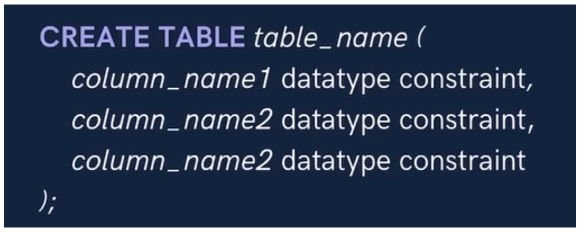
SQL Notes

1. CREATE database temp1; // create new database

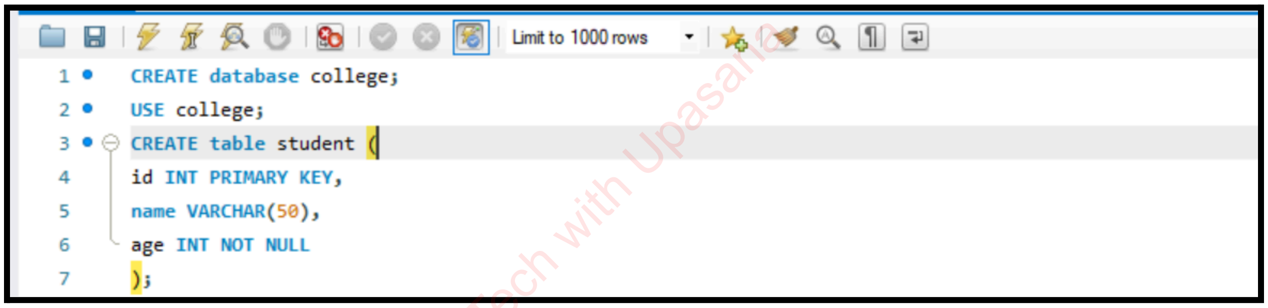
2. DROP database temp1;   // delete new database

3. USE college;   // is used when there are multiple databases in the SQL and the user or programmer specifically wants to use a particular database.

4. SELECT * from student;

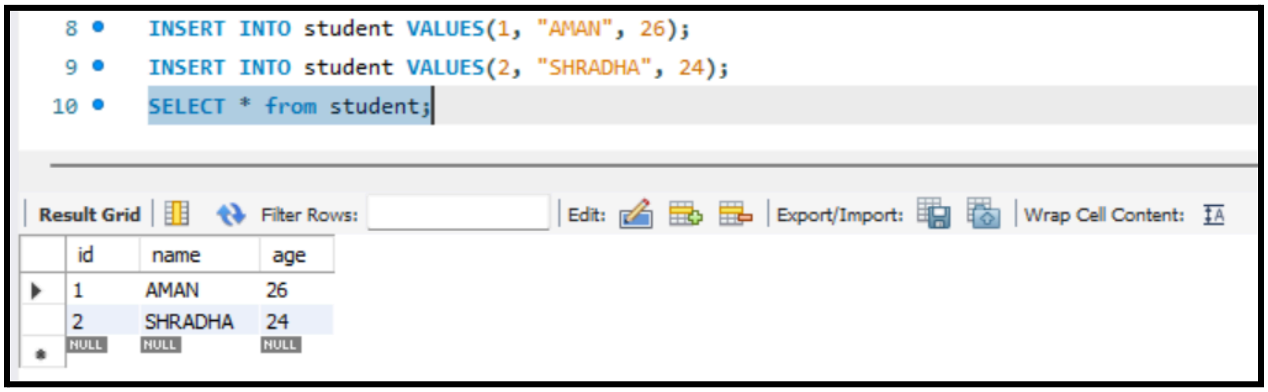


```
CREATE TABLE table_name (
    column_name1 datatype constraint,
    column_name2 datatype constraint,
    column_name2 datatype constraint
);
```



```
CREATE database college;
USE college;
CREATE table student (
id INT PRIMARY KEY,
name VARCHAR(50),
age INT NOT NULL
);
```

Primary key of every row is different. It's always not null.



```
INSERT INTO student VALUES(1, "AMAN", 26);
INSERT INTO student VALUES(2, "SHRADHA", 24);
SELECT * from student;
```

id	name	age
1	AMAN	26
2	SHRADHA	24
NULL	NULL	NULL

DATATYPE	DESCRIPTION	USAGE
CHAR	string(0-255), can store characters of fixed length	CHAR(50)
VARCHAR	string(0-255), can store characters up to given length	VARCHAR(50)
BLOB	string(0-65535), can store binary large object	BLOB(1000)
INT	integer(-2,147,483,648 to 2,147,483,647)	INT
TINYINT	integer(-128 to 127)	TINYINT
BIGINT	integer(-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)	BIGINT
BIT	can store x-bit values. x can range from 1 to 64	BIT(2)
FLOAT	Decimal number - with precision to 23 digits	FLOAT
DOUBLE	Decimal number - with 24 to 53 digits	DOUBLE
BOOLEAN	Boolean values 0 or 1	BOOLEAN
DATE	date in format of YYYY-MM-DD ranging from 1000-01-01 to 9999-12-31	DATE
TIME	HH:MM:SS	TIME
YEAR	year in 4 digits format ranging from 1901 to 2155	YEAR

CHAR - stores fixed length
VARCHAR - stores upto given length

Col1 CHAR(50) - isme agar hme PUNE store krana h to vo 4 nhi balki pure 50 bytes in the memory reserve kr lega chaye use ho ya na ho. Inefficient use of memory space. Extra memory is wasted.

Col2 VARCHAR(50) - isme agar PUNE store krana h to ye bs 4 bytes hi use krega in contrast of CHAR jo pure 50 bytes reserve krra h .

BLOB - used to store large binary objects.

