**SQL KOMUTLARI**

|  |  |
| --- | --- |
| **SELECT**  **FROM** | **SELECT** first\_name **FROM** employees; |
| **DISTINCT** | SELECT DISTINCT column\_name FROM table\_name  SELECT **DISTINCT** first\_name, gender FROM employees;  ***{Distinct=farklı. Unique=benzersiz. Belirtilen column(lar)daki distinct/unique* olan datayı çek getir. Birden fazla column belirtmişsek onların kesişimlerini alıyor}** |
| **WHERE** | SELECT column\_name(s) FROM table\_name WHERE condition(s);  SELECT \* FROM student\_table **WHERE** grade > 70 ***{if conditions gibi düşün}*** |
| **LIMIT** | SELECT column\_name(s) FROM table\_name LIMIT number\_rows;  SELECT \* FROM student\_table WHERE grade > 70 **LIMIT** 2;  ***{LIMIT’ten sonraki sayı kadar satır bilgisini seç getir}*** |
| **ORDER BY**  **ASC, DESC** | SELECT column\_name(s) FROM table\_name ORDER BY column\_name(s) ASC|DESC;  SELECT \* FROM employees **ORDER BY** first\_name **ASC**;  SELECT first\_name, last\_name, salary FROM employees **ORDER BY** salary **DESC**;  SELECT column\_name(s) FROM table\_name ORDER BY column1 ASC|DESC, column2 ASC|DESC, columnN ASC|DESC;  SELECT \* FROM employees **ORDER BY** ***gender* DESC, *first\_name*** **ASC**;  ***{Önce gender’i sıralıyor. Sonra gender’ler içinde first\_name’e göre sıralama yapıyor.}*** |
| **WHERE +**  **ORDER BY** | SELECT column\_name(s) FROM table\_name WHERE condition ORDER BY column\_name(s)s ASC|DESC;  SELECT \*  FROM employees  **WHERE** salary > 80000 **{Önce WHERE geliyor sonra ORDER BY!!}**  **ORDER BY** first\_name DESC; |
| **WHERE**  **AND** | WHERE left\_conditon AND right\_condition  SELECT \*  FROM employees  **WHERE** job\_title = 'Data Scientist' **AND** gender = 'Male'; |
| **WHERE**  **OR** | WHERE first\_condition OR second\_condition  SELECT \*  FROM employees  **WHERE** job\_title **=** 'Data Scientist' **OR** gender **=** 'Male'; |
| **WHERE NOT** | WHERE NOT first\_condition  **WHERE NOT** gender = 'Female'; |
| **WHERE**  **BETWEEN**  **AND** | WHERE test\_expression BETWEEN low\_expression AND high\_expression  Also we could write this query like this:  WHERE test\_expression >= low\_expression AND test\_expression <= high\_expression  SELECT \*  FROM employees  **WHERE** salary **BETWEEN** 80000 **AND** 90000;  Also we could write this query like this:  SELECT \*  FROM employees  **WHERE** salary **>=** 80000 **AND** salary **<=** 90000;  {Between’de sol ve sağdaki **iki değer de DAHİL** olarak kabul edilecek. **İstisnası**, eğer tarih verisi **date-time** şeklinde ise **sondaki tarih dahil olmuyor**. Buna bir gün eklememiz gerekiyor. Çünkü 23:59’dan sonra bir sonraki güne geçiyor ve date-time’da time olarak 00:00 yazıyor} |
| **WHERE**  **NOT BETWEEN**  **AND** | WHERE test\_expression NOT BETWEEN low\_expression AND high\_expression  SELECT \*  FROM employees  **WHERE** salary **NOT BETWEEN** 80000 **AND** 90000;  *Also we could write this query like this:*  SELECT \*  FROM employees  **WHERE** salary **<** 80000 **OR** salary **>** 90000; |
| **BETWEEN AND ORDER BY** | SELECT \* FROM employees  **WHERE** hire\_date **BETWEEN** '2018-06-01' **AND** '2019-03-31'  **ORDER BY** hire\_date; |
| **WHERE**  **NOT IN** | WHERE column\_name IN (value\_list)  SELECT \*  FROM employees  **WHERE** job\_title **IN** *('Data Scientist', 'Business Analyst', 'Project Manager', 'Web Developer')*;  SELECT \*  FROM employees  **WHERE** job\_title  **NOT IN** ('Operations Director', 'HR Manager', 'Sales Manager'); |
