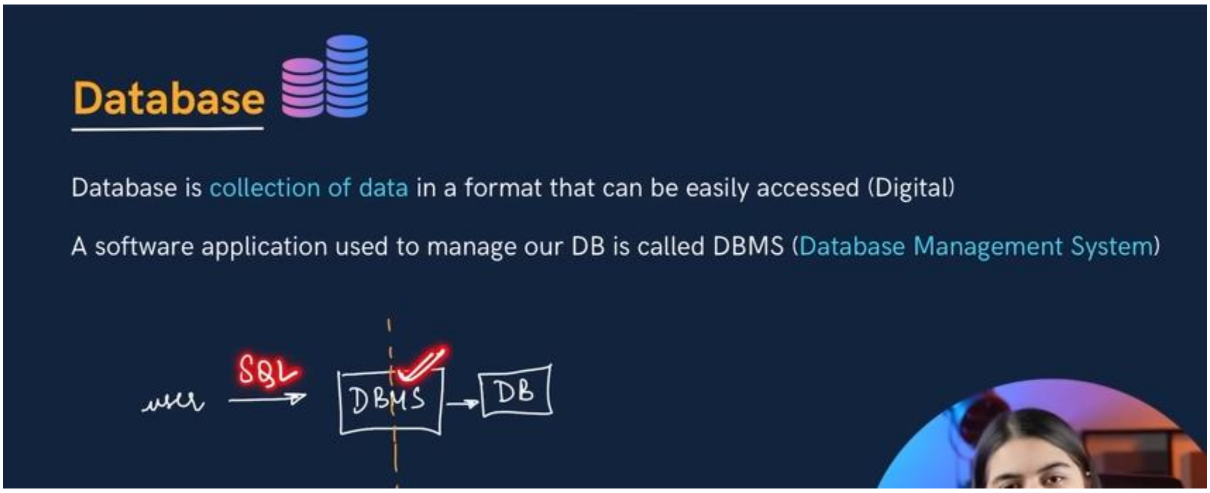
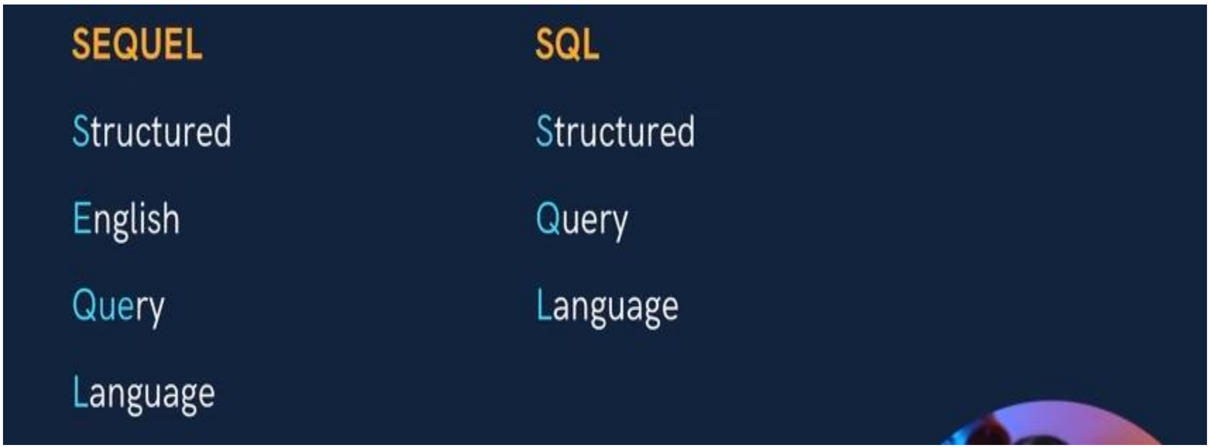


Database

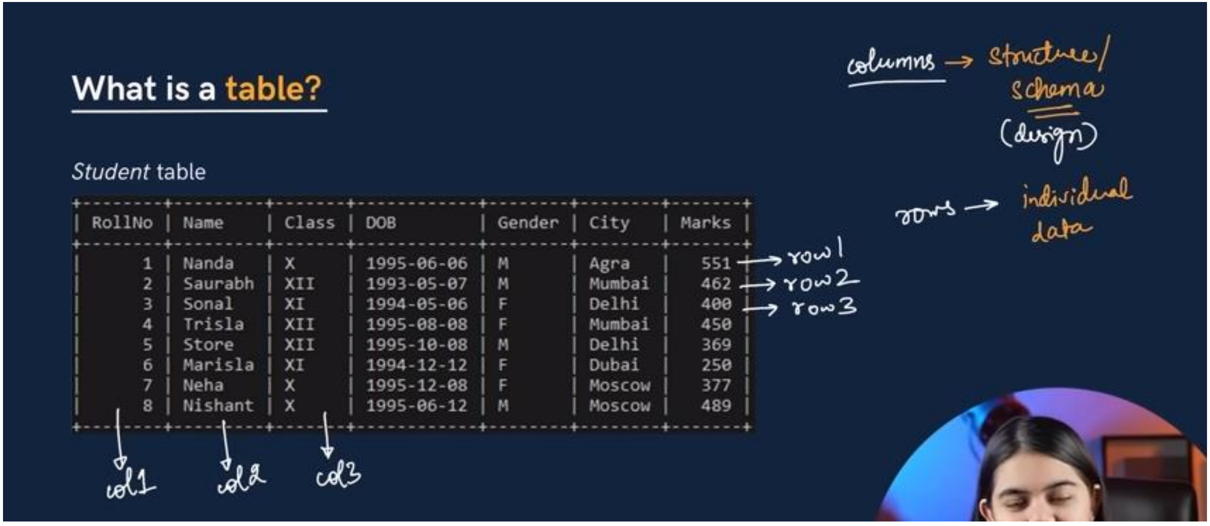
Database is collection of data in a format that can be easily accessed (Digital)

A software application used to manage our DB is called DBMS (Database Management System)

Earlier SQL was known as SEQUEL, developed by IBM.



SEQUEL	SQL
Structured	Structured
English	Query
Query	Language
Language	

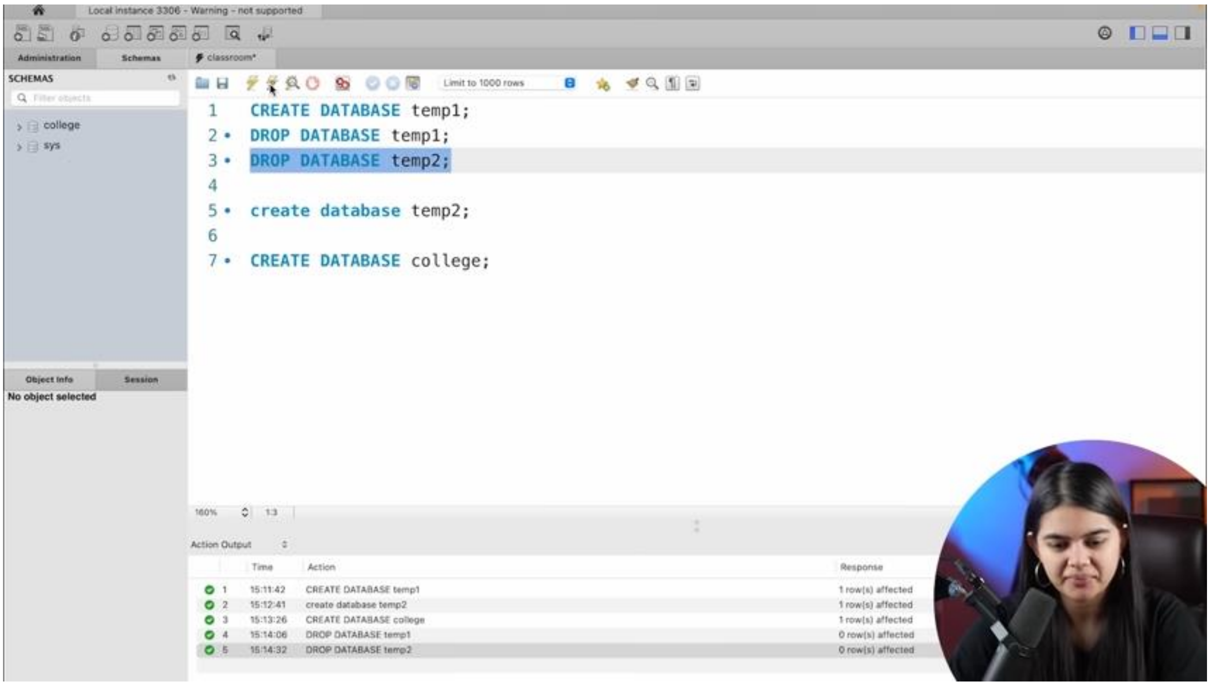


What is a table?

Student table

```
+--------+---------+-------+------------+--------+--------+-------+
| RollNo | Name    | Class | DOB        | Gender | City   | Marks |
+--------+---------+-------+------------+--------+--------+-------+
|      1 | Nanda   | X     | 1995-06-06 | M      | Agra   |   551 |
|      2 | Saurabh | XII   | 1993-05-07 | M      | Mumbai |   462 |
|      3 | Sonal   | XI    | 1994-05-06 | F      | Delhi  |   400 |
|      4 | Trisla  | XII   | 1995-08-08 | F      | Mumbai |   450 |
|      5 | Store   | XII   | 1995-10-08 | M      | Delhi  |   369 |
|      6 | Marisla | XI    | 1994-12-12 | F      | Dubai  |   250 |
|      7 | Neha    | X     | 1995-12-08 | F      | Moscow |   377 |
|      8 | Nishant | X     | 1995-06-12 | M      | Moscow |   489 |
+--------+---------+-------+------------+--------+--------+-------+
```

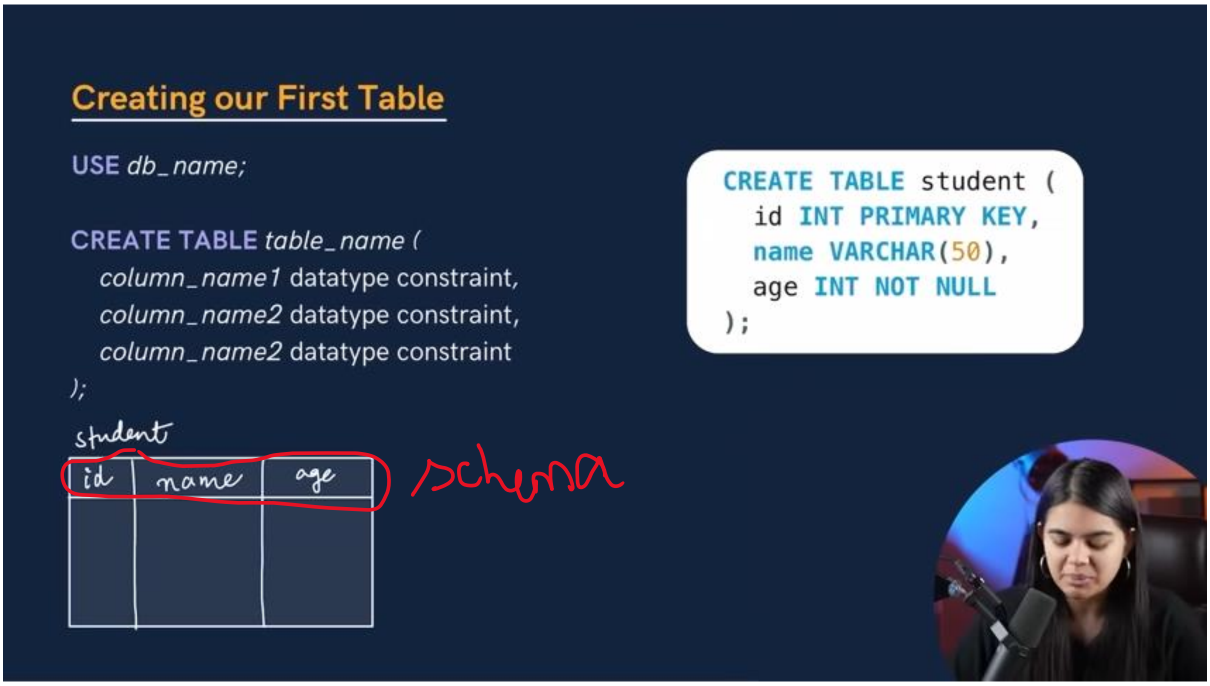
SQL is case insensitive but datas are case sensitive.



```sql
CREATE DATABASE temp1;
DROP DATABASE temp1;
DROP DATABASE temp2;

create database temp2;

CREATE DATABASE college;
```