INT - negative, positive both values

BIT(2) - 2 bit values store hongi like 10, 00, 11

BIT(1) - 1 bit values store hongi like 1,0

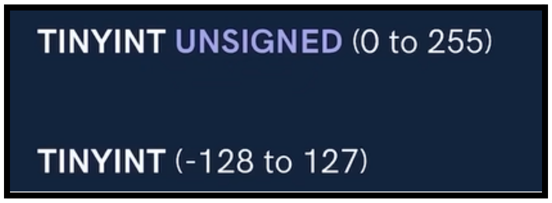
FLOAT - small decimal numbers

DOUBLE - large decimal numbers

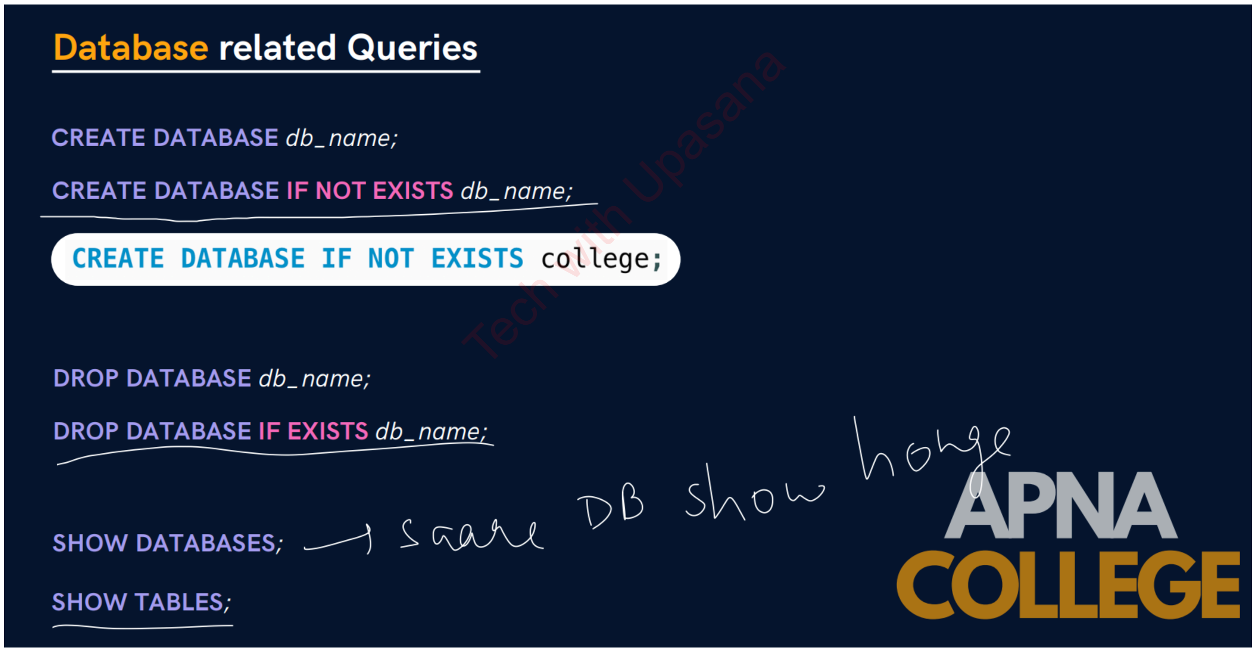
BOOLEAN Implementation in MYSQL - TINYINT

SIGNED - by default jo numeric data types hote h like int, float, double inke andar negative, positive aa skte h

Agar dono aate - signed
Agar sirf positive aayegi values - unsigned (salary, age)


TINYINT UNSIGNED (0 to 255)

TINYINT (-128 to 127)


Database related Queries

CREATE DATABASE *db_name;*

CREATE DATABASE IF NOT EXISTS *db_name;*

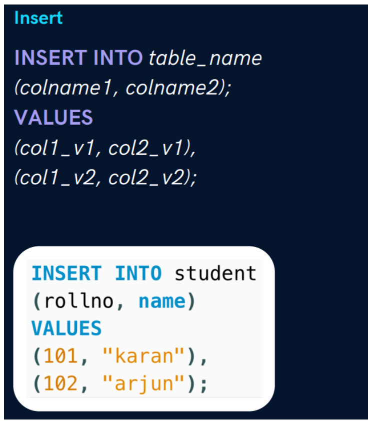
```
CREATE DATABASE IF NOT EXISTS college;
```

DROP DATABASE *db_name;*

DROP DATABASE IF EXISTS *db_name;*

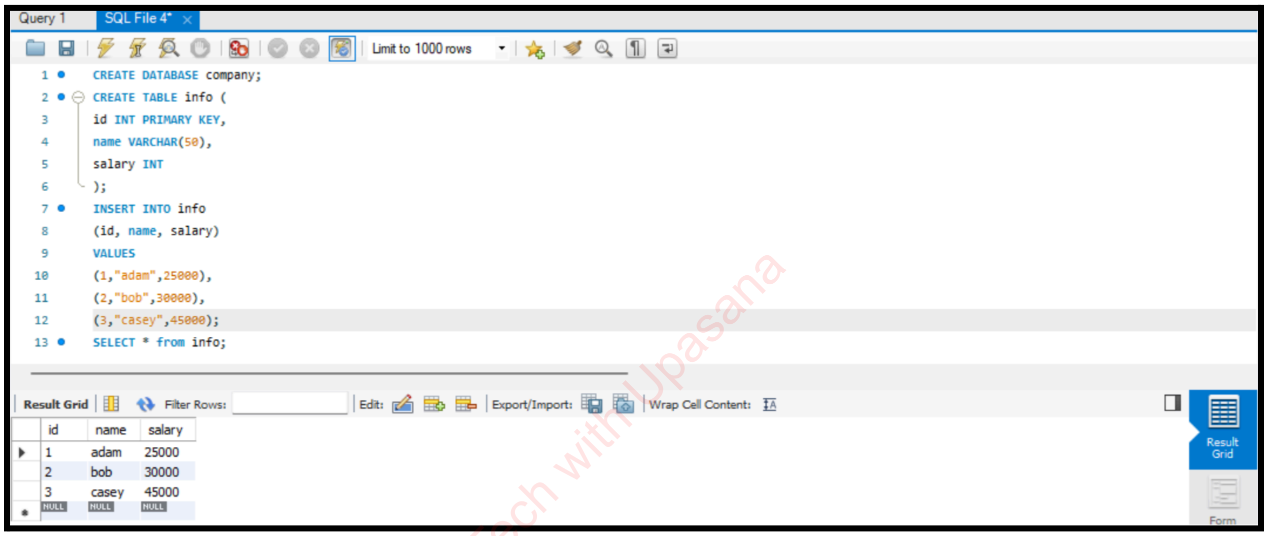
SHOW DATABASES; $\rightarrow$ saare DB show honge

