



# Structured Query Language

## What is database?

Gerektiginde kullanilmak sureti

Düzenlenmis veri koleksiyonlari. Verilerimizi saklar.

## How are databases used in the real-world?

Banka e-ticaret platformlari

"Unler web siteleri?

Müşteri, hasta, öğrenci bilgileri bu sayede tutulabilir.

Üyelik varsa database vardir.

## DBMS $\rightarrow$ Database Management System

(\*) Dersimizde relational database'lerden bahsedilecek

Yapilandirilmiş veri tablolaridir. (ex: gibidir.)

Non-relational ship'de böyle birsey yok.

(\*) Edgar Frank  $\rightarrow$  SQL'i icat etti.

(\*) What is database?

Yapilandirılmış verilerin bulutluşu koleksiyonları.

WHERE  $\leftrightarrow$  If gibi

## SQL Databases

Satır-Sütun var.  
Excel'e benzer  
Tel, bilgisayar, bğyerde  
kullanılır  
High-reliability

## NoSQL Databases

Python'daki dict'de tutulan  
key-value formatına  
benzer  
Cloud boyutlu db'st.

\* What is in a database?

Table = Row + Column

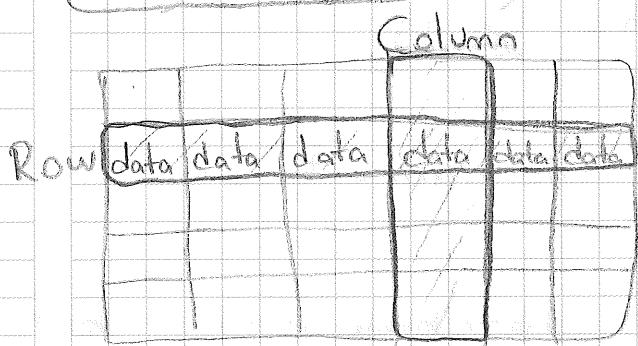


table = relation

row = record  
(satır)

column = attribute  
(sütun)

## SQL Language Elements

Strings

Expressions

Search Conditions

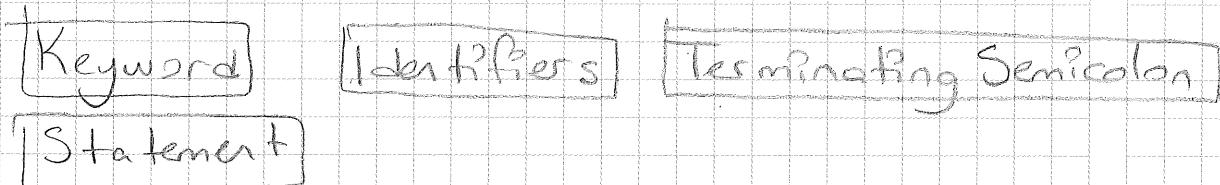
Special Values

Variables

Comments

Birlikte kullanılır.  
Birlikte yazılır.  
Her 5 satırda  
sonra

```
SELECT first_name FROM employees;
```



\* Select'i nasıl kullanırsınız?

Veri çekmek için kullanılır.

Database'den veri çekmek için.

```
SELECT column_name(s) FROM table_name;
```

Table name'ın column\_name(s)'ı那段에 denek

Case-sensitive değildir.

```
SELECT gender FROM employees;
```

### SELECTING MULTIPLE COLUMNS

(Birden fazla sözün (attribute) çekmek)

```
SELECT column1, column2 FROM table1;
```

\* Birlikte columnları nasıl alırız? => \* ile

```
SELECT AlbumId, Title, ArtistId FROM albums;
```

veya  
SELECT \* FROM Albums; (Göktan verilende bu mantıklı)

## DISTINCT Clause

Tekrar edenler birleştirir. (Tekrar tekrar vermez.)

Mesela bir şarkıcı birden fazla şarkı yazmıştır.

Aynı şarkılar ~~de~~ farklı eklenmesini isteyebiliriz.

SELECT'in hemen ordinanı DISTINCT kullanırız.

SELECT DISTINCT student FROM student\_table;

\* Birden fazla satır aynı anda DISTINCT olarak yazılabilir.

SELECT DISTINCT AlbumId, MediaTypeId  
FROM tracks;

## WHERE & LIMIT Clauses

↓  
Filtreleme      Sınırlama

'Where' den sonra bir condition (kısıtlı) belirtmek gereklidir. One göre sonus eklenebilir.

SELECT (column\_name) FROM table-name WHERE  
↓  
condition;  
Bunun yerine \* de olur.