Action Output

```
     Time      Action                     Response
1    15:11:42  CREATE DATABASE temp1      1 row(s) affected
2    15:12:41  create database temp2      1 row(s) affected
3    15:13:26  CREATE DATABASE college    1 row(s) affected
4    15:14:06  DROP DATABASE temp1        0 row(s) affected
5    15:14:32  DROP DATABASE temp2        0 row(s) affected
```



Creating our First Table

```
USE db_name;

CREATE TABLE table_name (
    column_name1 datatype constraint,
    column_name2 datatype constraint,
    column_name2 datatype constraint
);
```

```sql
CREATE TABLE student (
  id INT PRIMARY KEY,
  name VARCHAR(50),
  age INT NOT NULL
);
```

student

id	name	age

schema

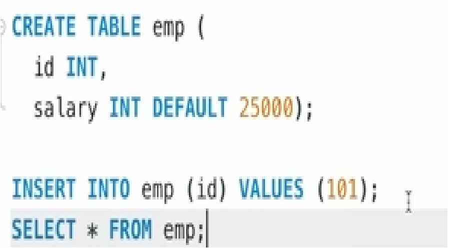
varchar consumes only that part of memory that satisfies its length.

Bit(1) implies we can only store 0 or 1

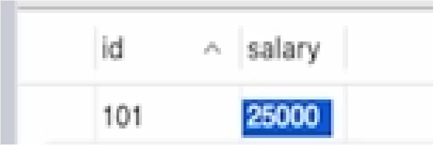
Bit(2) implies we can only store 00, 01, 10, 11

MySQL does not contain built-in Boolean or Bool data type. They provide a **TINYINT** data type instead of Boolean or Bool data types. MySQL considered value zero as false and non-zero value as true. If you want to use Boolean literals, use true or false that always evaluates to 0 and 1 value.

Default constraint



```
CREATE TABLE emp (
  id INT,
  salary INT DEFAULT 25000);

INSERT INTO emp (id) VALUES (101);
SELECT * FROM emp;
```



id	salary
101	25000

create table student(

rollno int primary key,

name varchar(50),

marks int not null,

grade varchar(1),

city varchar(20)

);

insert into student

(rollno, name, marks, grade, city)

values

(101, "anil", 78, "C", "Pune"),

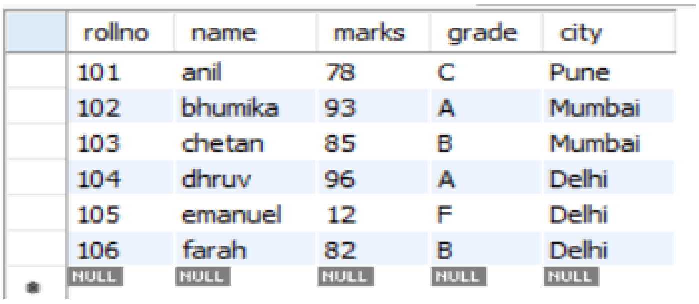
(102, "bhumika", 93, "A", "Mumbai"),

(103, "chetan", 85, "B", "Mumbai"),

(104, "dhruv", 96, "A", "Delhi"),

(105, "emanuel", 12, "F", "Delhi"),

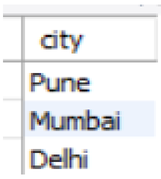
(106, "farah", 82, "B", "Delhi");



rollno	name	marks	grade	city
101	anil	78	C	Pune
102	bhumika	93	A	Mumbai
103	chetan	85	B	Mumbai
104	dhruv	96	A	Delhi
105	emanuel	12	F	Delhi
106	farah	82	B	Delhi
NULL	NULL	NULL	NULL	NULL

Distinct

select distinct city from student;



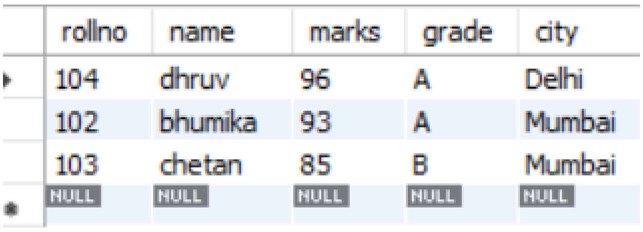
city
Pune
Mumbai
Delhi

to get marks of top 3 students

select *

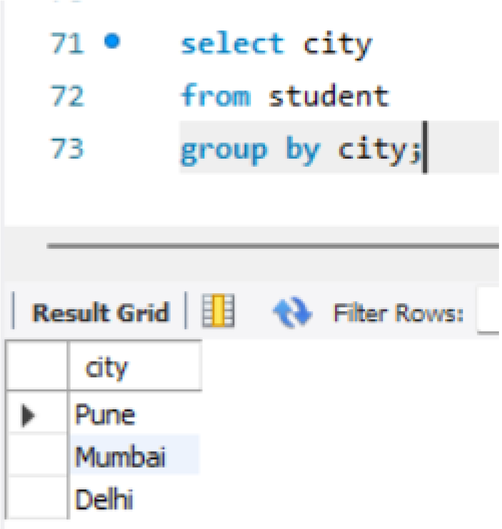
from student

order by marks desc limit 3;



rollno	name	marks	grade	city
104	dhruv	96	A	Delhi
102	bhumika	93	A	Mumbai
103	chetan	85	B	Mumbai
NULL	NULL	NULL	NULL	NULL

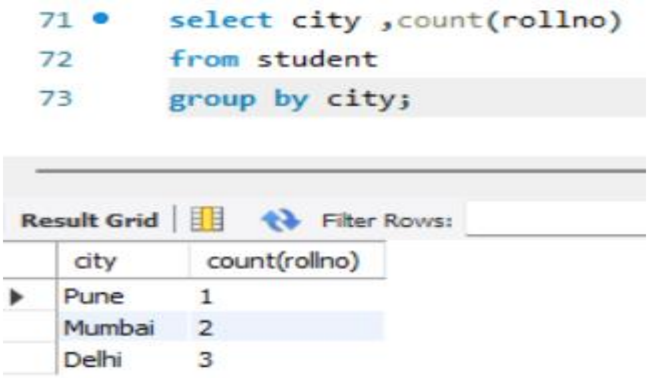
Group by



```
select city
from student
group by city;
```

city
Pune
Mumbai
Delhi

Gives no of student in each city. Groups on the basis of unique city



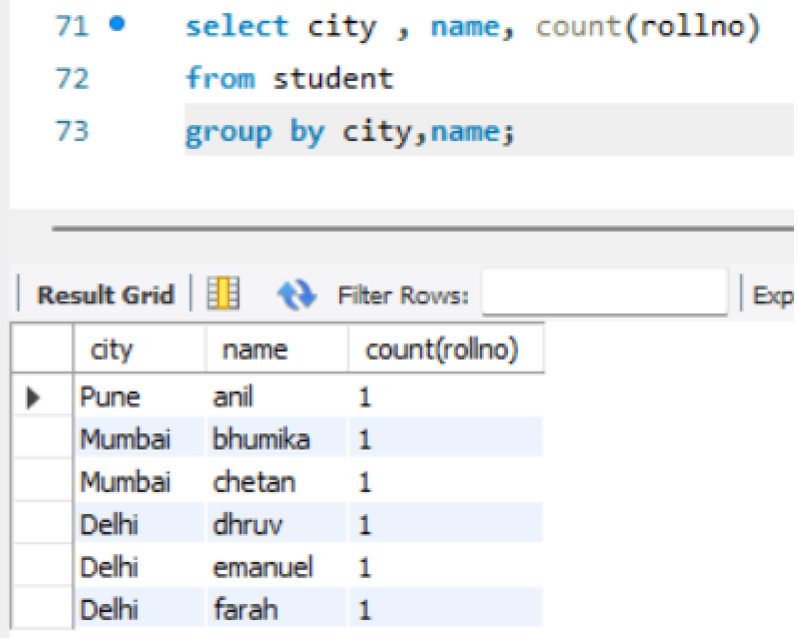
```
select city ,count(rollno)
from student
group by city;
```

city	count(rollno)
Pune	1
Mumbai	2
Delhi	3

Note: select city, name, count(rollno) -- **this will give error as you can only select those columns (which are not in aggregate function) which are used with group by clause**

from student -- so you can't use name directly

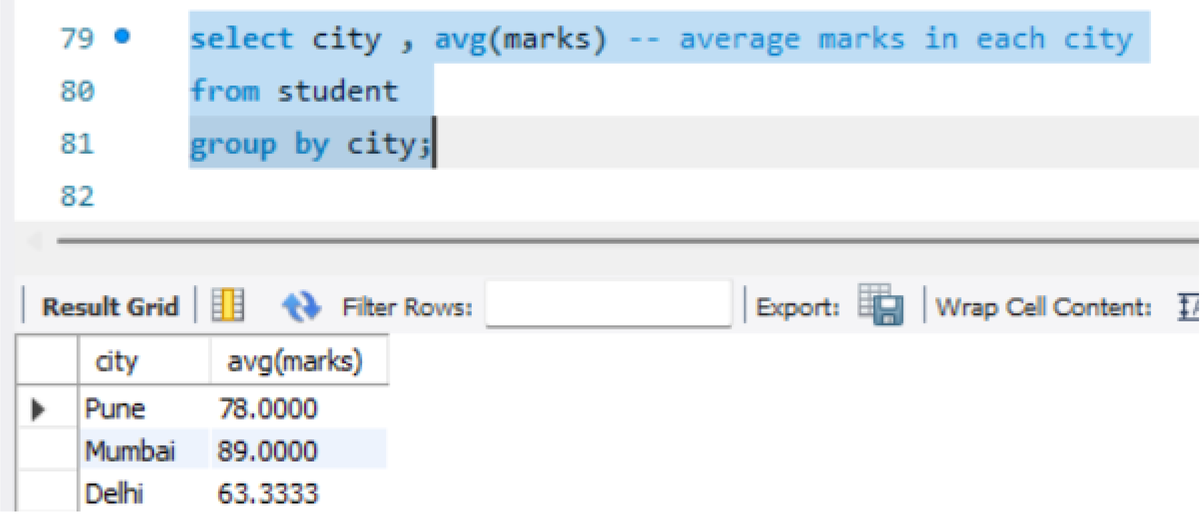
group by city;



```sql
select city , name, count(rollno)
from student
group by city,name;
```

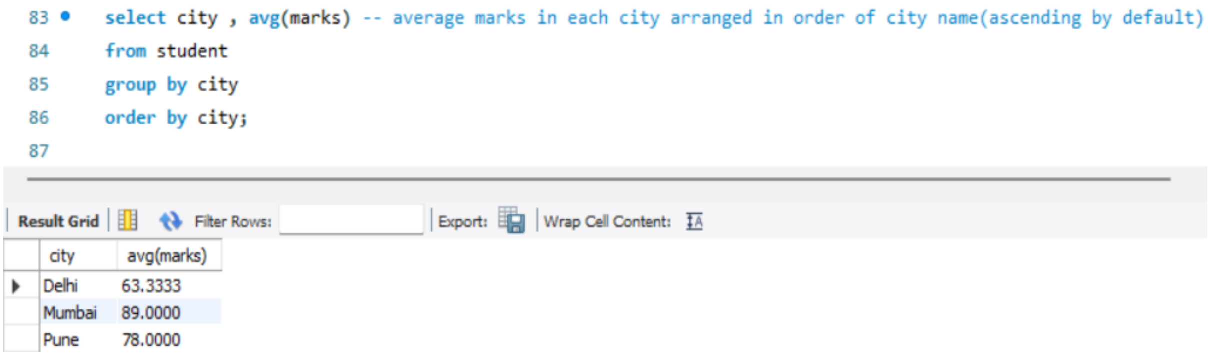
city	name	count(rollno)
Pune	anil	1
Mumbai	bhumika	1
Mumbai	chetan	1
Delhi	dhruv	1
Delhi	emanuel	1
Delhi	farah	1

This(above) is correct. Groups on the basis of unique city and names. If there were 2 Chetan leaving in Mumbai, then count would have been 2.