SHOW TABLES;


Insert

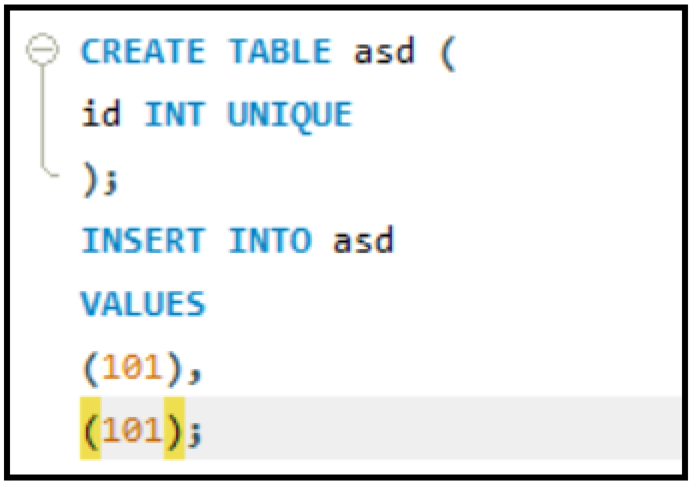
```
INSERT INTO table_name
(colname1, colname2);
VALUES
(col1_v1, col2_v1),
(col1_v2, col2_v2);
```

```
INSERT INTO student
(rollno, name)
VALUES
(101, "karan"),
(102, "arjun");
```



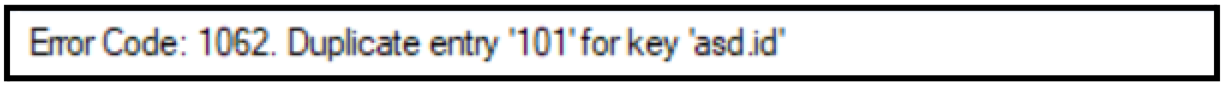
```
CREATE DATABASE company;
CREATE TABLE info (
id INT PRIMARY KEY,
name VARCHAR(50),
salary INT
);
INSERT INTO info
(id, name, salary)
VALUES
(1,"adam",25000),
(2,"bob",30000),
(3,"casey",45000);
SELECT * from info;
```

```
id    name    salary
1     adam    25000
2     bob     30000
3     casey   45000
NULL  NULL    NULL
```



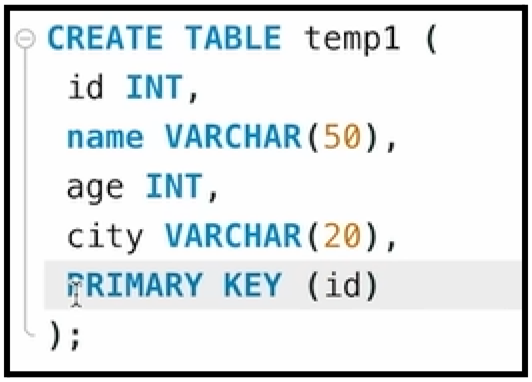
```
CREATE TABLE asd (
id INT UNIQUE
);
INSERT INTO asd
VALUES
(101),
(101);
```