~~SELECT \* FROM tracks WHERE AlbumId=1;~~

AlbumId'i 1 olan klassörler ekrana

### OPERATORS

= → Eşit

> → Büyük

< → Küçük

<> → Eşit değil

>= → Büyük eşit

<= → Küçük eşit

BETWEEN → [15, 20] → 15 ve 20 dahil

LIKE → Sunu gib olanları getir

IN → Bir degerin tabloda yer alıp almadığının kontrolü

\* Find the track names of Jimi Hendrix

SELECT name FROM tracks WHERE Composer = "Jimi Hendrix";

## TASK

Find all the info of the invoices of which total amount is greater than \$10.

```
SELECT * FROM invoices WHERE total > 10;
```

## 1 LIMIT CLAUSES

Kas satının öðemesini istiyorsak bunu limit clause ile belirtiyoruz.

Mesela sýnýye en çok paraþı harcayan bulmak istiyorsunuz. Bunu tek satırla limit clause ile saglayabiliriz. Bütünlerdeki suları istedigini altý gibi sýkýyi kullanmaya yarar. Yapı olarak FROM Sonra sonra WHERE en sonda kullanılır.

```
SELECT column_name FROM table_name LIMIT 3;  
(Tablodaki ilk 3'si getirir.)
```

WHERE - LIMIT aynı anda kullanılabılır fakat WHERE önce kullanılır. Bütün işlemler yapıldıktan sonra limitlene集团股份.

```
SELECT * FROM student_table WHERE grade > 70  
LIMIT 2;
```

70'den büyüklerden ilk 2'sini seç demek.

## TASK

Find all the info of the invoices of which total amount is greater than \$10. Just return the first 4 rows.

```
SELECT * FROM invoices WHERE total > 10  
LIMIT 4
```

ORDER BY,

Alfabetik veya numeric olarak sıralar.

```
SELECT column_name FROM table_name ORDER BY  
column_name ASC/DESC;
```

Ari<sup>n</sup>an  
stra<sup>d</sup>  
↓  
Tos<sup>t</sup>en  
stra<sup>n</sup>

\* SELECT \* FROM employees ORDER BY first\_name ASC;  
(Ari<sup>n</sup>an stra<sup>d</sup>)

\* SELECT \* FROM employees ORDER BY first\_name DESC;  
(Azalon stra<sup>n</sup>)

\* SELECT first\_name, last\_name, salary FROM employees  
ORDER BY salary DESC;  
(Maasları göre azalon sıralama)

\* SELECT column\_name FROM table\_name ORDER BY  
column1 ASC/DESC, column2 ASC/DESC;

Coklu sıralama (Once arası soru azalon)

\* SELECT \* FROM employees ORDER BY gender DESC,  
first\_name ASC;

↳ Sözlük sırası

### BEAUTIFYING (Alt alta yazın)

Okunabilirliği artırın !

Keyword'ları satır satır yazınız.

Böyle yazılımı hafızaya edin.

```
SELECT *
FROM employees
WHERE salary > 80000
```

\* Tablodan 10'dan büyük depleri aralıksız olarak gösterin.

```
SELECT *
FROM invoices
WHERE total > 10
ORDER BY total DESC;
```

↳ Belirtmemesin ASC kabul eder

AND = OR - NOT

WHERE 'job\_title = 'Data\_Scientist' AND gender = 'Male';

Her ikinci de şartın geçerlidir.  
(Kesilim)

SELECT \*

FROM employees

WHERE job\_title = 'Data\_Scientist' OR gender = 'Male';

Birinin sağlanması yeterliければ diğerini askıra.  
(Bırakım)

SELECT \*

FROM employees

WHERE NOT gender = 'Female';

Cinsiyet kodu sağlanır. Buna göre

WHERE gender < > "Female"

WHERE gender = "Male"

değerlendirilir.

Invoices tablosundaki?

(X) Fatura adresi USA olmayanları total amountu  
arter strada göster.

SELECT \*

FROM Invoices

WHERE NOT BillingCountry = "USA"

ORDER BY total ASC;

WHERE BillingCountry <> "USA"

BETWEEN

Sayılar dahil

Mesela maası 15.000 - 50.000 arasındaki kişiler

(Sayılar dahil)

Bunun yerine AND operatörü de kullanılır.

Bir birinin karşılığı

SELECT \*

FROM employees

WHERE salary BETWEEN 8000 AND 9000;

NOT BETWEEN

Maası 15.000 - 50.000 dışındakiler.

(Sayılar dahil değil)

! Tarihler çift tırnakta yerler.

SELECT \*

FROM employees

WHERE hire\_date BETWEEN "2018-06-01" AND "2019-03-31"  
ORDER BY hire\_date;

(\*) Invoices date? 2009-2011 arasında, slyp en yeni değer belirler? (En yeni? istendiği için ASC kullan)

SELECT \*

FROM invoices

WHERE InvoiceDate (Burada istiyor ya deşifreliği)  
BETWEEN "2009-01-01" AND "2011-12-31"  
ORDER BY InvoiceDate ASC → sıraladık ve  
LIMIT 1;

( ) En yeni değer? İzin  
gibi aldık.

Antanadim.