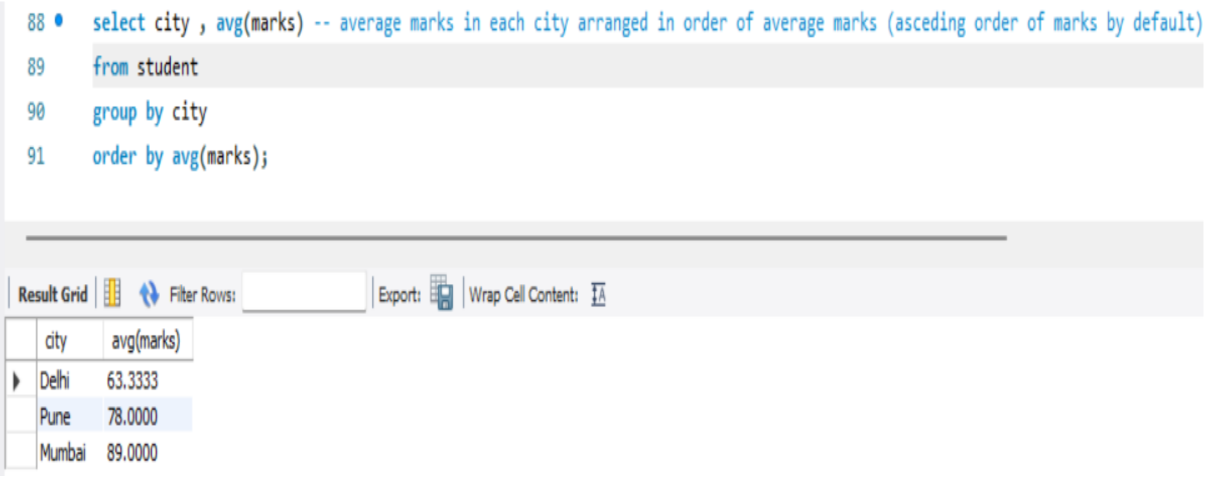
```sql
select city , avg(marks) -- average marks in each city
from student
group by city;
```

city	avg(marks)
Pune	78.0000
Mumbai	89.0000
Delhi	63.3333



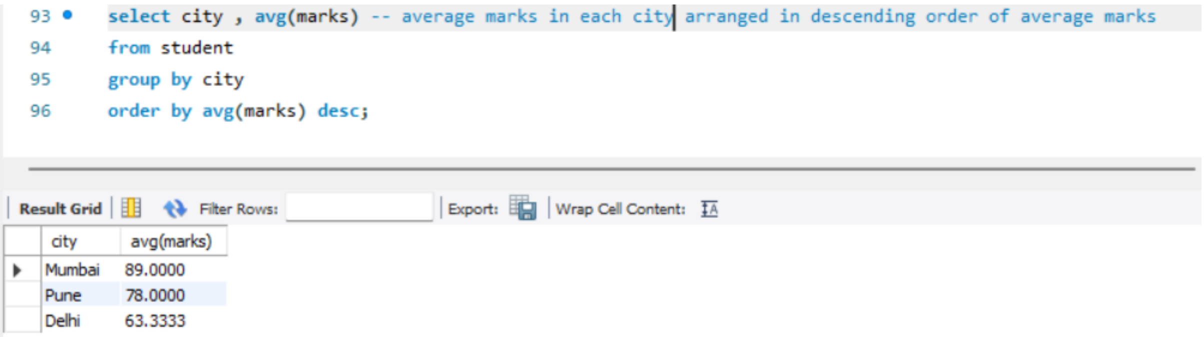
```sql
select city , avg(marks) -- average marks in each city arranged in order of city name(ascending by default)
from student
group by city
order by city;
```

city	avg(marks)
Delhi	63.3333
Mumbai	89.0000
Pune	78.0000



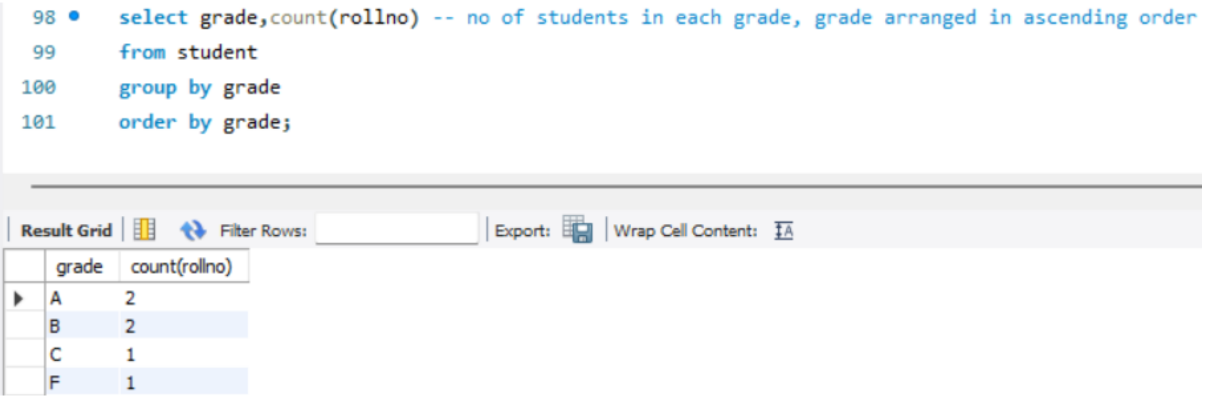
```sql
select city , avg(marks) -- average marks in each city arranged in order of average marks (asceding order of marks by default)
from student
group by city
order by avg(marks);
```

city	avg(marks)
Delhi	63.3333
Pune	78.0000
Mumbai	89.0000



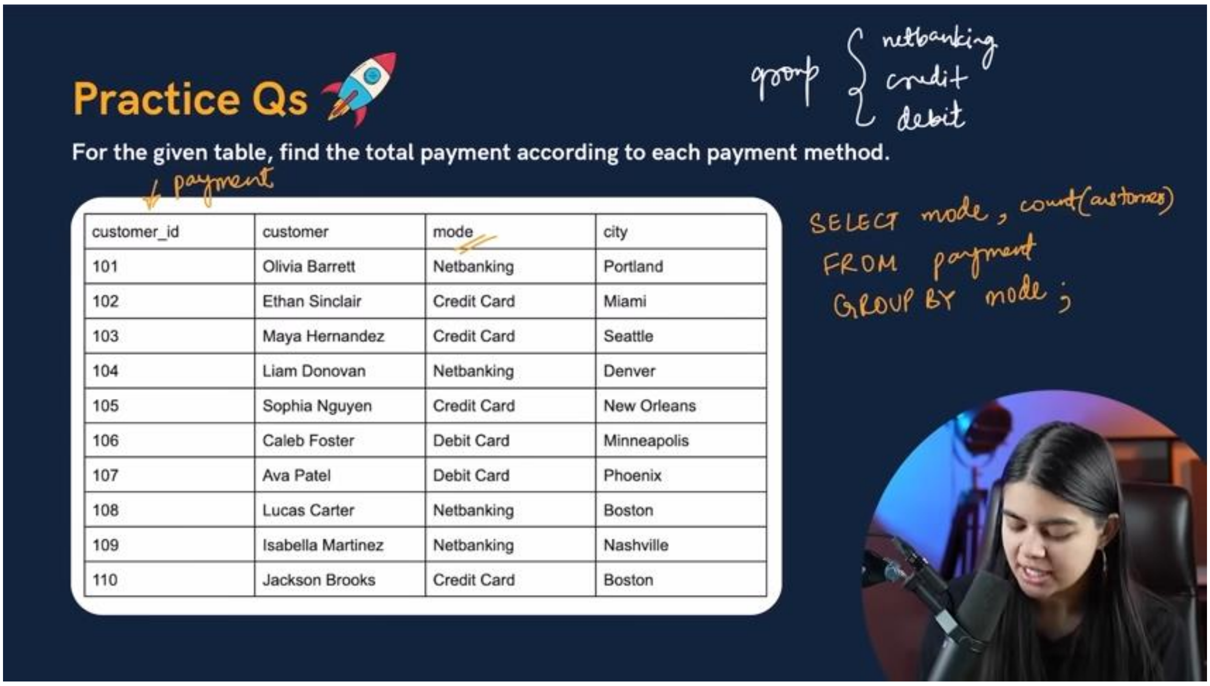
```sql
select city , avg(marks) -- average marks in each city arranged in descending order of average marks
from student
group by city
order by avg(marks) desc;
```

city	avg(marks)
Mumbai	89.0000
Pune	78.0000
Delhi	63.3333



```sql
select grade,count(rollno) -- no of students in each grade, grade arranged in ascending order
from student
group by grade
order by grade;
```

grade	count(rollno)
A	2
B	2
C	1
F	1



Practice Qs

group { netbanking
credit
debit

For the given table, find the total payment according to each payment method.

payment

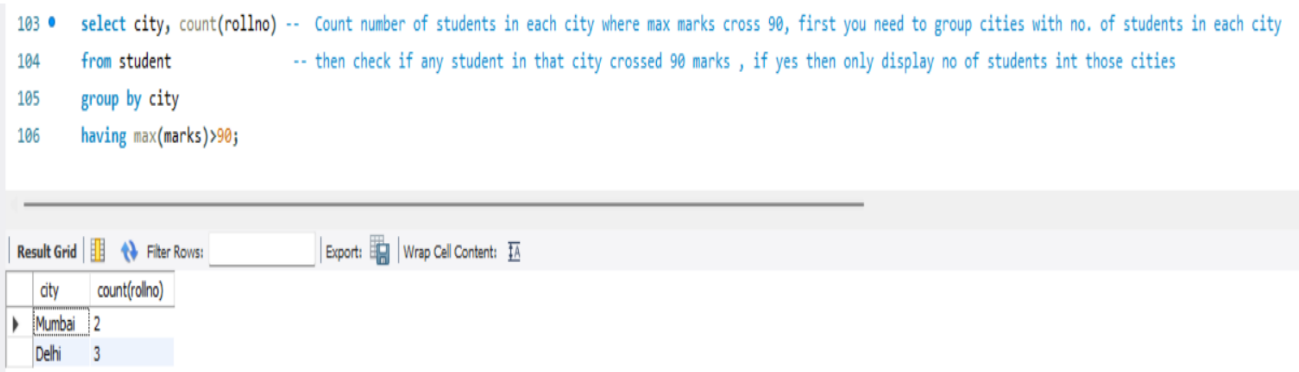
customer_id	customer	mode	city
101	Olivia Barrett	Netbanking	Portland
102	Ethan Sinclair	Credit Card	Miami
103	Maya Hernandez	Credit Card	Seattle
104	Liam Donovan	Netbanking	Denver
105	Sophia Nguyen	Credit Card	New Orleans
106	Caleb Foster	Debit Card	Minneapolis
107	Ava Patel	Debit Card	Phoenix
108	Lucas Carter	Netbanking	Boston
109	Isabella Martinez	Netbanking	Nashville
110	Jackson Brooks	Credit Card	Boston

```
SELECT mode, count(customers)
FROM payment
GROUP BY mode;
```