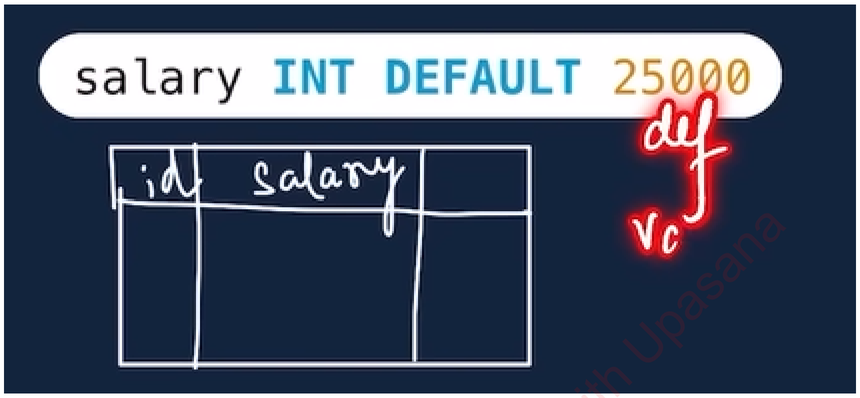
UNIQUE Constraint

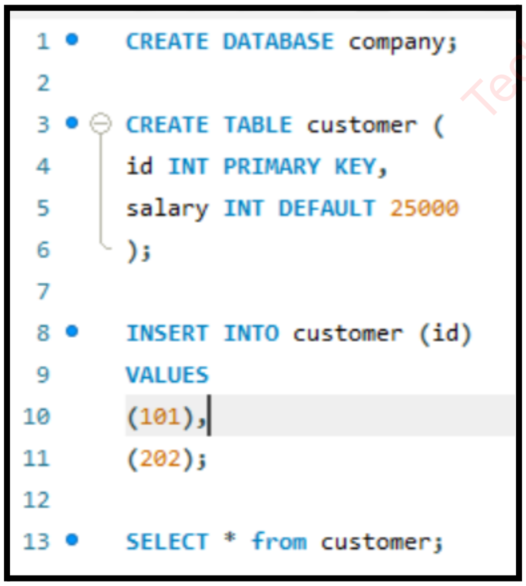


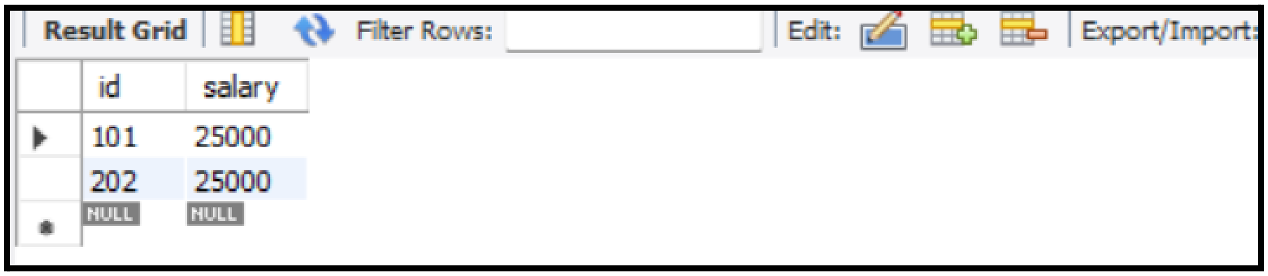
```
Error Code: 1062. Duplicate entry '101' for key 'asd.id'
```

2ND SYNTAX OF PRIMARY KEY


```
CREATE TABLE temp1 (
 id INT,
 name VARCHAR(50),
 age INT,
 city VARCHAR(20),
 PRIMARY KEY (id)
);
```
Also, we can make combination of 2 columns as primary key, their combination will be unique.

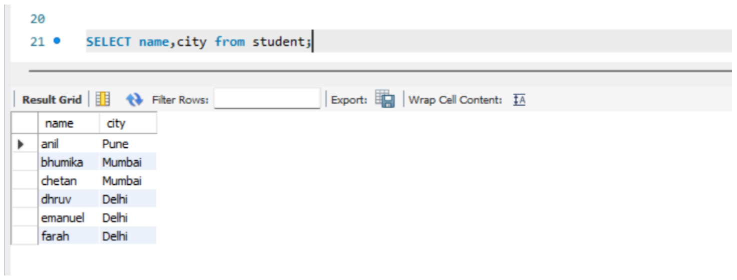



```
CREATE DATABASE company;

CREATE TABLE customer (
id INT PRIMARY KEY,
salary INT DEFAULT 25000
);

INSERT INTO customer (id)
VALUES
(101),
(202);

SELECT * from customer;
```
Yha hmne salary ko default bna diya, to hmne bs id ki values insert kri and salary by default 25000 ho jayegi.



id	salary
101	25000
202	25000
NULL	NULL

SELECT col1, col2 from student;

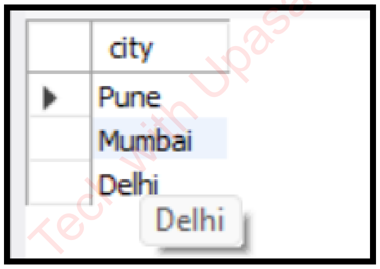


```
SELECT name,city from student;
```

name	city
anil	Pune
bhumika	Mumbai
chetan	Mumbai
dhruv	Delhi
emanuel	Delhi
farah	Delhi

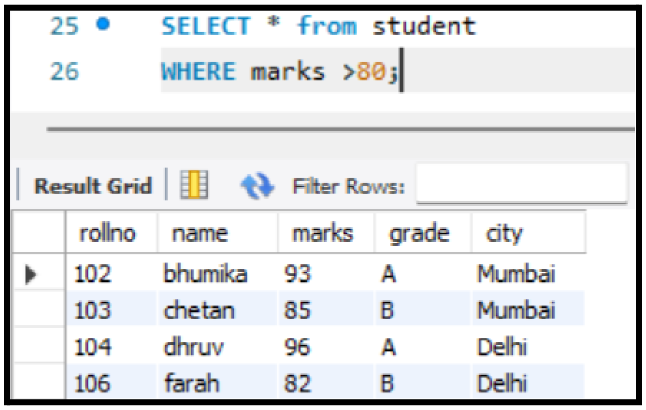
DISTINCT - only unique values will be visible, not the duplicate ones

SELECT DISTINCT city from student;



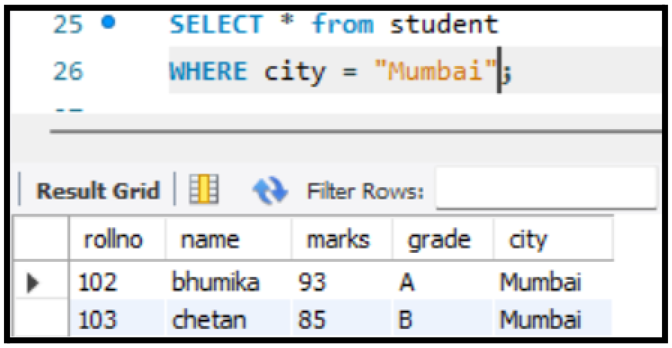
city
Pune
Mumbai
Delhi

WHERE CLAUSE



```
SELECT * from student
WHERE marks >80;
```

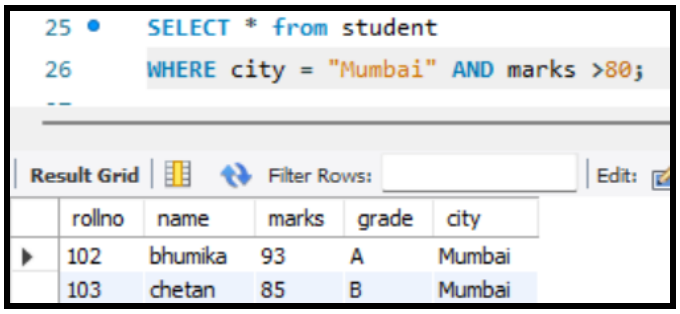
rollno	name	marks	grade	city
102	bhumika	93	A	Mumbai
103	chetan	85	B	Mumbai
104	dhruv	96	A	Delhi
106	farah	82	B	Delhi



```
SELECT * from student
WHERE city = "Mumbai";
```