→ Burda trick varmış, Öyle diyor  
hoca

(\*) SELECT Invoice Date

FROM invoices

WHERE InvoiceDate BETWEEN "2009-01-01" AND  
"2011-12-28"

ORDER BY Invoice Date DESC

LIMIT 1;

(2011'e atılmış ki 2012'ye atın  
diyor)

— command (yazın satırı)  
?ki alt tür

✓ Birçok farklı OR yerine IN kullanılır.

✓ \* → Birçok sütunları ? fak eder.

SELECT \* FROM employees ORDER BY  
gender DESC, first\_name ASC;

→ Her DESC her ASC kolumnis

IN

Birçok farklı OR kullanılmış IN içinde  
bunları birleştirir. Daha okunabilir olur.

WHERE 'den sonra kullanılır.

SELECT \*

FROM employees

→ WHERE job\_title IN ('Data Scientist', '...');

Veya

WHERE job\_title = 'Data Scientist' OR job\_title = ...

Dünyası

NOT IN

In'in tersi?

WHERE job-title NOT IN ("Data Scientist", "----")

LIKE

Linux'da \*  
SQL'de %

İçinde "sv" gelenler? getir onları vardır.

SELECT \*

FROM tracks

WHERE Composer LIKE "% Steven Tyler %" → Arada bu gessin

"% Steven" → Sonu Steven la bitsin

" Steven %" → Steven ile başla

- ALT TIRE

- → Bir karakter

% → Birden fazla -biç-hıdıl-şey

Adının adının St ile başladığını biliyorum. Karakteri oldugunu biliyorum ama hatırlıyorum.

SELECT \*

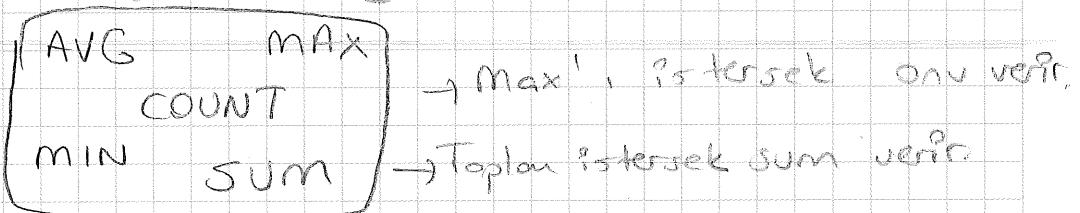
FROM tracks

WHERE Composer LIKE "St\_\_\_\_%"

SELECT'den  
sonra gelir.

## AGGREGATE FUNCTIONS

Makine gibi düşün. Birde fazla değer için atıyorsun. Bir değer istiyorsun.



### SUM and AVG

Sayısal değer kullanmak için dayalı.  
(Numeric values)

### MIN, MAX, COUNT

Numeric

Non-numeric (strings, date, etc.)

} İkisi de olur

### WHAT IS NULL?

} Missing cases

İçinde herhangi bir değer bulundurmeyen nüfular.

Görmesi unutulmaz olabilir.

Baş bir string deşildir. (NULL is not equal to itself.)

Nullar birbirine eşit olmaz. Kendine deşidir.

karakter.

## COUNT FONK. → Sayı

We use COUNT function to count the numbers of records (a.k.a row) in table.

SELECT' da sonra gelir.

SELECT COUNT(name)  
FROM tracks;

→ 3503

SELECT COUNT(name) AS sarki-sayisi

Yazılım ismi yazarsan NULL'ları dahil etmez.

## AS (Alias) KEYWORD

SELECT COUNT(\*) AS count\_result;

FROM tracks

WHERE Composer is NOT NULL;

As is used to rename a column or table with an alias.

Gitti kismina yazdigin sayi yazdirir.

## TASK

Invoices 'da kaç tane music store var?

```
SELECT COUNT(*)
```

```
FROM invoices;
```

→ 412

## COUNT DISTINCT

Birden fazla aynı olan ifadeyi tekrarlamasın ve  
sayısın.

```
SELECT COUNT(DISTINCT Composer)
```

```
FROM tracks;
```

→ 852

↳ Composer var. Buntara NULL dahil değil.

## MIN and MAX

→ NULL' u görmezler.  
Cunku NULL la karslama yapılmaz.

```
SELECT MIN(column_name)
```

```
FROM table_name;
```

→ NULL ları görmez.

`SELECT MIN(salary) AS lowest_salary`

→ lowest\_salary

- - - - -

55000

TASK

Track name'lerden en kısa süreliyi döndür.