Given table name: Payment,

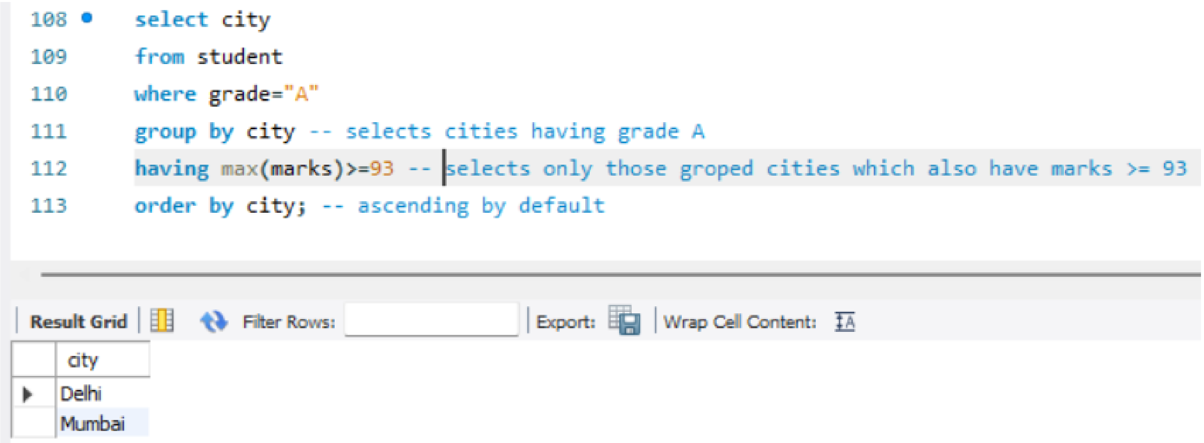
count(customer)/count(customer_id) gives no. of customer; better to use 2nd one.

having clause



```
select city, count(rollno) -- Count number of students in each city where max marks cross 90, first you need to group cities with no. of students in each city
from student               -- then check if any student in that city crossed 90 marks , if yes then only display no of students int those cities
group by city
having max(marks)>90;
```

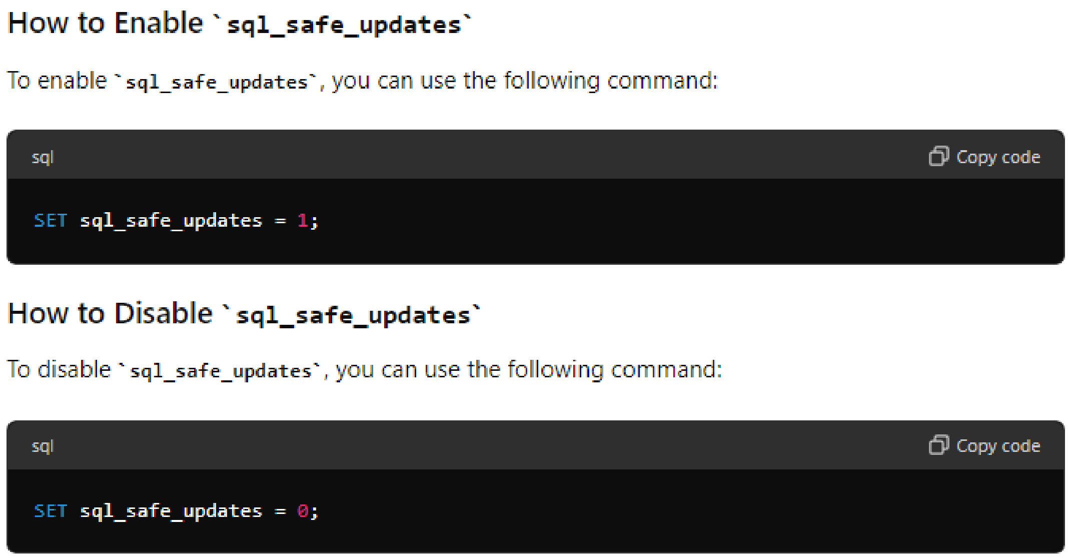
```
city    count(rollno)
Mumbai  2
Delhi   3
```



```
select city
from student
where grade="A"
group by city -- selects cities having grade A
having max(marks)>=93 -- selects only those groped cities which also have marks >= 93
order by city; -- ascending by default
```

```
city
Delhi
Mumbai
```

In MySQL, the sql_safe_updates mode is a safeguard (by providing error) that prevents you from executing potentially dangerous UPDATE or DELETE statements that do not include a WHERE clause or a LIMIT clause.



How to Enable `sql_safe_updates`

To enable `sql_safe_updates`, you can use the following command:

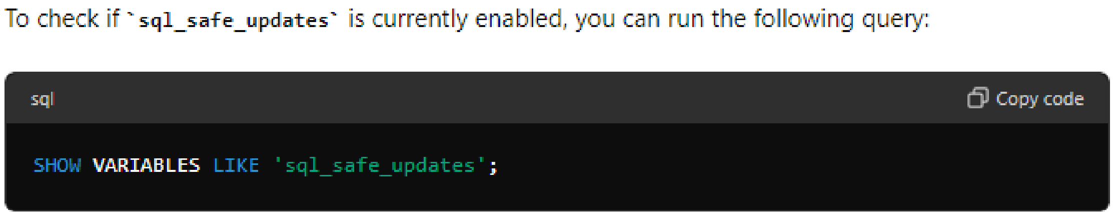
```sql
SET sql_safe_updates = 1;
```

How to Disable `sql_safe_updates`

To disable `sql_safe_updates`, you can use the following command:

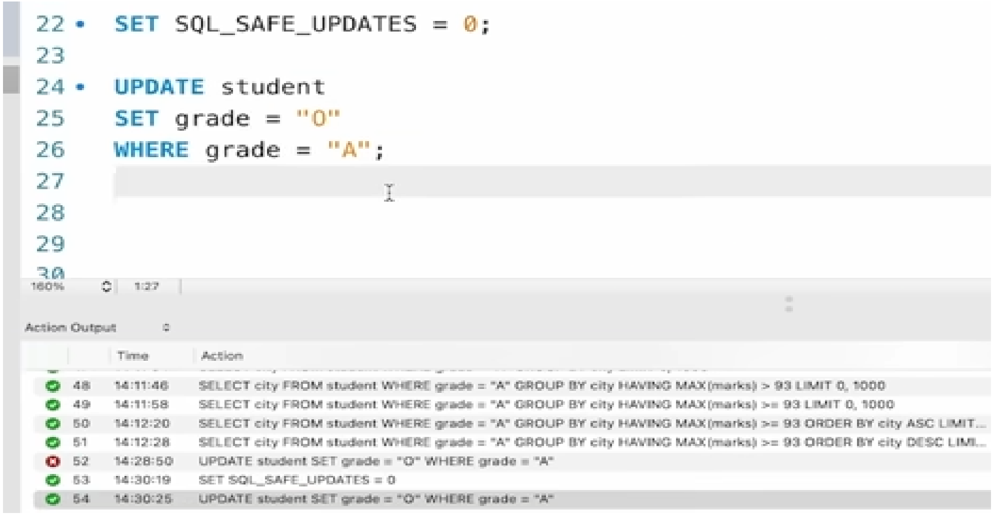
```sql
SET sql_safe_updates = 0;
```

By default it remains enabled when you download mysql. if on/enabled you can't execute potentially dangerous UPDATE or DELETE statements that do not include a WHERE clause or a LIMIT clause, it gives error.



To check if `sql_safe_updates` is currently enabled, you can run the following query:

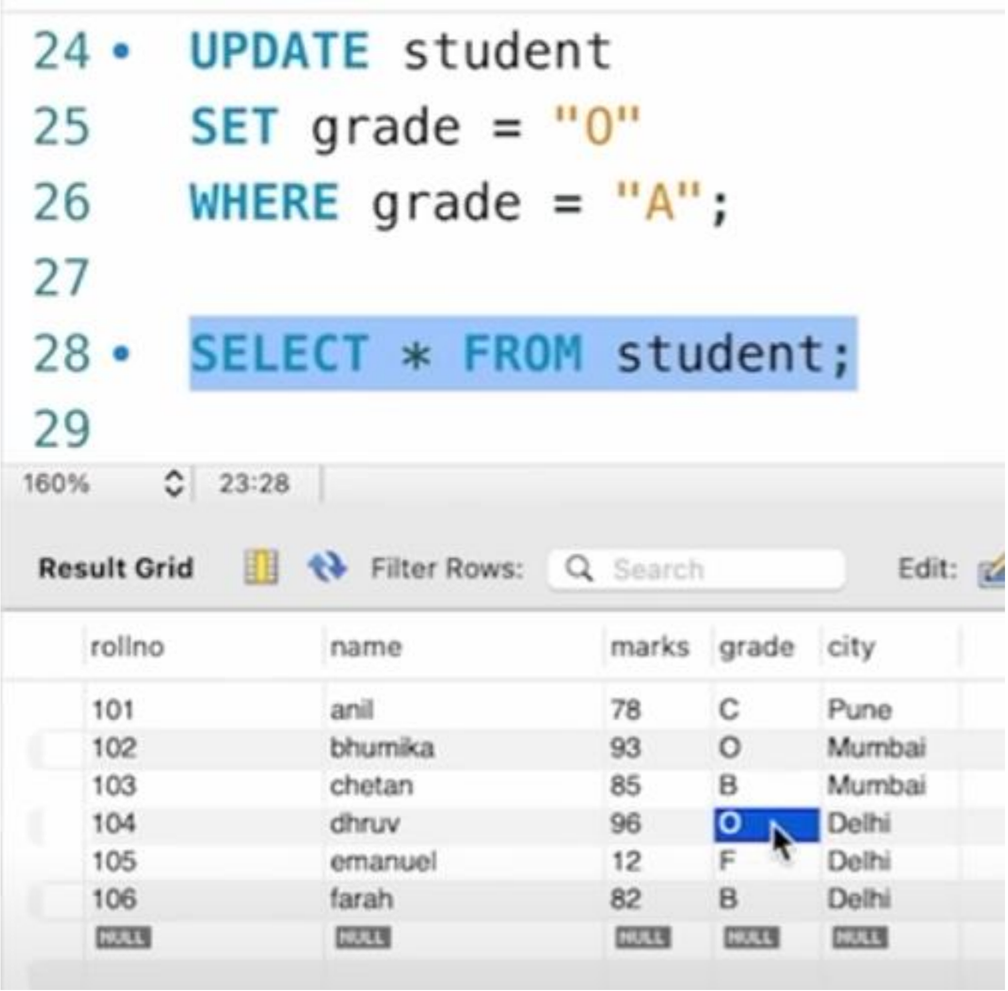
```sql
SHOW VARIABLES LIKE 'sql_safe_updates';
```



```sql
SET SQL_SAFE_UPDATES = 0;

UPDATE student
SET grade = "O"
WHERE grade = "A";
```

```
     Time      Action
48   14:11:46  SELECT city FROM student WHERE grade = "A" GROUP BY city HAVING MAX(marks) > 93 LIMIT 0, 1000
49   14:11:58  SELECT city FROM student WHERE grade = "A" GROUP BY city HAVING MAX(marks) >= 93 LIMIT 0, 1000
50   14:12:20  SELECT city FROM student WHERE grade = "A" GROUP BY city HAVING MAX(marks) >= 93 ORDER BY city ASC LIMIT...
51   14:12:28  SELECT city FROM student WHERE grade = "A" GROUP BY city HAVING MAX(marks) >= 93 ORDER BY city DESC LIMI...
52   14:28:50  UPDATE student SET grade = "O" WHERE grade = "A"
53   14:30:19  SET SQL_SAFE_UPDATES = 0
54   14:30:25  UPDATE student SET grade = "O" WHERE grade = "A"
```