Result Grid | Filter Rows:

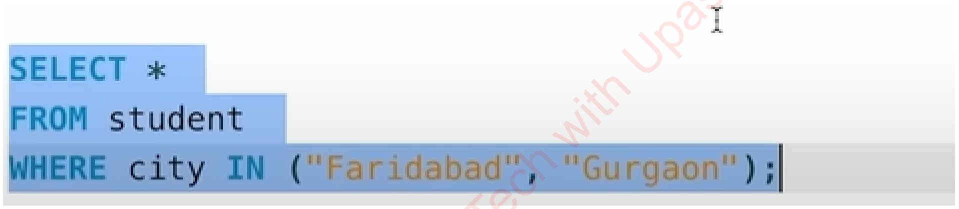
rollno	name	marks	grade	city
102	bhumika	93	A	Mumbai
103	chetan	85	B	Mumbai



```
SELECT * from student
WHERE city = "Mumbai" AND marks >80;
```

Result Grid | Filter Rows: | Edit:

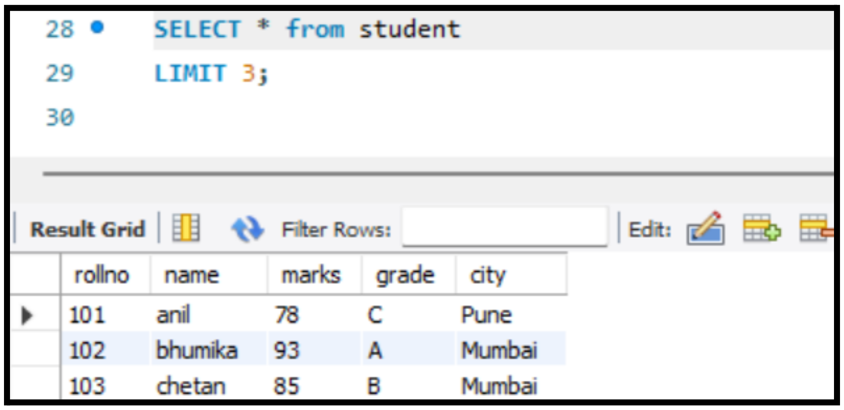
rollno	name	marks	grade	city
102	bhumika	93	A	Mumbai
103	chetan	85	B	Mumbai



```
SELECT *
FROM student
WHERE city IN ("Faridabad", "Gurgaon");
```

Isme hmari empty table print hogi qki asa koi data nhi h db mein jisme faridabad, gurgaon hai.

LIMIT CLAUSE

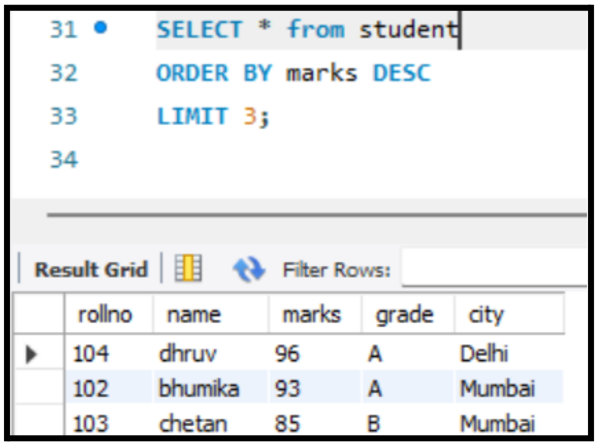


```
SELECT * from student
LIMIT 3;
```

Result Grid | Filter Rows: | Edit:

rollno	name	marks	grade	city
101	anil	78	C	Pune
102	bhumika	93	A	Mumbai
103	chetan	85	B	Mumbai

