, sütun_adi2, sütun_adi3 FROM tablo_adi
ORDER BY 1 DESC, 2 ASC;

Şeklinde de kullanabiliriz.

Bir gruplama yapıyorsak ve bu gruplamaya bir fonksiyon uyguluyorsak bundan sonra WHERE komutunu kullanamayacağımız için HAVING kullanırız. Where komutu gibi filitreleme yapmak için kullanırız.

Örneğin;

SELECT sütun_adi, COUNT(sütun_adi2) FROM tablo_adi
GROUP BY sütun_adi
HAVING COUNT(sütun_adi2) BETWEEN değer1 AND değer2;

HAVING ifadesini group by ile birlikte kullanılır.

Group by ile kullanılmadığı zaman where yerine kullanılabilir ama sık kullanılan bir işlem değildir.

HAVING komutunun asıl kullanımı ise

Mesela tüm müşterilerin yaptığı alışverişlerin toplamını görmek istersek öncelikle

SELECT sütun_adi, SUM(sütun_adi2) FROM tablo_adi
GROUP BY sütun_adi

Komutlarını kullanırız.

Bu alışveriş tutarları içinden örneğin 100'den büyük olanları listelemek istersek HAVING burada devreye girer.

SELECT sütun_adi, SUM(sütun_adi2) FROM tablo_adi
GROUP BY sütun_adi
HAVING SUM(sütun_adi2) > 100;

Şeklinde yazarız.

```
SELECT * FROM tablo_adi
ORDER BY sütun_adi DESC
LIMIT değer;
```

Komutuyla istediğimiz veriyi sınırlayarak listeleyebiliriz.

Farklı bir kullanım şeklide aşağıdaki gibidir;

```
SELECT sütun_adi FROM tablo_adi
LIMIT değer, örn:4 , değer2, örn:10;
```

Şeklinde yazarsak listeleyeceğimiz verilerden ilk 4 satırı atlayarak sonraki 10 satırı getirmesini isteriz.

Şuana kadar yazdığımız komutları belli bir kullanım sırası vardır;

```
SELECT FROM
JOIN
IN
WHERE
GROUP BY
HAVING
ORDER BY
LIMIT
```

JOIN kavramının kendi nitelikleri vardır bunlar;

```
INNER JOIN -> kesişim
LEFT JOIN -> sol tablo (en çok kullanılacak olan)
RIGHT JOIN -> sağ tablo
FULL JOIN -> hepsi
```

Tablo oluşturmak için girmemiz gereken komutlar sırasıyla

```
CREATE DATABASE Deneme;     /deneme isimli bir veritabanı oluşturuyoruz
USE Deneme                  / denem veritabanını kullan diyoruz


CREATE TABLE Müşteri        / müşteri isimli bir tablo yarat diyoruz

(
müşteriNo int,
ŞehirId int,                /tabloda olmasını istediğimiz bilgi başlıklarını yazıyoruz
MüşteriAdı varchar(25)      /kolon isimleri/ Max 25 karakterli olacak
);

CREATE TABLE Şehir

(
ŞehirId int,
ŞehirAdı varchar(25)
);

INSERT INTO Müşteri (MüşteriNo,ŞehirId,MüşteriAdı)
VALUES
(1,1,"Esma"),
(2,1,"Ali"),                /ilgili bilgileri virgülle ayırarak dolduruyoruz
(3,2,"Ayşe"),
(4,NULL,"Mehmet");
```

Oluşturduğumuz bu tablolarda ortak olan ŞehirId kolonları ortak. Bu ortak kolonları kullanarak bu tabloları birleştirebiliriz. Bunun içinde JOIN kullanırız.

Bir tabloda benzersiz değer alan (her değer için farklı olan) kolon primary key olarak adlandırılır. Şehir tablosunun primary key ifadesi şehirId'dir. Boş değer alamaz. Müşteri tablosunda da müşteriNo primary key'dir.

Müşteri tablosunda bulunan ŞehirId kolonu başka bir tablonun primary key'i ile bağlantılı olan kolondur ve foreign key olarak adlandırılır.

Oluşturduğumuz bu müşteri ve şehir tablolarından müşteriNo, müşteri Adı yazsın, şehirID ve şehirAdı yazsın gibi bir işlem yapmak istediğimizde

```
SELECT * FROM müşteri
INNER JOIN şehir
ON müşteri.ŞehirId=şehir.ŞehirId
```

Komutunu kullanırız. Buradaki INNER JOIN komutu iki kolonda da ortak olan şehirId'leri getirir.

LEFT JOIN kavramı soldaki tabloyu (ilk yazılan) olduğu gibi getirir. Diğer tabloyla kesişenleri de gösterir.

RIGHT JOIN ise sağdaki tablonun (ikinci yazılan, komuttan sonraki) tamamını olduğu gibi getirir. Soldaki tablo ile kesişenleri getirir.

JOIN kullanırken tabloları yan yana birleştirmiştik. Tabloları alt alta birleştirmek istediğimizde UNION ve UNION ALL kullanırız.

Union all komutu iki tabloda bulunan tüm verileri listelerken union komutu çoklayan verileri göstermez. Tabloların veri tipi aynı olmalı.