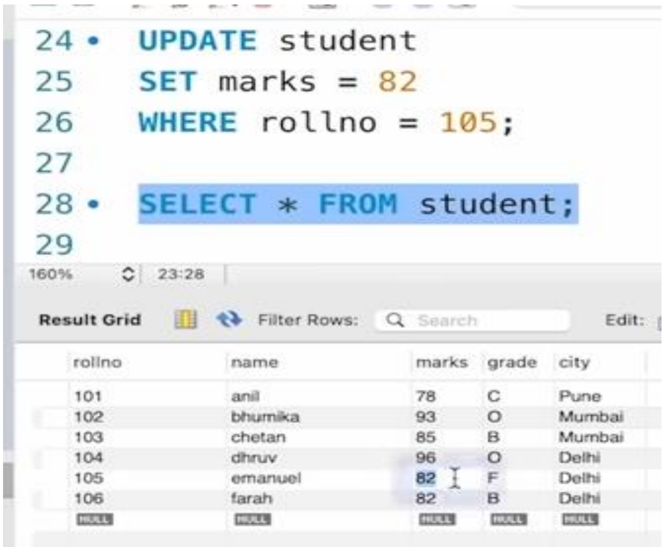
Earlier sql_safe_updates was on, so gave error. Therefore, it was disabled (0)



```
UPDATE student
SET grade = "O"
WHERE grade = "A";

SELECT * FROM student;

```

rollno	name	marks	grade	city
101	anil	78	C	Pune
102	bhumika	93	O	Mumbai
103	chetan	85	B	Mumbai
104	dhruv	96	O	Delhi
105	emanuel	12	F	Delhi
106	farah	82	B	Delhi
NULL	NULL	NULL	NULL	NULL

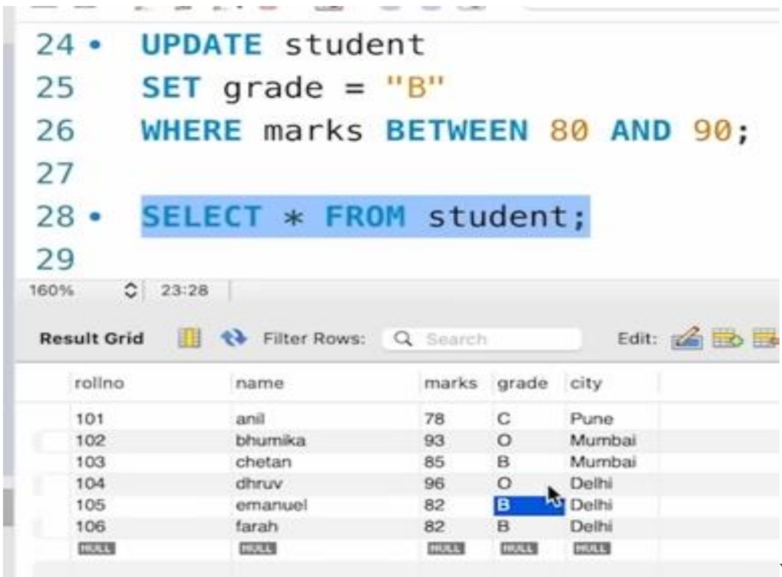
Wherever there was A grade, it was made O;



```
UPDATE student
SET marks = 82
WHERE rollno = 105;

SELECT * FROM student;

```

rollno	name	marks	grade	city
101	anil	78	C	Pune
102	bhumika	93	O	Mumbai
103	chetan	85	B	Mumbai
104	dhruv	96	O	Delhi
105	emanuel	82	F	Delhi
106	farah	82	B	Delhi
NULL	NULL	NULL	NULL	NULL

Emanuel marks updated to 82 from 12

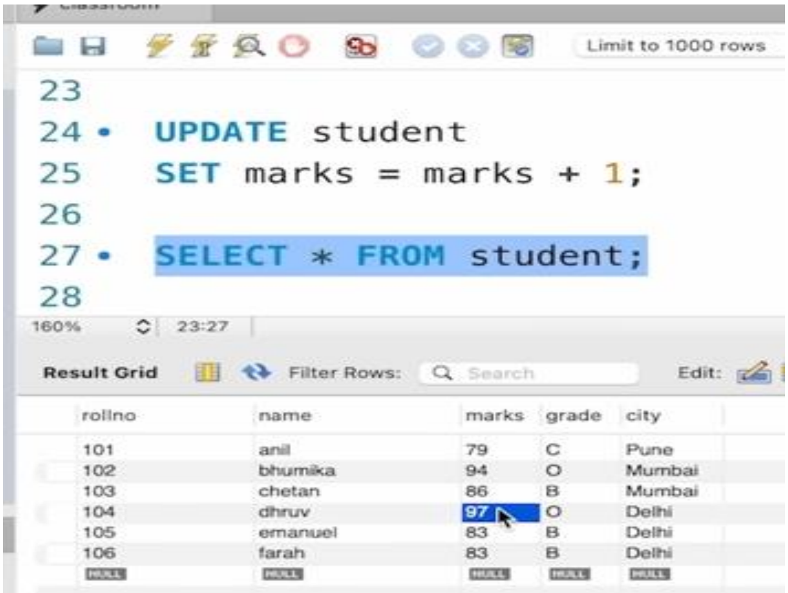


```sql
UPDATE student
SET grade = "B"
WHERE marks BETWEEN 80 AND 90;

SELECT * FROM student;
```

rollno	name	marks	grade	city
101	anil	78	C	Pune
102	bhumika	93	O	Mumbai
103	chetan	85	B	Mumbai
104	dhruv	96	O	Delhi
105	emanuel	82	B	Delhi
106	farah	82	B	Delhi
NULL	NULL	NULL	NULL	NULL

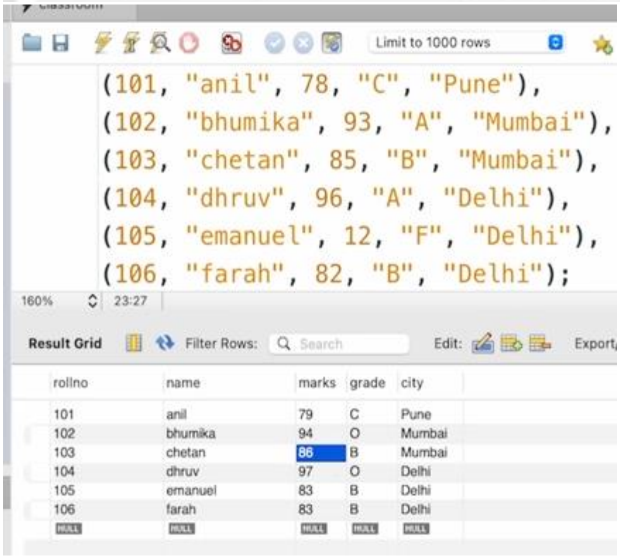
whoever obtained marks between 80 and 90 his marks, their grade made to B (so was with Emanuel)

Suppose marks of every student has to be increased by 1 as there was error in 1 MCQ



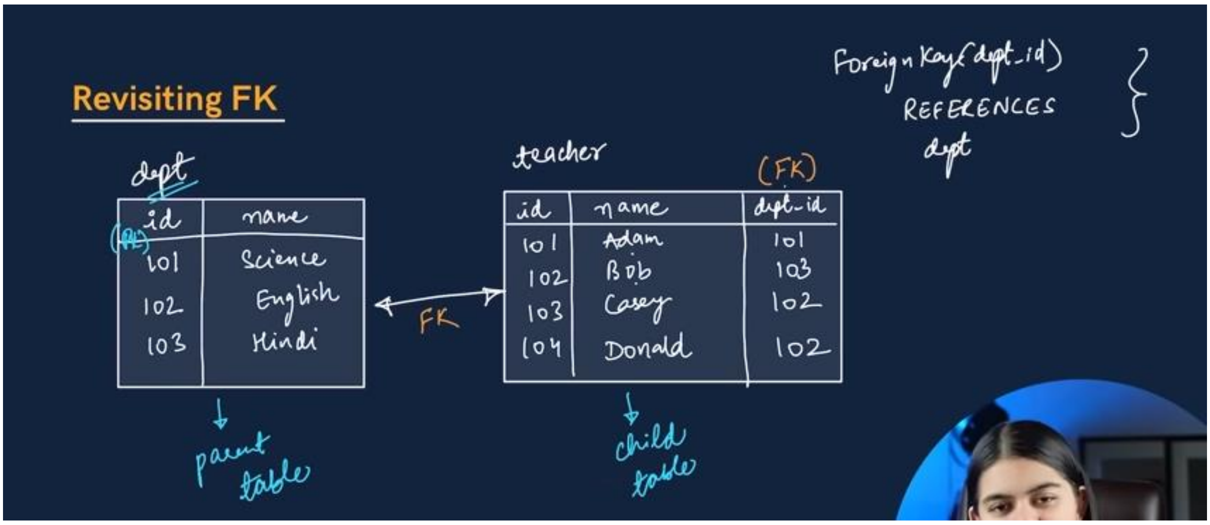
```sql
UPDATE student
SET marks = marks + 1;

SELECT * FROM student;
```

rollno	name	marks	grade	city
101	anil	79	C	Pune
102	bhumika	94	O	Mumbai
103	chetan	86	B	Mumbai
104	dhruv	97	O	Delhi
105	emanuel	83	B	Delhi
106	farah	83	B	Delhi
NULL	NULL	NULL	NULL	NULL



```sql
(101, "anil", 78, "C", "Pune"),
(102, "bhumika", 93, "A", "Mumbai"),
(103, "chetan", 85, "B", "Mumbai"),
(104, "dhruv", 96, "A", "Delhi"),
(105, "emanuel", 12, "F", "Delhi"),
(106, "farah", 82, "B", "Delhi");
```

rollno	name	marks	grade	city
101	anil	79	C	Pune
102	bhumika	94	O	Mumbai
103	chetan	86	B	Mumbai
104	dhruv	97	O	Delhi
105	emanuel	83	B	Delhi
106	farah	83	B	Delhi
NULL	NULL	NULL	NULL	NULL