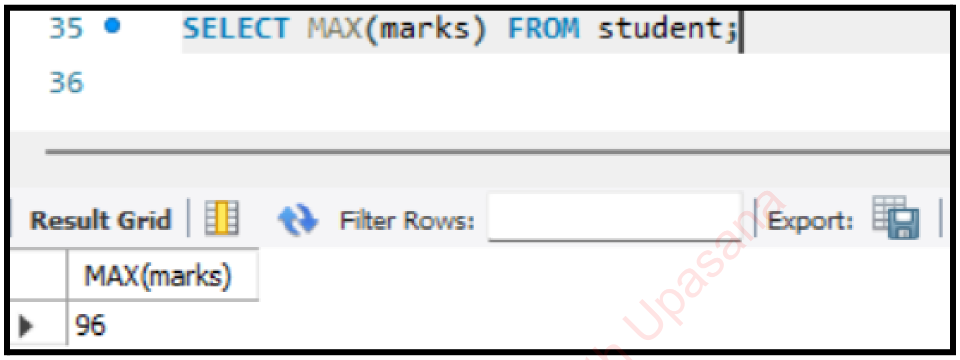
ORDER BY Clause



```
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

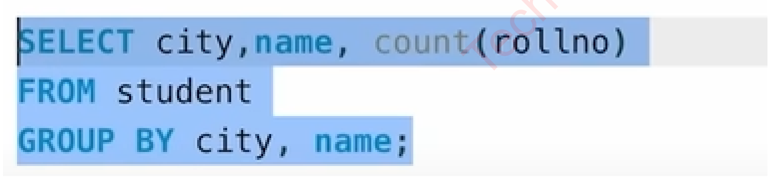
AGGREGATE FUNCTION returns a single value as an output.



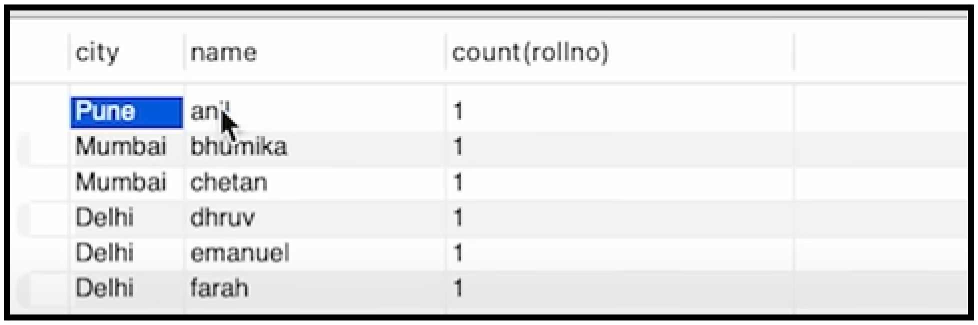
```
SELECT MAX(marks) FROM student;
```

MAX(marks)
96

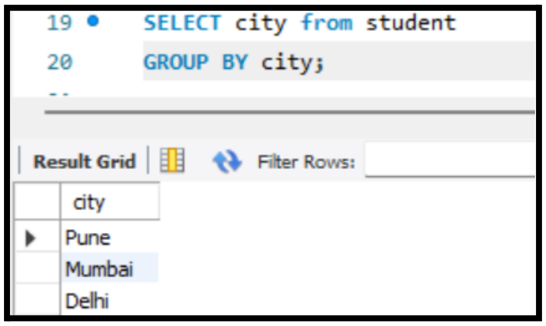
In GROUP BY, columns should be same in SELECT, GROUP BY.



```
SELECT city,name, count(rollno)
FROM student
GROUP BY city, name;
```

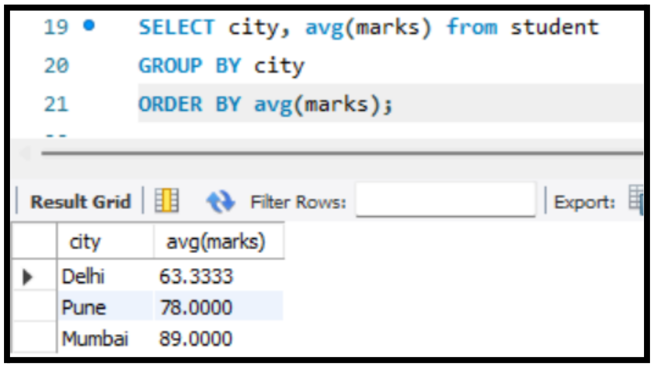


city	name	count(rollno)
Pune	anil	1
Mumbai	bhumika	1
Mumbai	chetan	1
Delhi	dhruv	1
Delhi	emanuel	1
Delhi	farah	1



```
SELECT city from student
GROUP BY city;
```

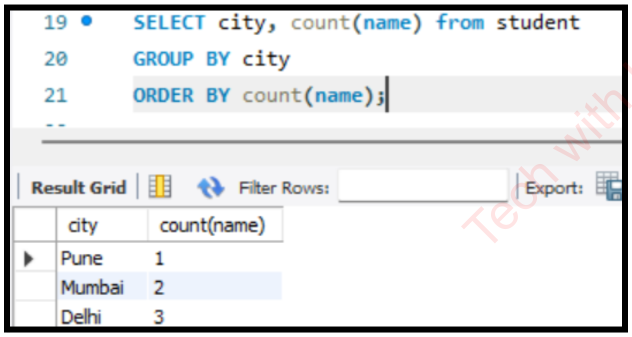
city
Pune
Mumbai
Delhi



```
SELECT city, avg(marks) from student
GROUP BY city
ORDER BY avg(marks);
```

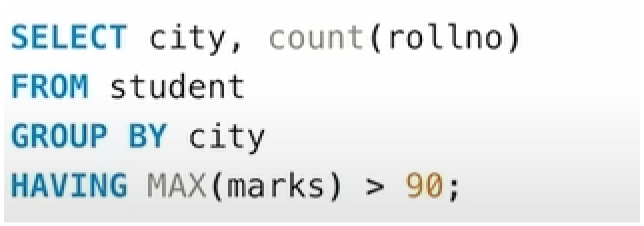
city	avg(marks)
Delhi	63.3333
Pune	78.0000
Mumbai	89.0000

By default, ascending mein hi aayega



```
SELECT city, count(name) from student
GROUP BY city
ORDER BY count(name);
```

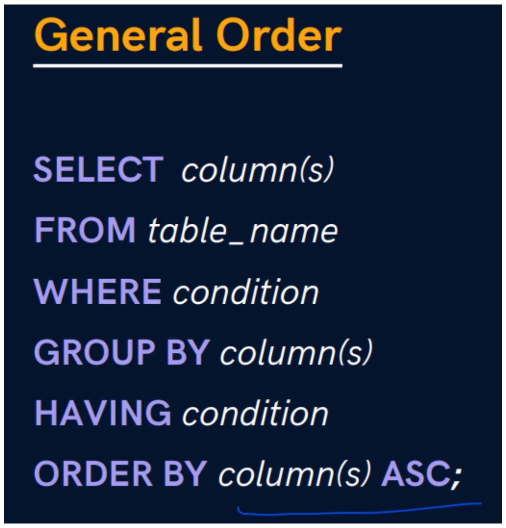
city	count(name)
Pune	1
Mumbai	2
Delhi	3



