**BETWEEN, NOT BETWEEN**

->city ID si 3 ile 5 arasında olanları getir

\*Select \* from city where city\_id between 3 AND 5;

-> city ID si 3 ile 5 arasında olmayanları getir

\*Select \* from city where city\_id not between 3 AND 5;

->city ID’si 3 ile 5 arasında olanlar ve country ID’si 1 ve 3 arasında olanları getir.

\*Select \* from city,country where (city\_id between 3 AND 5) AND (country\_id between 1 AND 3); ->

**LIKE , NOT LIKE, ILIKE, NOT ILIKE**

->username’i A(büyük a) ile başlayıp y(küçük y) ile biten userları getir.

\*Select \* from users where username like ‘A%y’;

->username’i A(büyük a) ile başlayıp y(küçük y) ile biten ve aradaki TEK harf olup bu harfin önemsiz olduğu userları getirir.

\*Select \* from users where username like ‘A\_y’;

->ILIKE harfin büyük küçük duyarsız oldugu durumlar icin kullanılır.

-> usernamesi a ile başlayıp(büyük küçük farketmez) devamında tek bir harf olanları getir.

\*Select \* from users where username ilike ‘A\_’;

**DISTINCT , COUNT**

Distinct : Aynı olan verilerden sadece birini göster

Count: Adeti göster

->users tablosundaki şehirleri sadece 1 kere getir.Örneğin 2 tane Türkiyeli olan users var ama sadece 1 tane Türkiye getirir.

\*Select distinct county\_name from users

->usernamesi sadece 2 harfli olan ve sonu küçük büyük farketmez t harfi ile biten userların sayısını getir.

\*Select count(\*) from users where user\_name ilike ‘\_t’;

->country namesi sadece 2 harfli olan ve sonu küçük büyük farketmez t harfi ile biten country namelerin sayısını getir.(2 country var ve aynıysa sadece birini getirir distinct ten ötürü.)

\*Select count(distinct country\_name) from users where country\_name ilike ‘\_t’;

**ORDER BY**

Order By : Sıralama

select vld\_def\_id ,is\_actv from vld\_def vd order by vld\_def\_id ASC

select vld\_def\_id ,is\_actv from vld\_def vd order by vld\_def\_id DESC

**LİMİT , OFFSET**

Limit : Kaç adet veri getireceğimiz belirler

Offset : Belirlediğimiz kadar veriyi atlar

->Usernamesi Samet olanların user IDlerini ASC sırala, ilk 4 veriden sonrasını (5 dan başlayacak şekilde) sonraki 10 veriyi getir.

\*Select \* from users where user\_name = ‘Samet’ order by user\_id ASC OFFSET 4 LIMIT 10;

**AGGREGATES(MIN,MAX,AVG,SUM)**

->Prod ID’si 1000’den sonra olanların prod\_price ortalamasını getir.

\*Select avg(prod\_price) from prod where prod\_id >= 1000;

-> prod\_price ların toplamını getir

\*Select sum(prod\_price) from prod where prod\_id >= 1000;

->Minimum prod\_price getir.

\*Select min(prod\_price) from prod where prod\_id >= 1000;

->Maximum prod\_price getir.

\*Select max(prod\_price) from prod where prod\_id >= 1000;

**LENGTH**

->firstName’sinin lengthi 5 ten büyük olanları getir

select \* from employees where length(first\_name) > s5

**GROUP BY**

Bu group by’ı sum,avg,min,max,count gibi aggregates queryler ile gruplayarak kullanırız.

->Customerları ülkelerine göre gruplayıp sayılarını getir.

\*select country,count(\*) from customers group by customers.country;

->Çalışanları title\_of\_courtesy lerine göre grupla

\*select title\_of\_courtesy,count(\*) from employees group by title\_of\_courtesy;

**HAVING**

Group By ile grupladığımız verilere koşul ekler.

-> Tedarikçiye göre fiyatların ortalamasını grupla ve fiyatı 100den fazla olan verileri getir.

select supplier\_id,avg(unit\_price) as "Fiyat" from products group by supplier\_id having avg(unit\_price) > 100;

->Employee id ye göre taşıma ücreti toplamı 10000 den fazla olan verileri getir.

select employee\_id,sum(freight) from orders group by employee\_id having sum(freight)>10000;

**CHECK**

Tablo oluştururken tabloya daha sonra atılacak insert into kayıtlarına belli koşullar ekler.

CREATE TABLE Persons(IDint NOT NULL,  
    LastName varchar(255) ,  
    FirstName varchar(255),

LastName varchar(255) NOT NULL UNIQUE,  
    Age int CHECK (Age>=18)  
);

**JOIN OPERASYONLARI**

**INNER JOIN**

İki tane seçtiğimiz tablodaki alanların kesişimlerini bize getirir.

->Orders tablosu ile order\_details tablosunu order\_id’ye göre eşitle buradan belirtilen alanları getir.

SELECT order\_details.order\_id ,product\_id , customer\_id , unit\_price

FROM orders

INNER JOIN order\_details ON **orders.order\_id = order\_details.order\_id;**

**LEFT JOIN**

2 tablo seçtiğimizi varsayalım. 1 tanesinde(LEFT) herşeyi ve diğer tablodaki sadece kesişen verileri getirir.

->Aşağıdaki sorguda orders tablosundan herşeyi getirecek fakat sağdaki tablodan yani order\_details tablosundan sadece order\_idleri birbirine eşit olanı getirecektir.

select order\_details.order\_id ,product\_id , customer\_id , unit\_price

from orders

left join order\_details on **orders.order\_id = order\_details.order\_id;**

**FULL JOIN**

Her iki tablodaki eşleşen tüm verileri getirir.

select \*

from students

full join teachers on students.teacher\_id = teachers.teacher\_id;

**UNION**

Bu union 2 sorgunun sonucunu tek seferde getirmeye yarar.

Bu sorguda student no ya göre sıralayıp 2 veri ve student name göre sıralayıp 3 veri getirmesini istedik. Sonuç olarak bize 5 veri dönmeli fakat eğer bu iki sorguda kesişen veriler varsa daha az veri getirebilir. Şayet kesişenleride 2 kere getirmek isteseydik union all kullanırdık. O zaman 5 veri gelirdi fakat kesişenler 2 kere gelirdi.

( select \* from students order by student\_no asc limit 2 )

union

( select \* from students order by student\_name asc limit 3 );

( select \* from students order by student\_no asc limit 2 )

union all

( select \* from students order by student\_name asc limit 3 );