Revisiting FK

dept

id (PK)	name
101	Science
102	English
103	Hindi

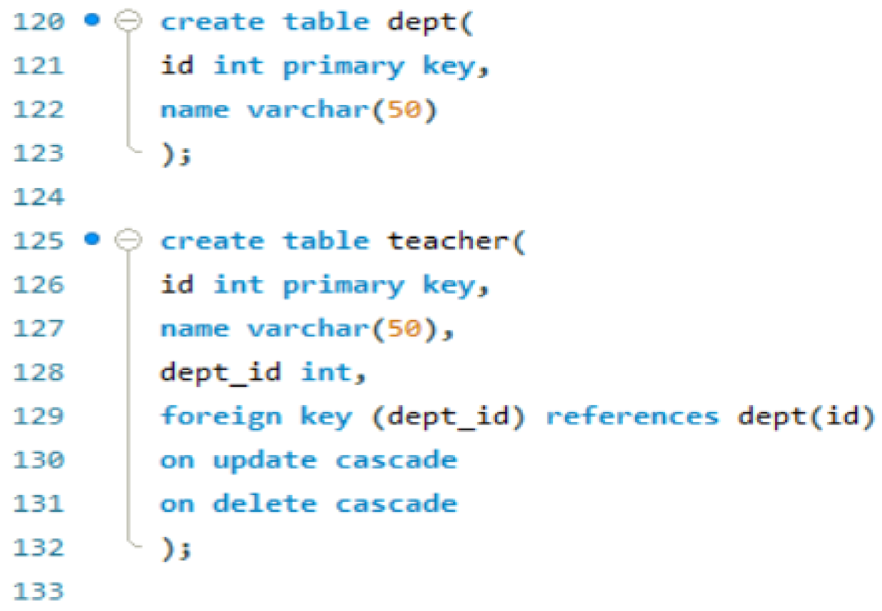
teacher

id	name	dept_id (FK)
101	Adam	101
102	Bob	103
103	Casey	102
104	Donald	102

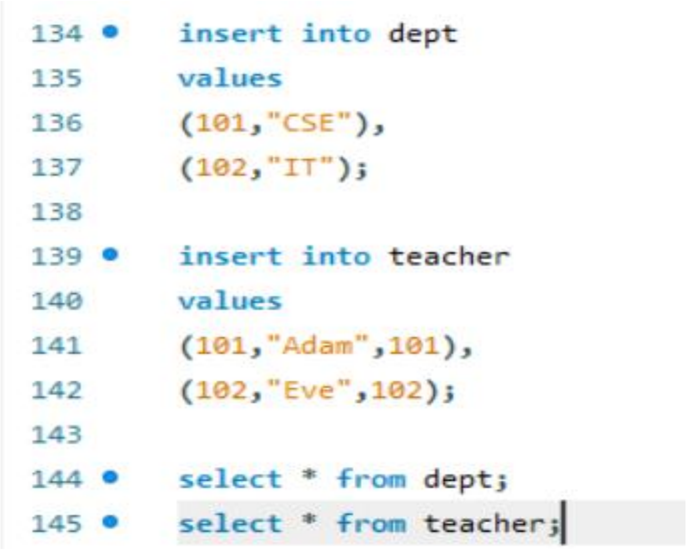
Foreign Key(dept_id) REFERENCES dept

FK: foreign key

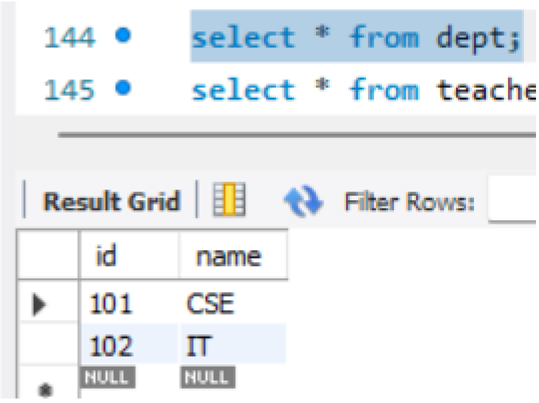
Cascading FK



```
create table dept(
id int primary key,
name varchar(50)
);

create table teacher(
id int primary key,
name varchar(50),
dept_id int,
foreign key (dept_id) references dept(id)
on update cascade
on delete cascade
);
```

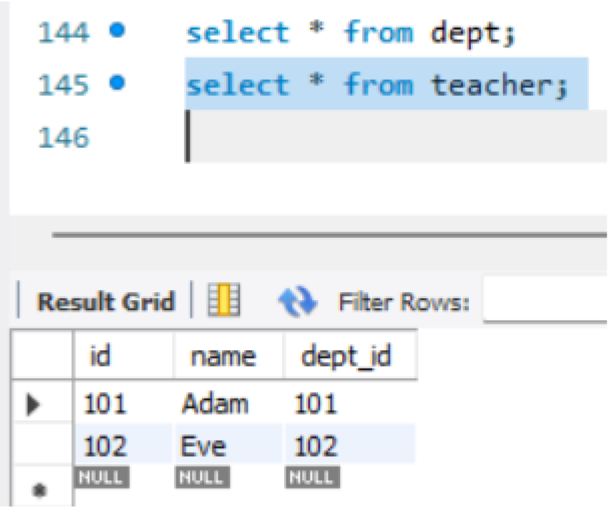


```
insert into dept
values
(101,"CSE"),
(102,"IT");

insert into teacher
values
(101,"Adam",101),
(102,"Eve",102);

select * from dept;
select * from teacher;
```



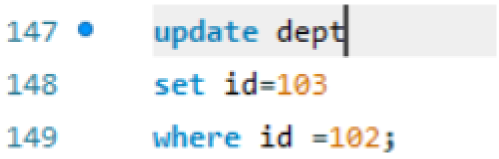
```
select * from dept;
select * from teache
```

id	name
101	CSE
102	IT
NULL	NULL

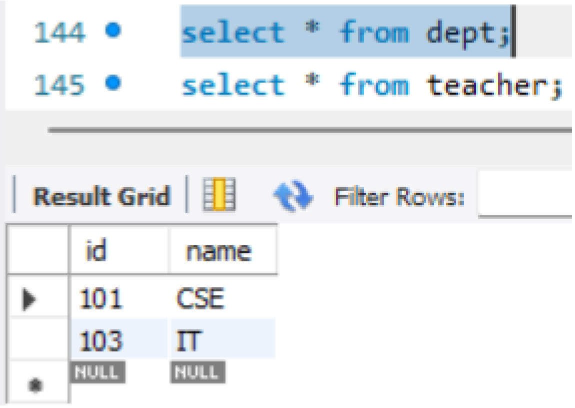


```
select * from dept;
select * from teacher;
```

id	name	dept_id
101	Adam	101
102	Eve	102
NULL	NULL	NULL

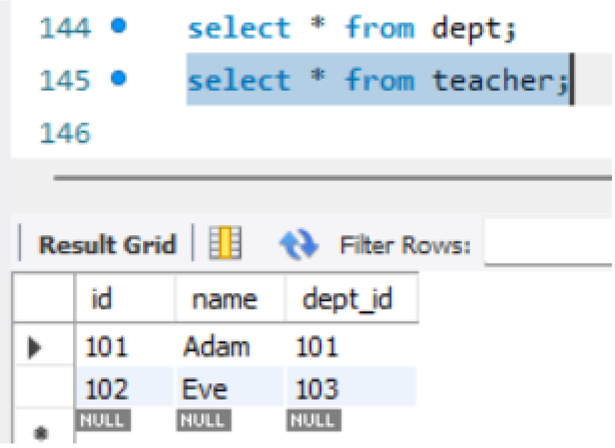


```
update dept
set id=103
where id =102;
```



```
select * from dept;
select * from teacher;
```

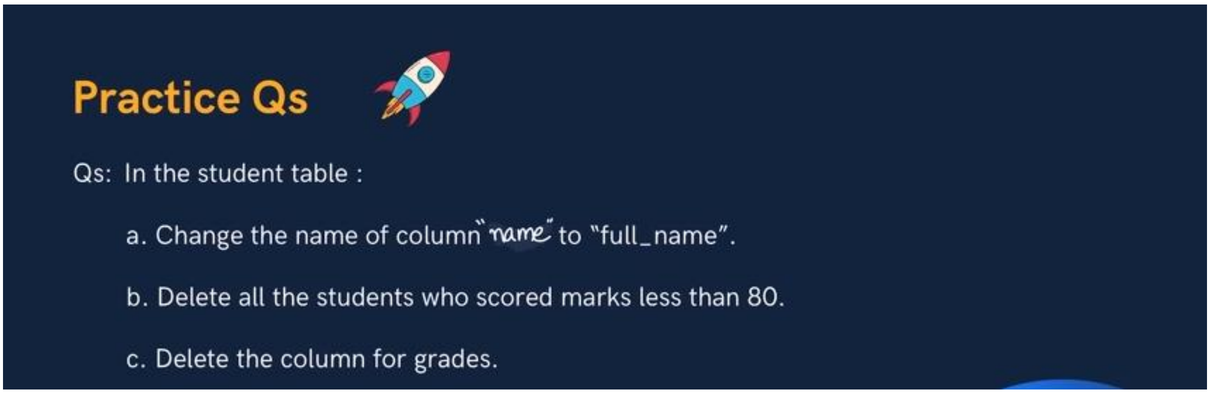
id	name
101	CSE
103	IT
NULL	NULL



```
select * from dept;
select * from teacher;
```

id	name	dept_id
101	Adam	101
102	Eve	103
NULL	NULL	NULL

Though, we updated dept id only in dept table it gets reflected in teacher table also, due to cascading(update cascade).



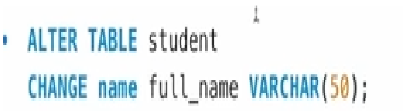
Practice Qs

Qs: In the student table :

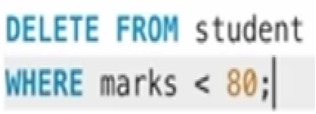
a. Change the name of column "name" to "full_name".

b. Delete all the students who scored marks less than 80.

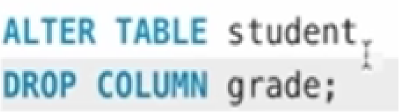
c. Delete the column for grades.



```
ALTER TABLE student
CHANGE name full_name VARCHAR(50);
```

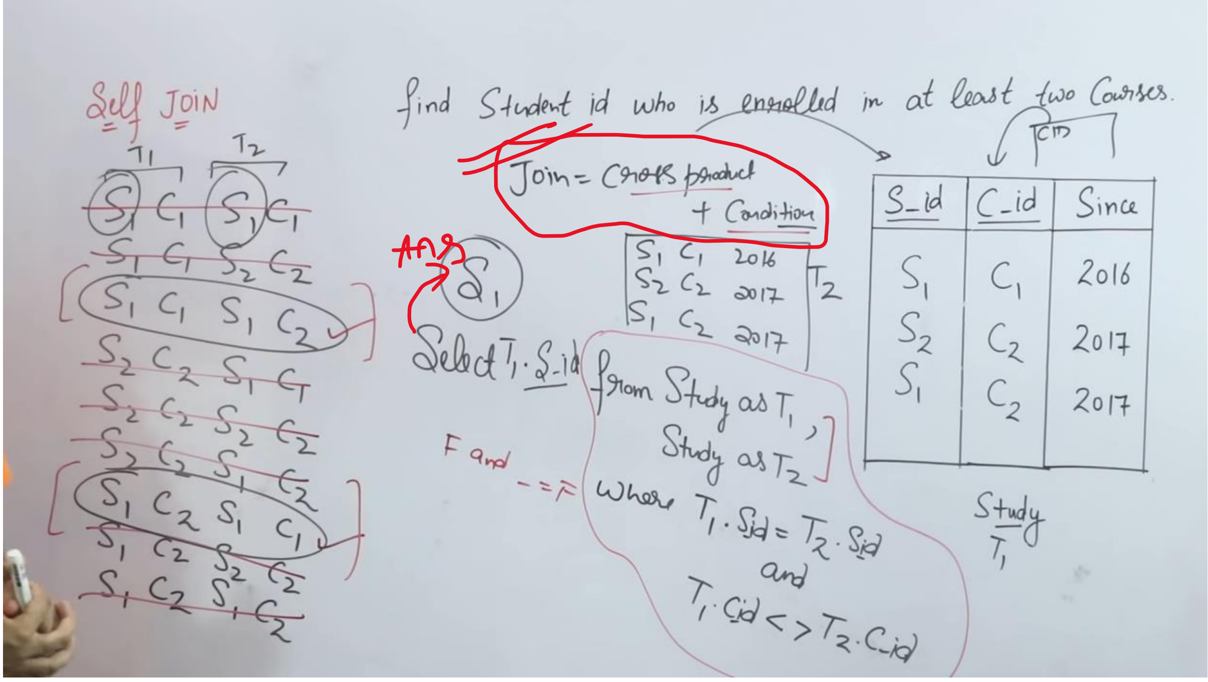


```
DELETE FROM student
WHERE marks < 80;
```



```
ALTER TABLE student
DROP COLUMN grade;
```

Self Join



Self JOIN

T1		T2	
~~S1~~	~~C1~~	~~S1~~	~~C1~~
~~S1~~	~~C1~~	~~S2~~	~~C2~~
S1	C1	S1	C2
~~S2~~	~~C2~~	~~S1~~	~~C1~~
~~S2~~	~~C2~~	~~S2~~	~~C2~~
~~S2~~	~~C2~~	~~S1~~	~~C2~~
S1	C2	S1	C1
~~S1~~	~~C2~~	~~S2~~	~~C2~~
~~S1~~	~~C2~~	~~S1~~	~~C2~~

find Student id who is enrolled in at least two Courses.