`SELECT name, MIN(milliseconds)`  
FROM tracks;

MAX

`SELECT MAX(salary) AS highest_salary`  
FROM employees;

→ highest\_salary

- - - - -

110.000

TASK

Track name'i Festa olan but.

```
SELECT name, MAX(milliseconds)  
FROM tracks;
```

Sum FUNK.

```
SELECT sum(salary) AS total_salary  
FROM employees;
```

→ total\_salary

836000

TASK

Store'umuz ne kadar para kazanmış?

```
SELECT sum(total)  
FROM invoices;
```

## TASK 5

$\%y \Rightarrow \text{yıl gain}$   
 $\%m \Rightarrow \text{ay gain}$

- 1) Invoices tablosundan kaç tane yıl var?

```
SELECT DISTINCT strftime("%Y", InvoiceDate) AS years  
FROM invoices;
```



```
SELECT strftime("%Y", InvoiceDate) AS yillar,  
COUNT(strftime("%Y", InvoiceDate)) AS adet  
FROM invoices  
GROUP BY strftime("%Y", InvoiceDate);
```

→

Yillar	Adet
- - -	- - -
- - -	- - -
- - -	- - -
- - -	- - -

AVG

Average (ort.) bulmak

```
SELECT AVG(milliseconds)  
FROM tracks;
```

② Average'ı bul, average'dan büyük olanları bul.

```
SELECT *  
FROM tracks
```

```
WHERE Milliseconds > (SELECT AVG(milliseconds));
```

Az önceki sondan  
kopyaladık.

Bunun yerine direkt  
sayısal değer  
yazabilirsin.

### GROUP BY

Mesela gender'a göre gruplar. Kime gibi

- SELECT'de yazın isim  
GROUP BY'da da yazınak zanneda  
Yapı olarak FROM'dan sonra gelir.  
WHERE 'de sonra ORDER BY'dan önce gelir

```
SELECT gender, COUNT(gender) ) ikisinde de  
FROM employees  
GROUP BY gender;
```

yazınak  
zanneda  
şenlik

Select > Distinct > Aggregate > From > Where  
Group By > Order By > Limit

Bir değeri bir kez gösterir.

SELECT gender, COUNT(job\_title)  
FROM employees

WHERE job\_title = 'Data Scientist'

GROUP BY gender;

gender	Count(job_title)
Female	1
Male	1

TASK

SELECT Composer, COUNT(name)  
FROM tracks

GROUP BY Composer;

Composer NULL var yazarsan olmaz.



## TASK

Find the total number of each composer's track.

Your result will include name of the composers and number

```
SELECT Composer, count(Composer)  
FROM tracks  
GROUP BY Composer;
```

→ 853 satır döndü

## TASK

How many customers do we have from each country? (Your result will include name of the country and number.)

```
SELECT Country, COUNT(Country)  
FROM customers  
GROUP BY Country;
```

→ 24 satır döndü

SELECT --> Ser ne olsun istiyorum?  
Onu yaz iste :)

Group BY with MIN & MAX

```
SELECT gender, MIN(salary) AS min_salary  
FROM employees  
GROUP BY gender;
```

→ gender min\_salary

Female	67000
Male	55.000

TASK

Find the min duration of track for each album.  
Your result will include album id and min duration.  
(Albumlere en az 5'lik sehip şarkıları göster)

```
SELECT albumId, MIN(milliseconds)  
FROM tracks  
GROUP BY AlbumId;
```

Group BY with SUM & AVG

```
SELECT gender, sum(salary) AS total_salary  
FROM employees  
GROUP BY gender;
```

## TASK

Find the total amount of invoice for each country.  
Your result will include country name and total amount.

SELECT \*

FROM invoices;

→ Yazdik içinde ne var baktik

SELECT Billing Country, sum (total)

FROM invoices

Group By Billing Country;

## JOINS

Tablolari birlestirmeye yarar.



Inner Join



Left Join

Bunlar bilirsen diger jointeri bilmek kolay.

2 ya da daha fazla tabloyu birlestirmek için kullanılır. Bunlar "primary key" ve "foreign key" ile yapılır.

Bir tabloya özel  
unique key'dir.

TC gibi

(Sam bir anoltar resmi var.)

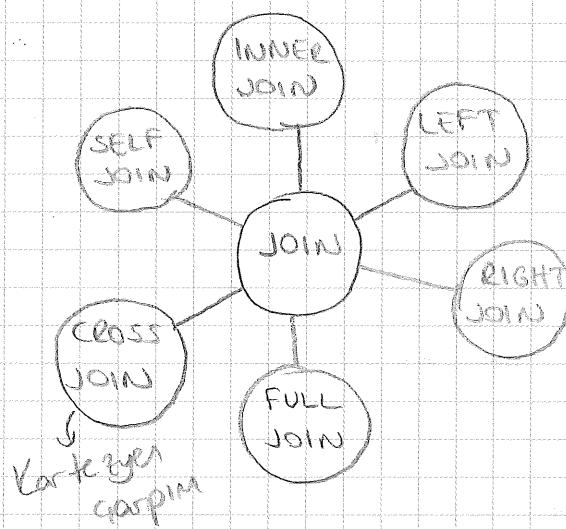
Bir başka tablonun primary  
key'idi.

(Günüş bir anoltar resmi  
var.)

Tablolar arasında bu iki key sayesinde bağlantı kurulur.