| **WHERE**  **LİKE**  **%**  **\_** | SELECT column\_name(s)  FROM table\_name  WHERE column\_1 LIKE pattern;  SELECT \*  FROM student\_info  **WHERE** county **LIKE** 'Wo**%**'; -- Wo ile başlayan ülkeler  SELECT \*  FROM student\_info  **WHERE** field **LIKE** '**%**Developer'; --sonu Developer ile bitenler  SELECT first\_name  FROM employees  **WHERE** first\_name **LIKE** 'El**\_**is';  ***{Linda’daki i ve n harleri için 2 alt tire kullandık}*** |
| **PRAGMA** | -- if you want to make **LIKE** operator **case-sensitive**, you need to use **PRAGMA** statement  PRAGMA case\_sensitive\_like = true;  **PRAGMA** *case\_sensitive\_like = true*;  SELECT \*  FROM student\_info  WHERE field LIKE '%developer';, |

|  |  |
| --- | --- |
| **COUNT( )** | SELECT COUNT(column\_name)  FROM table\_name;  SELECT **COUNT(first\_name)**  FROM student\_info;  ***{COUNT (\*) şeklinde kullanırsan tekrarları ve içeriği NULL olan satırları da getirir. Sütun ismiyle kullanırsan tekrarları getirir ama NULL ları getirmez. NULL ların sayılmasını istemiyorsak \* kullanmayıp parantez içine mutlaka sütun ismi girmeliyiz.}*** |
| **AS** | SELECT COUNT(first\_name) **AS** count\_of\_students  FROM student\_info;  ***{AS komutu ile listelenen bilgiye takma ad veriyoruz böylece ekrana getirilen sonucun ne olduğunu daha iyi anlayabiliyoruz.}*** |
| **COUNT( DISTINCT)** | SELECT **COUNT(DISTINCT field)** AS count\_of\_field  FROM student\_info;  **{Birbirinden farklı (unique) kaç bilgi varsa onun sayısını verir.}** |
| **COUNT**  **WHERE** | SELECT COUNT(\*) AS count\_of\_students  FROM student\_info  WHERE state = 'Virginia'; |
| **MIN** | SELECT MIN(column\_name)  FROM table\_name; ***{ignores the NULL values}***  SELECT **MIN(salary)** AS lowest\_salary  FROM employees;  SELECT **MIN(salary) AS** lowest\_salary  FROM employees  **WHERE** gender = 'Female';  SELECT **MIN(hire\_date)** AS earliest\_date  FROM employees; |
| **MAX** | SELECT MAX(column\_name)  FROM table\_name; ***{ignores the NULL values}***  SELECT **MAX(salary)** AS highest\_salary  FROM employees;  SELECT **MAX(salary)** AS highest\_salary  FROM employees  **WHERE** gender = 'Male';  SELECT **MAX(hire\_date)** AS newest\_date  FROM employees; |

|  |  |
| --- | --- |
| **SUM** | SELECT SUM(column\_name)  FROM table\_name; {***returns the sum of a numeric column}***  ***Sadece numeric datalara uygulanıyor!!***  SELECT **SUM(salary)** AS total\_salary  FROM employees; |
| **AVG** | SELECT AVG(column\_name)  FROM table\_name; ***{calculates the average of numeric colum}***  ***Sadece numeric datalara uygulanıyor!!***  SELECT **AVG(salary)** AS average\_salary  FROM employees;  SELECT name, AVG(Milliseconds)  FROM tracks  WHERE Milliseconds > 393599.212103911;  SELECT name  FROM tracks  WHERE Milliseconds > (SELECT AVG(Milliseconds)  FROM tracks); |