Join = Cross product + Condition

Ans S_1

S1	C1	2016
S2	C2	2017
S1	C2	2017

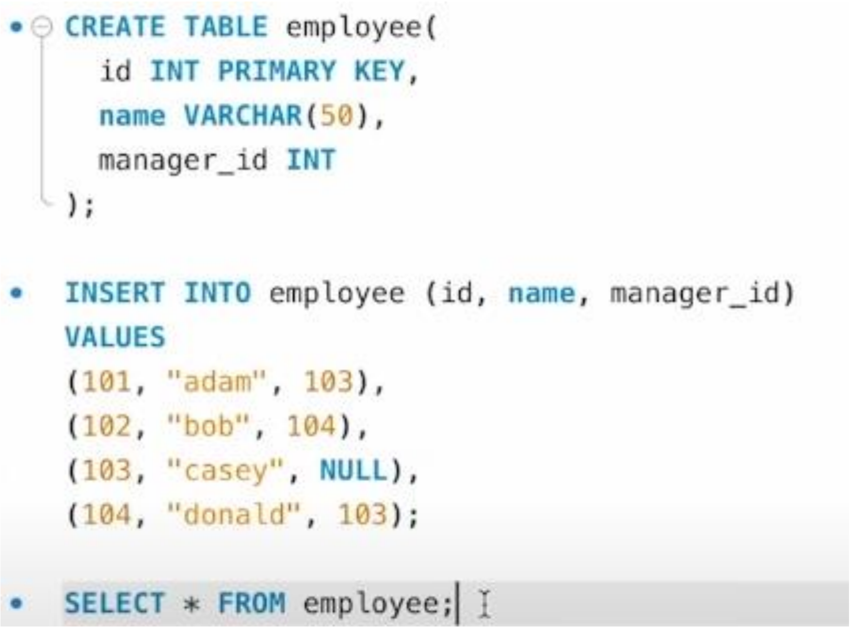
T2

```
Select T1.S_id
from Study as T1,
     Study as T2
Where T1.Sid = T2.Sid
and
T1.Cid <> T2.C_id
```

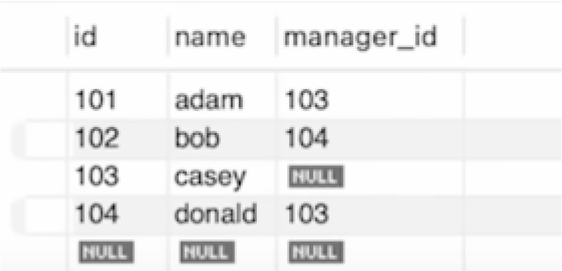
F and

S_id	C_id	Since
S1	C1	2016
S2	C2	2017
S1	C2	2017

Study
T1

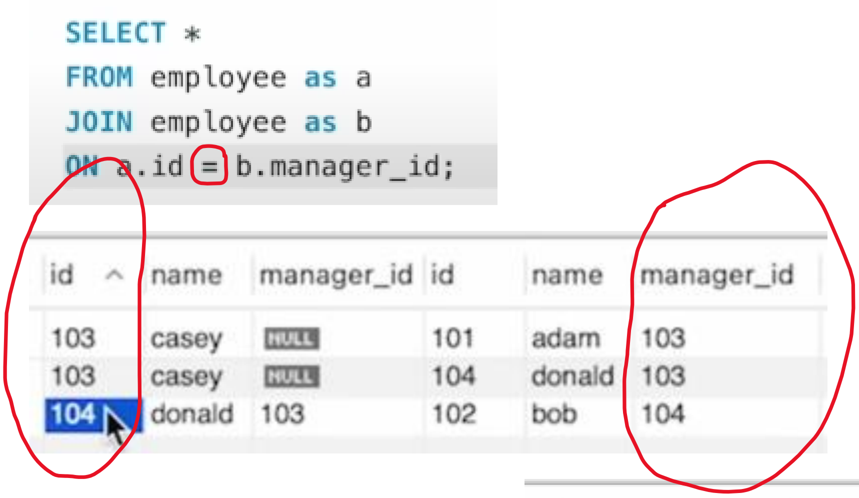


```
CREATE TABLE employee(
  id INT PRIMARY KEY,
  name VARCHAR(50),
  manager_id INT
);

INSERT INTO employee (id, name, manager_id)
VALUES
(101, "adam", 103),
(102, "bob", 104),
(103, "casey", NULL),
(104, "donald", 103);

SELECT * FROM employee;
```



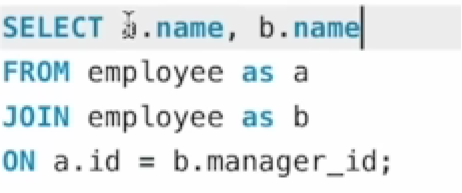
id	name	manager_id
101	adam	103
102	bob	104
103	casey	NULL
104	donald	103
NULL	NULL	NULL

casey is manager of adam, donald. donald is manager of bob. We need to show this insight.

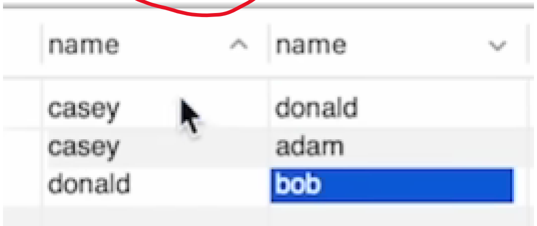


```
SELECT *
FROM employee as a
JOIN employee as b
ON a.id = b.manager_id;
```

id	name	manager_id	id	name	manager_id
103	casey	NULL	101	adam	103
103	casey	NULL	104	donald	103
104	donald	103	102	bob	104

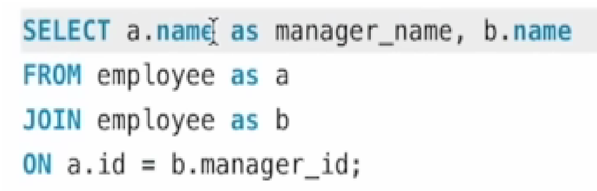


```
SELECT a.name, b.name
FROM employee as a
JOIN employee as b
ON a.id = b.manager_id;
```

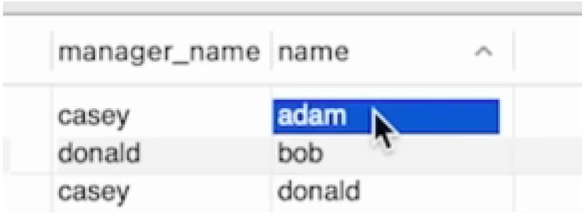


name	name
casey	donald
casey	adam
donald	bob

Finally,

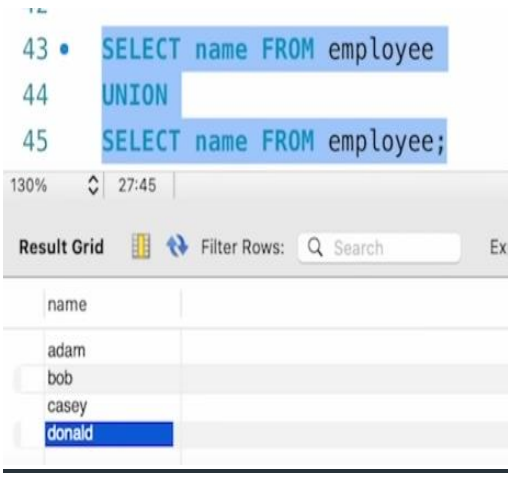


```
SELECT a.name as manager_name, b.name
FROM employee as a
JOIN employee as b
ON a.id = b.manager_id;
```



manager_name	name
casey	adam
donald	bob
casey	donald

Union

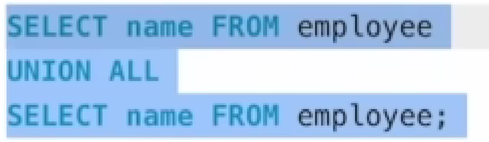


```
SELECT name FROM employee
UNION
SELECT name FROM employee;
```

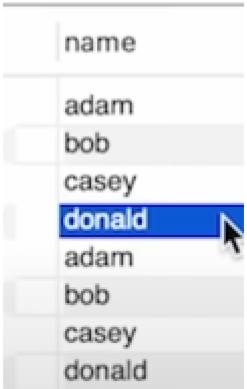
name
adam
bob
casey
donald

Union gives unique records (not duplicate)

Union All



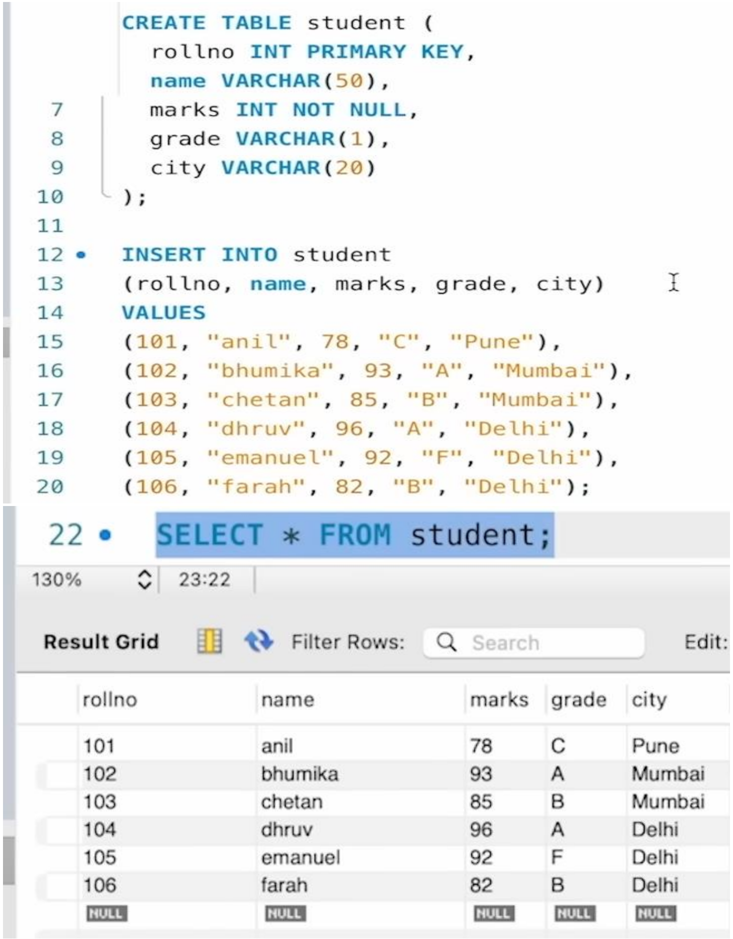
```
SELECT name FROM employee
UNION ALL
SELECT name FROM employee;
```



name
adam
bob
casey
donald
adam
bob
casey
donald

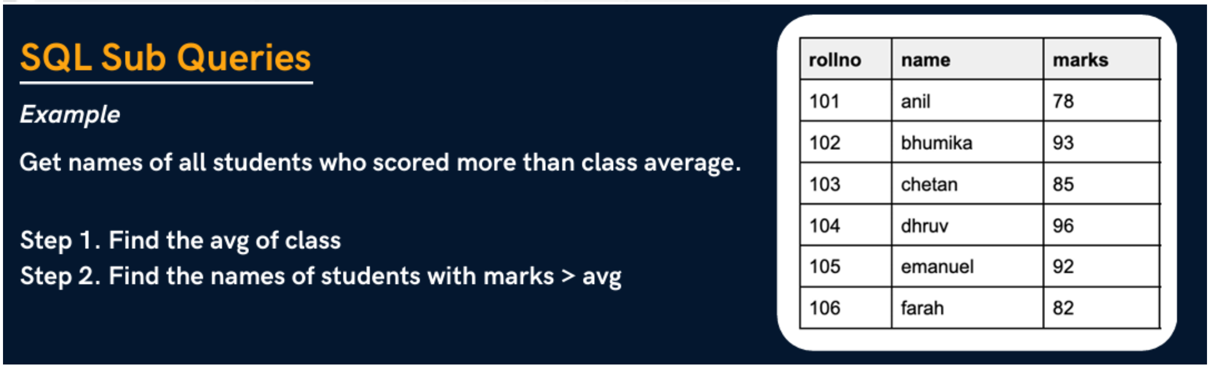
union all may give duplicate values as records may to be common to both the tables.