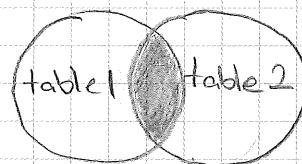
Primary key  $\rightarrow$  Olmak zorunda

Foreign key  $\rightarrow$  Olmak zorunda değil



### INNER JOIN

İki tablonun kesisen kismi kullanılır.



SELECT columns

FROM table-A

INNER JOIN table-B ON join-condition

Primary key  $\Rightarrow$  PK

Foreign key  $\Rightarrow$  FK

Kümelerin kesiştiği  
yer

students

name	exam	score
John	SQL	75
Mary	AWS	80
Clark	Python	60

↓

Aldıkları notlar

tests

exam	passing-score
SQL	70
AWS	80
Python	70
Network	60

↓

Gecme notu.

Bu birleşmeyi inner join ile yaparız.

Inner join Network'ı görmezden gelir. Çünkü ilk tabloda yok. Ama left join'de birleşim kümesi yazılacağı için Network'ı da yazar.

SELECT students.name, students.exam, students.score,  
tests.passing-score

FROM students

INNER JOIN tests ON students.exam = tests.exam;

↳ Her sonda  
yazılır.

Students tablosundan name? al demek.

Students tablosundan  
exam al demek

Tests tablosundan  
passing-score al.

output of the query

name	exam	score	passing-score
John	SQL	75	70
Mary	AWS	80	80
Clark	Python	60	70

### TASK

Find the genre of each track.

"Once GenresId'ının içinde ve track tablalarında ne var galistirarak bak, sonra kodu yaz.

SELECT \* FROM genres; → yaz icine bi bak neler var.

SELECT \* FROM tracks; → " " " "

Neler konsiyorsa kopya onun istinden kinalasak.

SELECT tracks.Name , genres.Name

FROM tracks

INNER JOIN genres ON tracks.GenreId = genres.GenreId

## TASK

Find the customer name of each invoice. Your result will include invoice id and customer name.

SELECT \* FROM customer;

SELECT \* FROM invoices;

→ Pekişinde ortak customer Id var. Bağlantı ordan kurulmuş.

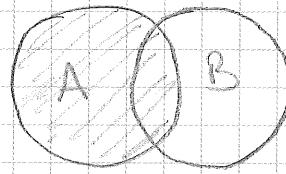
SELECT invoices.InvoiceId, customers.FirstName,  
customers.LastName

FROM invoices → istersen customer da yazarılmıştır. Pekişini birleştiriyorum.

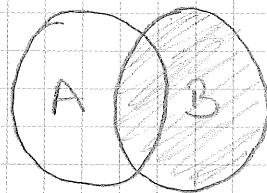
INNER JOIN customers ON invoices.CustomerId =

customers.CustomerId;

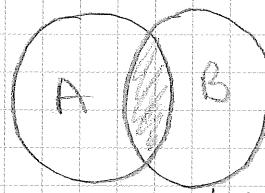
## SQL JOINS



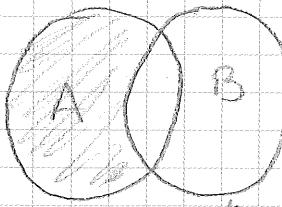
```
SELECT * FROM  
A [LEFT] JOIN B  
ON A.KEY=B.KEY
```



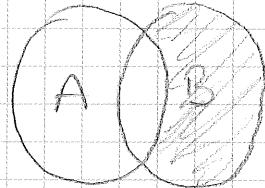
```
SELECT * FROM  
A [RIGHT] JOIN B  
ON A.KEY=B.KEY
```



```
SELECT * FROM  
A [INNER] JOIN B  
ON A.KEY=B.KEY
```



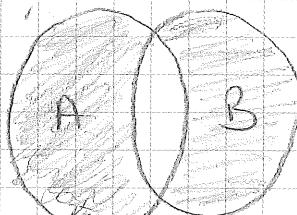
```
SELECT * FROM  
A [LEFT] JOIN B  
ON A.KEY=B.KEY  
WHERE B.KEY IS NULL
```



```
SELECT * FROM  
A [RIGHT] JOIN B  
ON A.KEY=B.KEY  
WHERE A.KEY IS NULL
```



```
SELECT * FROM  
A [FULL OUTER] JOIN B  
ON A.KEY=B.KEY
```



```
SELECT * FROM  
A [FULL OUTER] JOIN B ON  
A.KEY=B.KEY WHERE A.KEY  
IS NULL OR B.KEY IS NULL
```

## LEFT JOIN

RIGHT'da kullandığımız not tablasını kullanacağız.

SELECT tests.exam, tests.passing-score,

students.name, students.score

FROM tests

LEFT JOIN students ON tests.exam=students.exam;



Soldaki tabloyu değiştirmeyiz. Sağdakini sone ekle.

Boslukları da NULL ile doldurur.

## TASK

Find the artists' album info.

(Artistten albümlelerin bilgisi isteniyor.)

(ArtistId, ArtistName, AlbumId gibi getir.)

SELECT \* FROM albums; } ArtistId İleinden

SELECT \* FROM artists; } bağlantı kuracağınız.