```
SELECT city, count(rollno)
FROM student
GROUP BY city
HAVING MAX(marks) > 90;
```

WHERE clause vhi kaam aayega jha condition satisfy hogi, APPLIED BEFORE GROUPING
It can't contain aggregate function

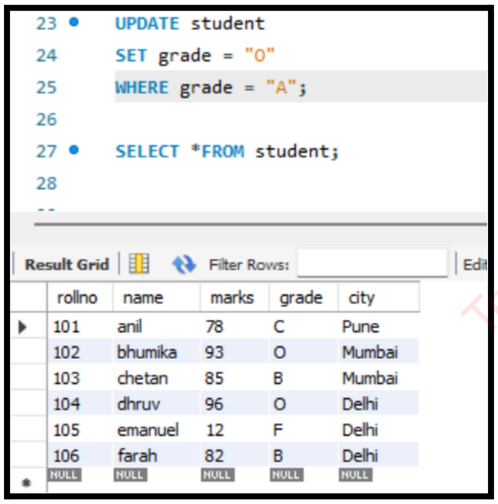
HAVING CLAUSE IS APPLIED AFTER GROUPING
It can contain aggregate function



General Order

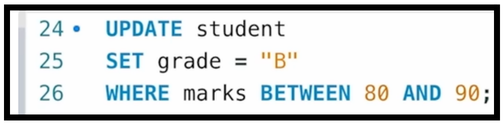
```
SELECT column(s)
FROM table_name
WHERE condition
GROUP BY column(s)
HAVING condition
ORDER BY column(s) ASC;
```

SAFE MODE - prevents us from doing the change in the database
SET SQL_SAFE_UPDATES = 0; (0 matlab off)

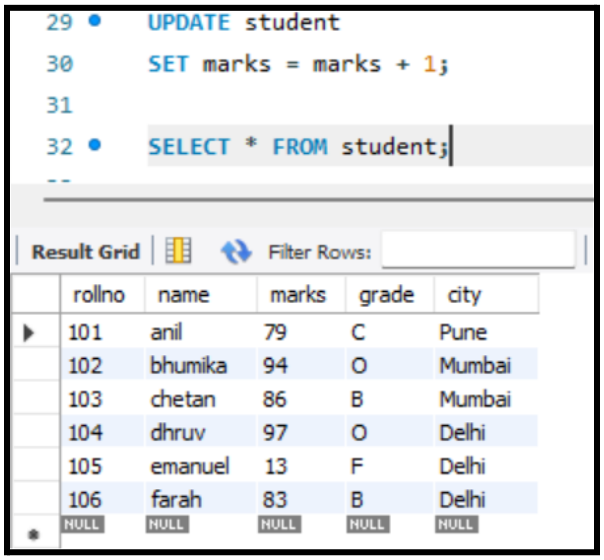


```sql
UPDATE student
SET grade = "O"
WHERE grade = "A";

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



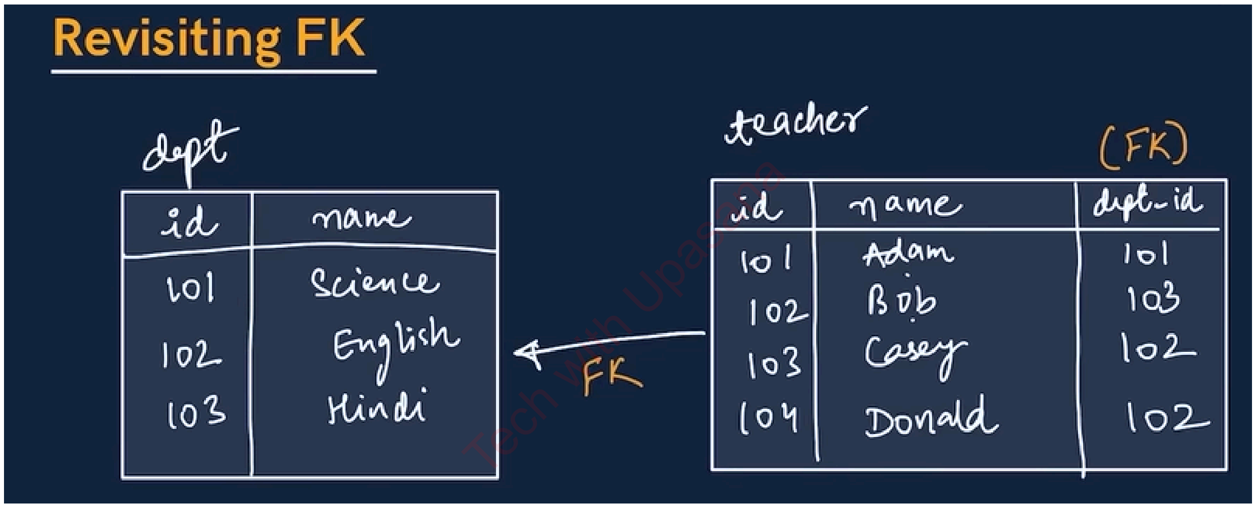
```sql
UPDATE student
SET grade = "B"
WHERE marks BETWEEN 80 AND 90;
```