| **GROUP BY** | SELECT column\_1, **aggregate\_function**(column\_2)  FROM table\_name {**Her zaman WHERE’den sonra gelir**  GROUP BY column\_1; **Her zaman ORDER BY’dan önce gelir.}**    **{veri grubu başına yalınızca bir sonuç döndürür. Bunun için verileri unique olarak alır yani kendi içinde DISTINCT yapar. Mesela aşağıdaki örneğe göre Male ve Female’ler için iki ayrı sonuç döndürür.}**    !!! ***SELECT’teki nonaggregate ifade (column\_1), GROUP BY cümlesinde olmalı!!! Yani neyi GROUP BY’ın yanına yazıp grupluyor isek aynen onu SELECT’in yanına yapıştır.***    SELECT *gender*, **COUNT**(gender)  FROM employees  **GROUP BY** *gender*; ***{gender’ların sayılarını gender lara göre (yani gender sütunundaki verileri unique/DISTINCT yaparak) grupla. Yani burada gender Male ve Female’lerden oluştuğu için, Male ve Female başlıkları altında iki grup olarak sayıları ver.}*** |
| **GROUP BY**  **ÖRNEKLERİ** | SELECT gender, **COUNT**(job\_title)  FROM employees  **WHERE** job\_title = 'Data Scientist'  **GROUP BY** gender; **{job\_title’lar arasında Data Scientist’leri çek, onları gender lara göre gruplandır.}**  SELECT job\_title, count(job\_title)  FROM employees  GROUP BY gender, job\_title; {gender’lara göre hangi iş pozisyonlarından kaçar tane var}  SELECT job\_title, gender, count(gender)  FROM employees  GROUP BY job\_title, gender ; { iş pozisyonlarına göre gender’lardan kaçar tane var}  SELECT job\_title, gender, count(gender)  FROM employees  WHERE job\_title='Data Scientist' {Data Scientist olarak gender’lardan kaçar tane GROUP BY job\_title, gender ; var}  SELECT Composer, COUNT(TrackId)  FROM tracks  GROUP BY Composer; {Her bestecinin kaçar şarkısı var?  Burada null olanları da alır.  SELECT Composer, COUNT(TrackId)  FROM tracks  WHERE Composer IS NOT NULL {Her bestecinin kaçar şarkısı var?  GROUP BY Composer; burada null olanları çıkarttık  SELECT Country, COUNT(CustomerId)  FROM customers  GROUP BY Country; {her ülkeden kaçar tane müşteri var?}  SELECT AlbumId, MIN(Milliseconds)  FROM tracks  GROUP BY AlbumId; {her albümdeki süresi en kısa olan şarkıyı listele} |
|  | **GROUP BY, Aggregate function’ı çağırmadan önce sonuçları gruplandırır. Bu, tüm sorgu yerine gruplara aggregate func. uygulamanıza olanak tanır.**  **WHERE cümlesi, aggregate’ten önceki veriler üzerinde operasyon yapar (çalışır).**  **WHERE cümlesi, GROUP BY cümlesinden önce çalışır.**  **Dolayısıyla ana tablomuzun sadece WHERE cümlesindeki şartları sağlayan satırları gruplanır.**  **ORDER BY, GROUP BY’dan sonra gelir.**  ***Yani ilk önce WHERE şartı uygulanıp data filtrelenir, sonra üzerine GROUP BY, sonra aggregate function uygulanır. Son olarak da sonuç ORDER BY a göre sıralanır.***  ***(çalışma sütunlarını + koşula göre seç + fonksiyona göre gruplandır + sıralamasını belirle)*** | | |
| **MIN**  **GROUP BY** | SELECT gender, **MIN**(salary) AS min\_salary  FROM employees  **GROUP BY** gender; | | |
| **MAX**  **GROUP BY** | SELECT gender, **MAX**(salary) AS max\_salary  FROM employees  **GROUP BY** gender; | |
| **GROUP BY**  **ORDER BY** | SELECT gender,  MAX(salary) AS max\_salary  FROM employees  **GROUP BY** gender  **ORDER BY** max\_salary DESC ;  {salary içerisindeki maksimum ücreti genderlarla grupla yani F’lerden en yüksek alan ile M’lerden en yüksek alanları bir araya getir. bunları yüksekten alçağa doğru sırala}  ***{ yukarda MAX(salary)’e max\_salary olarak takma ad vermiştik. Ve bu adı ORDER BY max\_salary ifadesinde kullandık. Bunun yerine ORDER BY MAX(salary) olarak da kullanabiliyoruz.}*** | |
| **SUM / AVG**  **GROUP BY** | SELECT gender, **SUM(salary)** **AS** total\_salary  FROM employees  **GROUP BY** gender;  SELECT gender, **AVG(salary)** **AS** average\_salary  FROM employees  GROUP BY gender; | |