SELECT artis.ArtistId, artis.name, alb.AlbumId

FROM artists artis

/ artists de yazarım

LEFT JOIN albums alb ON artis.ArtistId=alb.

ArtistId;

! Okunabilirliği artırma adına tablolarda AS kullanılabillir. Sütunlarda kullanılır.

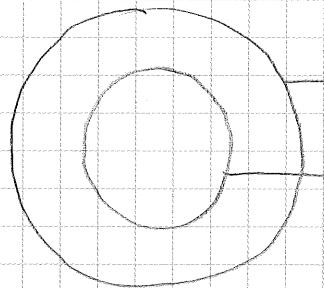
## SUBQUERIES

İç içe queriler demek.

```

SELECT - , } FROM JOIN
FROM+JOIN; } How the query
WHERE } is written
GROUP BY } ...
HAVING } (Böyle yazılış) --- SELECT
ORDER BY } ORDER BY
LIMIT } LIMIT
  
```

How you  
should  
think about  
it  
(Böyle  
içler)



Outer query } İçi sistem tek  
sorguda yapılıabilir.  
Inner query }

```

SELECT column-name
FROM table-1, table-2
WHERE column-name OPERATOR
  
```

→ Outer query or  
enclosing query

```

(SELECT column-name
FROM table-1,table-2);
  
```

→ Inner query,  
nested query  
or subquery

A subquery may be used in:

- ① SELECT clause
- ② FROM clause
- ③ WHERE clause

### Single-Row Subqueries

```
SELECT first-name, last-name, salary } Burden moos,  
FROM employees } butor  
WHERE salary > (SELECT salary } Burden da  
FROM employees } ?sm? butor.  
WHERE first-name = "Rodney");
```

```
SELECT first-name, last-name, salary  
FROM employees  
WHERE salary > (SELECT AVG(salary)  
FROM employees);
```

## TASK

Retrieve track id ,track name, album id info of the  
Album title Faceless!

```
SELECT *  
FROM albums  
WHERE Title = "Faceless";
```

```
SELECT *  
FROM tracks;
```

1. YOL  
SELECT TrackId, name, AlbumId  
FROM tracks

WHERE AlbumId = (SELECT AlbumId

veya

FROM albums

WHERE Title = "Faceless")

2. YOL,

```
SELECT tracks.TrackId, tracks.name, albums.AlbumId  
FROM tracks
```

```
INNER JOIN albums ON tracks.AlbumId = albums.AlbumId  
WHERE albums.Title = "Faceless";
```

## 1 TASK

Ortalama Id 'ten 340.000 'in üzerinde olanları  
getir.

```
SELECT AlbumId, avg(Milliseconds) as duration  
FROM tracks  
GROUP BY AlbumId  
HAVING duration > 340.000  
ORDER BY duration;
```

⚠️ Aggregate fonksiyonları ?'ın filtrelenme yapacak-  
sak 'HAVING' kullanılır.

WHERE ile birlikte kullanılmaz.

### WHERE

Filtrelenme yapar  
Aggregate fonk.'ları,  
filtrelemez.

### HAVING

Filtrelenme yapar.  
Aggregate fonk.'ları  
filtreler.

⚠️ Hem ortalama ... istiyorsa (AVG)

\* hem WHERE kullanıcağımızda WHERE kullanamayız.

Bu yüzden

HAVING + AVG

uullanıyoruz

Sonra olacak  
bir değer getir.

## Single-row Subqueries

Single-row subqueries return one row with only one column and are typically used with single-row operators such as =, >, >=, <>, != especially in WHERE clause.

### TASK

Tracks albümünün içindeki sarkıların milliseconds'ının ortalaması 340.000'den fazla olanları getir.  
(Bu WHERE ile yapamazsın, çünkü AVG ile WHERE kullanılmaz.)

```
SELECT AlbumId, avg(Milliseconds)
FROM tracks
GROUP BY AlbumId
HAVING AVG(Milliseconds) > 340.000;
```

### TASK

Find out the employees who get paid more than the average salary

```
SELECT first_name, last_name, salary
```

```
FROM employees
```

```
WHERE salary >
```

```
(SELECT AVG(salary)
```

```
FROM employees);
```

## Multiple -row Subqueries

Sonuç olarak birden fazla değer gelir.

Multiple-row subqueries return sets of rows  
and are used with multiple-row operators such as  
IN, NOT IN, ANY, ALL. (Bunda = kullanılmıyor.)

### EXAMPLE

Find the employees (first name, last name from  
employees table) who work under the Operations  
department (department table)

② [ SELECT first\_name, last\_name  
FROM employees  
WHERE emp\_id IN

① [ (SELECT emp\_id  
FROM departments  
WHERE dept\_name = 'Operations');

} Sonra bordan  
çalışanların  
kimlikleriyle  
ismi soyismi  
yazdırılır.

} Önce bunuyla  
operations'ları  
seçer.