Sub Queries



```
CREATE TABLE student (
  rollno INT PRIMARY KEY,
  name VARCHAR(50),
  marks INT NOT NULL,
  grade VARCHAR(1),
  city VARCHAR(20)
);

INSERT INTO student
(rollno, name, marks, grade, city)
VALUES
(101, "anil", 78, "C", "Pune"),
(102, "bhumika", 93, "A", "Mumbai"),
(103, "chetan", 85, "B", "Mumbai"),
(104, "dhruv", 96, "A", "Delhi"),
(105, "emanuel", 92, "F", "Delhi"),
(106, "farah", 82, "B", "Delhi");

SELECT * FROM student;
```

rollno	name	marks	grade	city
101	anil	78	C	Pune
102	bhumika	93	A	Mumbai
103	chetan	85	B	Mumbai
104	dhruv	96	A	Delhi
105	emanuel	92	F	Delhi
106	farah	82	B	Delhi
NULL	NULL	NULL	NULL	NULL



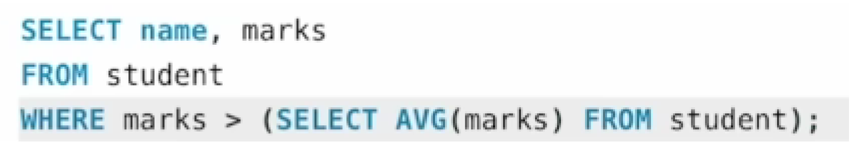
SQL Sub Queries

Example

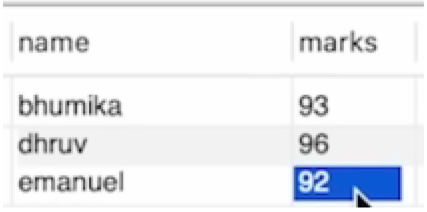
Get names of all students who scored more than class average.

Step 1. Find the avg of class
Step 2. Find the names of students with marks > avg

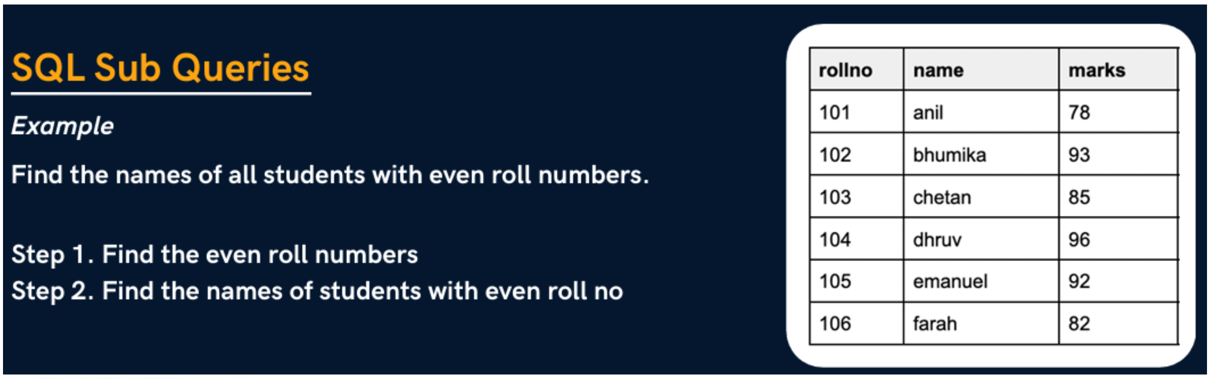
rollno	name	marks
101	anil	78
102	bhumika	93
103	chetan	85
104	dhruv	96
105	emanuel	92
106	farah	82



```
SELECT name, marks
FROM student
WHERE marks > (SELECT AVG(marks) FROM student);
```



name	marks
bhumika	93
dhruv	96
emanuel	92



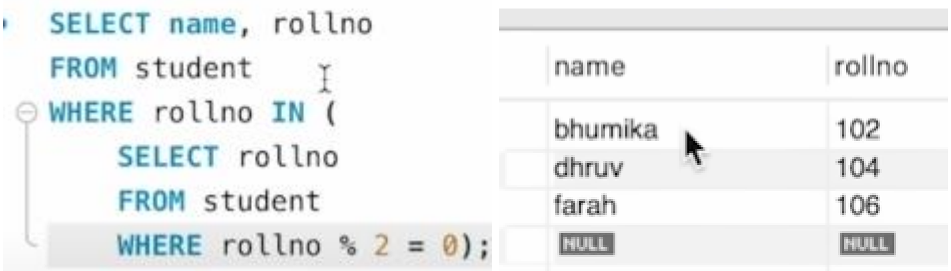
SQL Sub Queries

Example

Find the names of all students with even roll numbers.

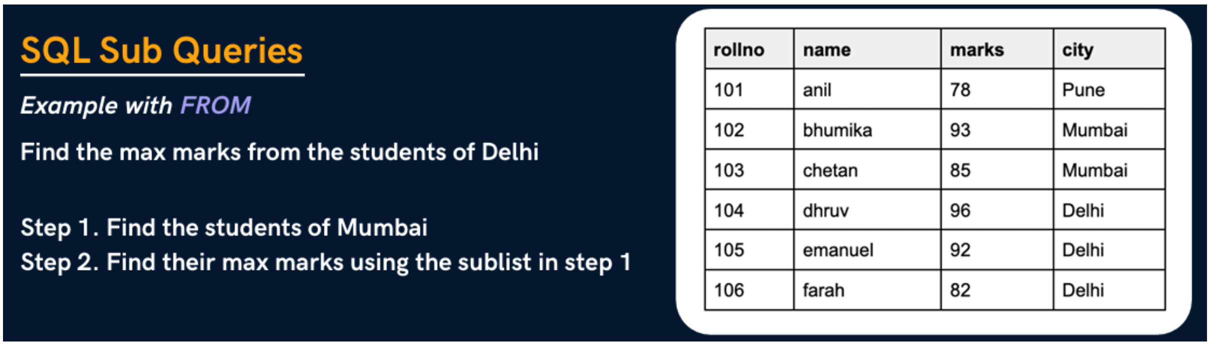
Step 1. Find the even roll numbers
Step 2. Find the names of students with even roll no

rollno	name	marks
101	anil	78
102	bhumika	93
103	chetan	85
104	dhruv	96
105	emanuel	92
106	farah	82



```
SELECT name, rollno
FROM student
WHERE rollno IN (
    SELECT rollno
    FROM student
    WHERE rollno % 2 = 0);
```

```
name        rollno
bhumika     102
dhruv       104
farah       106
NULL        NULL
```



SQL Sub Queries

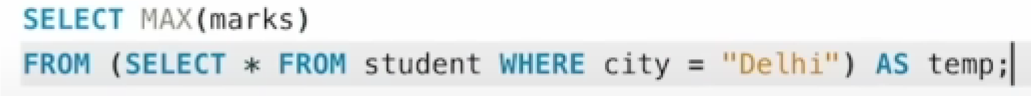
Example with FROM

Find the max marks from the students of Delhi

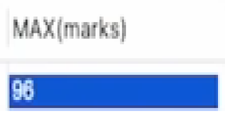
Step 1. Find the students of Mumbai
Step 2. Find their max marks using the sublist in step 1

rollno	name	marks	city
101	anil	78	Pune
102	bhumika	93	Mumbai
103	chetan	85	Mumbai
104	dhruv	96	Delhi
105	emanuel	92	Delhi
106	farah	82	Delhi

Writing subquery in from



```
SELECT MAX(marks)
FROM (SELECT * FROM student WHERE city = "Delhi") AS temp;
```



```
MAX(marks)
96
```

As here is alias

We could’ve written without using subquery(actually there are several methods to write same thing):

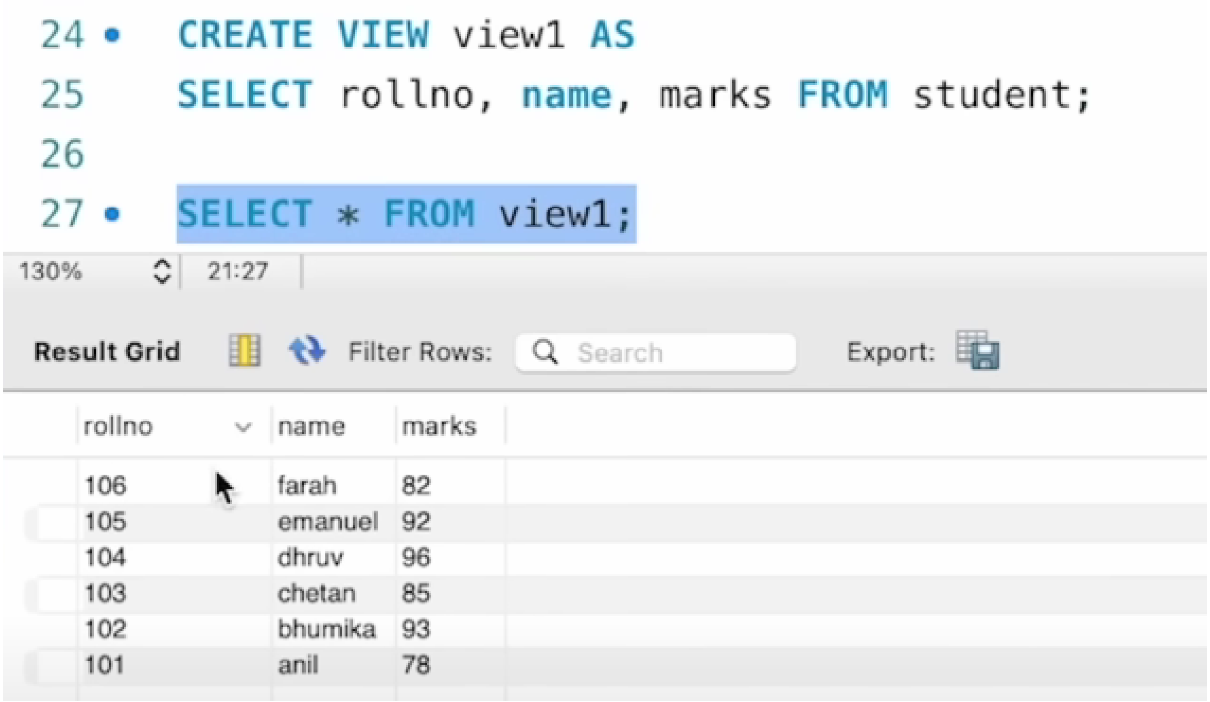
Select max(marks)

from student

where city=”Delhi”;

View

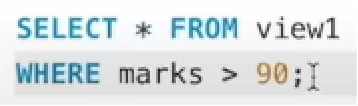
Suppose a teacher may require only require only rollno, name, marks info (not interested in city).Then we can provide the teacher a virtual table having the required info.once the table is provided teacher can perform actions on that table.



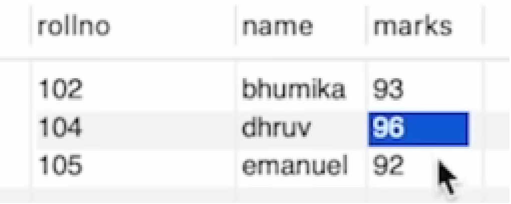
```
CREATE VIEW view1 AS
SELECT rollno, name, marks FROM student;

SELECT * FROM view1;
```

rollno	name	marks
106	farah	82
105	emanuel	92
104	dhruv	96
103	chetan	85
102	bhumika	93
101	anil	78

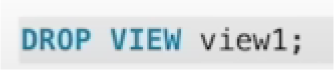
Now we can run multiple queries on this view table.



```
SELECT * FROM view1
WHERE marks > 90;
```



rollno	name	marks
102	bhumika	93
104	dhruv	96
105	emanuel	92



```
DROP VIEW view1;
```

Once view is deleted the virtual table is deleted.

Views do not change data in the actual table/base table unless you perform DML operations (INSERT, UPDATE, DELETE) on an updatable view, and those operations will reflect on the underlying base tables.