## TASK

Retrieve track Id , track name , album Id info  
of the Album title "Faceless" and 'Let There  
Be Rock'

SELECT \* } Bu tabloya bir bak  
FROM albums; } neler var.

Buna da } SELECT \*  
bask } FROM albums  
Burdan 2 deger } WHERE Title IN ("Faceless", "Let There Be Rock")  
gelenek O yuzden esittir degil IN kullanacagiz. Hangisine esitleyecegini bilmez,

SELECT TrackId, Name, AlbumId  
FROM tracks

WHERE AlbumId IN (SELECT AlbumId

FROM albums

WHERE Title = "Faceless"

or Title = "Let There Be Rock")

## 1 DDL Commands

Data Definition Language

(Tablolari olustur, degistir, sil islemi yapilir.)

Ven tabanimizin alt yapisini olusturmaya alakali.

Database'in en gor kismi tablolari olusturmak.

Some statements used in DDL are:

CREATE (olustur)

ALTER (degistir)

DROP (sil)

## 2 Data Manipulation Language (DML)

Tabloya veri giris → INSERT

Guncelleme → UPDATE

Silme → DELETE

Tablodan veri → SELECT\*

} are the statements used in DML

DDL → Data Definition Language

DML → Data Manipulation Language

DCL → Data Control Language

TCL → Transaction Control Language

## Data Types

String }  
Numeric  
Date and Time

3 tane veri tipi var

CHAR(10) yazarsan  
5 karakter 5 yaşasın  
bile yazsan 255 tür.  
yer

The string data types are : CHAR

VARCHAR

VARCHAR(10) dersen  
karakter 10 i asamağı.  
5 karakter olursa yazarsan  
5 karakter ol boyle tutar.

TEXT

Karakter sınırları yok  
Belli etmemek gerek yok

## CREATE TABLE

When creating a table , we use CREATE TABLE statement.

CREATE TABLE table-name  
(column-name1 data-type,  
column-name2 data-type),

3

## DCL

GRANT → Birine yetki vermek

REVOKE → Yetki kısıtlamak

## EXAMPLE 1

```
CREATE TABLE customers2  
    (customer_id INT,  
     first_name TEXT,  
     last_name TEXT);
```

## EXAMPLE 1

```
CREATE TABLE employee  
    (first_name VARCHAR(15),  
     last_name VARCHAR(20),  
     age INT,  
     hire_date DATE);
```



Values in VARCHAR columns are variable-length strings. The length can be specified as a value from 0 to 65,535.



DROP TABLE IF EXISTS employee;

Eğer employee adında bir table varsa onu sil demek

DROP TABLE employee de díyebilirim ama bu table yoksa hata vere.

## TASK

vacation\_plan adında bir table oluştur.

Column names: place\_id

country

hotel\_name

employee\_id

vacation\_length

budget

CREATE TABLE vacation\_plan

(place\_id INT, → NALC HAIR  
country TEXT, ya tabii  
hotel\_name TEXT,  
employee\_id INT,  
vacation\_length INT,  
budget INT);

6 tane sütun oluşturduk

④

## TCL

COMMIT ➔ Yapılan değişikliği kalıcı hale getirir.

SAVEPOINT ➔ Geçici kayit (Staging Area gibi)

ROLLBACK ➔ SAVEPOINT'e geri dönüs.

▼ Oluşturduğumuz tabloyu silmek:

DROP TABLE table-name;

Tabloyu tamamen sile

TRUNCATE TABLE table-name;

Y Tablonun içini boşaltır, tablo kalır.  
Ama SQL bu kodu tanıtmaz.

TASK

Aşağıdaki sorudaki vacation\_planı silelim.

DROP TABLE vacation\_plan;

INSERT INTO

INSERT INTO table-name(column1, column2, ....)  
VALUES (value1, value2, ....),  
        (value1, value2, ....);

(\*) column1'e value1 gelecek demek.

! Bir tablo oluşturulduktan sonra o tabloya veri eklemek için insert INTO kullanılır.

Mesela hepsi burada kişi musteri mail adresini, şifresini yazdı. Aşka planlı bu bilgileri database' e ekleyecek olan komut bu.

## EXAMPLE

```
CREATE TABLE customer2
```

```
(customer_id INT,  
first_name TEXT,  
last_name TEXT);
```

Bunu  
önce yapmamak

```
INSERT INTO customers2 (customer_id, first_name,  
last_name)
```

```
VALUES (1, "ahmet", "mehmet"),  
(2, "samer", "mer");
```

↓  
(Veri? görüntüle komşudan da  
depistiklik yapabilişin.)

## CONSTRAINTS

→ Bu kişiye kullanıcı burayı doldurmadan sonraki  
sayfa'a geçemez.

**NOT NULL** → Ensures that a column cannot have a **NULL**  
value

**DEFAULT** → Sets a default value for a column when  
no value is specified

**UNIQUE** → Ensures that all values in a column are  
different

**PRIMARY KEY** → Uniquely identifies each row in a table

**FOREIGN KEY** → Uniquely identifies a row/record  
in another table.

PRIMARY KEY → Ana tablo (NULL olamaz)  
FOREIGN KEY → Bağlantılı tablolar.

Y Primary Key tablolar arasındaki ilişkiyi sağlar.  
Primary key' e başka bir tablonun foreign key' i ile ulaşabiliyoruz. Amacı, tablolalar arası ilişkileri sağlamak.

PRIMARY KEY → unique

CREATE TABLE table-name (  
 column\_1 INT PRIMARY KEY,  
 column\_2 TEXT,  
 ...  
);

iki kullanımı  
var.

iki de  
aynt.

Artık column\_1  
primary key oluyor.

CREATE TABLE table-name (  
 column\_1 INT,  
 column\_2 TEXT,  
 ...  
 PRIMARY KEY (column\_1)  
);

tables

## FOREIGN KEY

customers

```
CREATE TABLE customers (customer_id INT PRIMARY KEY,  
first_name TEXT,  
second_name TEXT);
```

orders

```
CREATE TABLE orders (  
order_id INT PRIMARY KEY,  
order_number INT,  
customer_id INT,  
FOREIGN KEY (customer_id)  
REFERENCES customers (customer_id))
```

;

Amacı: Yukardaki customer\_id yi al  
foreign key olarak tanımlama-  
şını yap.

## EXAMPLE

```
CREATE TABLE customers3 (
```

```
customer_id INT PRIMARY KEY,  
first_name VARCHAR(25),  
second_name TEXT);
```

```
CREATE TABLE orders (
```

```
order_id INT,
```

```
order_number INT,
```

```
customer_id INT,
```

```
PRIMARY KEY (order_id),
```

→ FOREIGN KEY (customer\_id)  
REFERENCES customers3 (cus-  
tomer\_id));

Pk

satırda da  
yazılabilir.

## NOT NULL

Bir verinin ıçında oblu olmasının istiyorsak  
NOT NULL kullanırız.

Primary key ile NOT NULL'a gerek yok.  
Çünkü kendinden NOT NULL.

## TASK

Please drop the table as you've just created  
writing `DROP TABLE vacation-plan;`

Then, recreate the vacation-plan table  
adding constraints as below:

Column names: place\_id → PRIMARY KEY

country

hotel-name → NOT NULL

employees → FOREIGN KEY

vacation-length

budget

## NOT NULL

Bir verinin ıçında oblu olmasının istiyorsak  
NOT NULL kullanırız.

Primary key ile NOT NULL'a gerek yok.  
Çünkü kendinden NOT NULL.

## TASK

Please drop the table as you've just created  
writing `DROP TABLE vacation-plan;`

Then, recreate the vacation-plan table  
adding constraints as below:

Column names: place\_id → PRIMARY KEY

country

hotel-name → NOT NULL

employees → FOREIGN KEY

vacation-length

budget

DROP TABLE IF EXISTS vacation\_plan;

CREATE TABLE vacation\_plan(  
 place\_id INT PRIMARY KEY,  
 country TEXT,  
 hotel\_name VARCHAR(20) NOT NULL,  
 EmployeeId INT,  
 Vacation\_length INT,  
 budget INT  
 FOREIGN KEY (EmployeeId)  
 REFERENCES employees(EmployeeId);

## ALTER TABLE

Bir tabloyu değiştirmek için kullanılır.

Bir tabloya yeni bir sütun eklemek için  
su yapıılır:

```
ALTER TABLE table-name  
ADD column-name data-type;
```

### EXAMPLE

```
ALTER TABLE vacation-plan  
ADD city TEXT;
```



Bir tablodaki sütunu silme:

```
ALTER TABLE table-name  
DROP column-name;
```



Bir tablodaki sütunun data tipini değiştirmek:

SQL  
modify'yi  
tanımıyor.

```
ALTER TABLE table-name  
MODIFY COLUMN column-name data-type
```

## EXAMPLE

ALTER TABLE vacation\_plan  
DROP vacation\_length;

} vacation\_length  
silindi.

## EXAMPLE

ALTER TABLE vacation\_table  
MODIFY budget DOUBLE;

} Yanlış yazmışım  
DOUBLE olarak  
değiştirmek  
isteyorum.

Baska database sistemlerinde kullanı-  
bilirim. Sadece hata verir. Kullanıldığın  
database dokümanını kullanırken okumakta  
fayda var.

Database'lerin en büyük amacı tablolar arası iliş-

ki kılendirmeye. Mesela Amazon bir yıl içinde en  
çok hangi ürün satmış, tablolar arası bağlantı  
yaparak bunu buluyor. Buradan para kazanır  
sağlıyor. Geride database'in amacı paradise.

Bütün tabloları birleştirmek karışıklığa yol  
olar. Bu yüzden aynı tablolar yapılıp buları ilişk-  
i kılendirmek daha kolaydır. Primary key ve foreign key bunun  
için var.