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

SELECT * FROM student;
```

Result Grid | Filter Rows:

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	13	F	Delhi
106	farah	83	B	Delhi
NULL	NULL	NULL	NULL	NULL



Revisiting FK

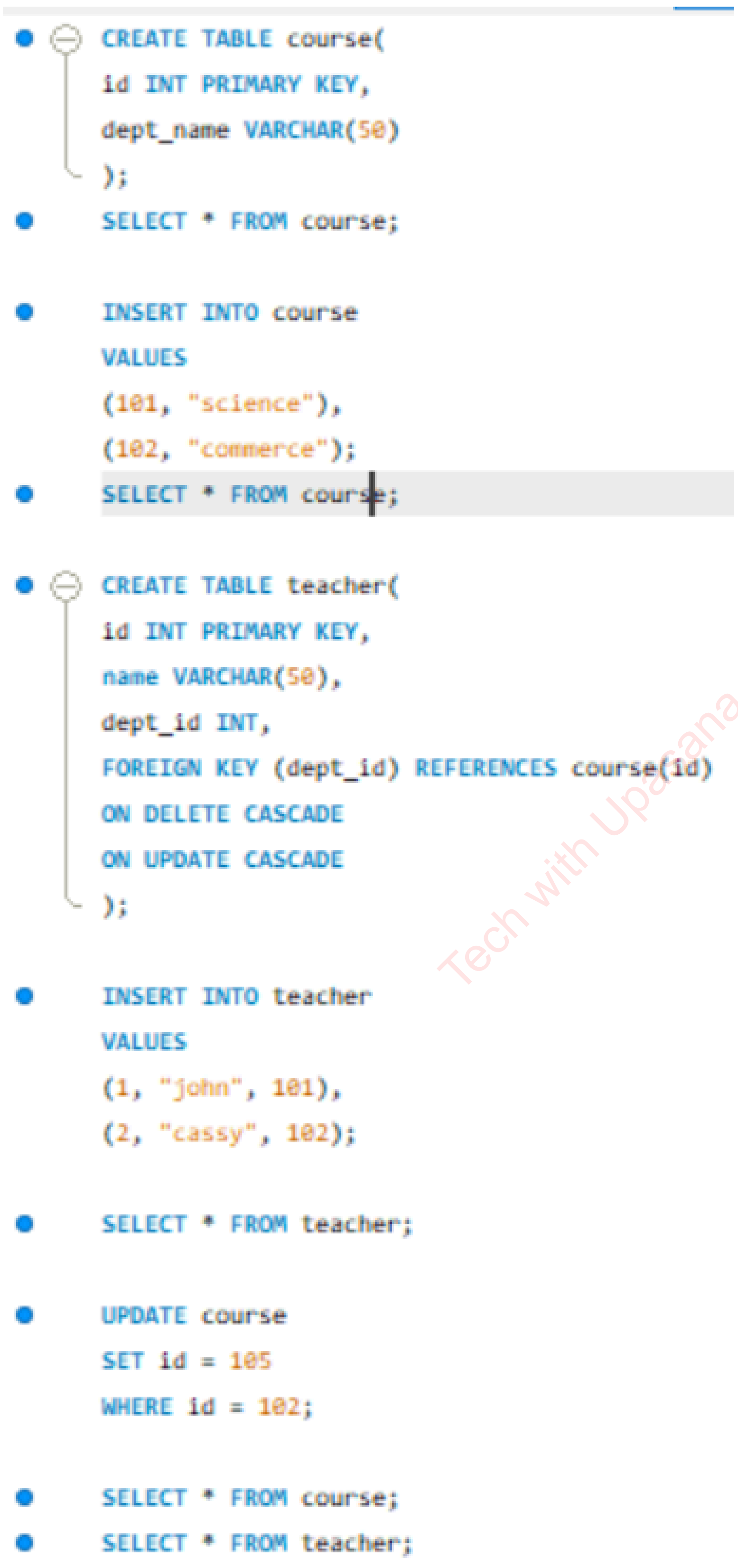
dept

id	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

CASCADING - ek table ke andar se kuch update hora h to dusri se bhi update ho jaye



```sql
CREATE TABLE course(
id INT PRIMARY KEY,
dept_name VARCHAR(50)
);
SELECT * FROM course;

INSERT INTO course
VALUES
(101, "science"),
(102, "commerce");
SELECT * FROM course;

CREATE TABLE teacher(
id INT PRIMARY KEY,
name VARCHAR(50),
dept_id INT,
FOREIGN KEY (dept_id) REFERENCES course(id)
ON DELETE CASCADE
ON UPDATE CASCADE
);

INSERT INTO teacher
VALUES
(1, "john", 101),
(2, "cassy", 102);

SELECT * FROM teacher;

UPDATE course
SET id = 105
WHERE id = 102;

SELECT * FROM course;
SELECT * FROM teacher;
```

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

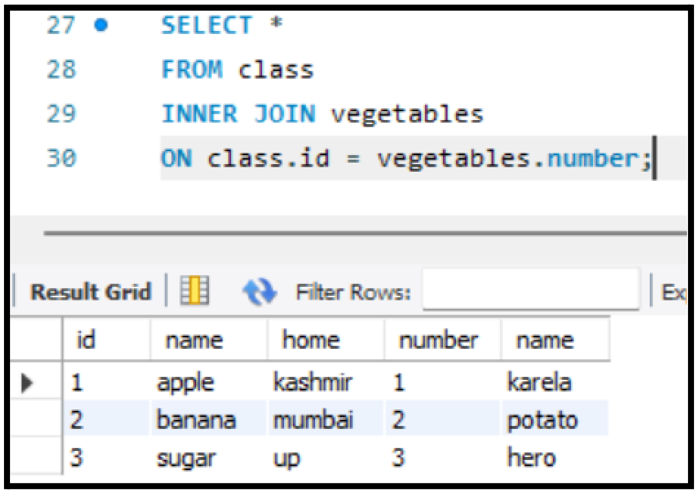
```
ALTER TABLE student
DROP city;
```

```
ALTER TABLE student
RENAME TO stud;
```

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

JOINS - are used to combine the rows from 2 or more tables based on the similar column b/w them.

Inner Join - common b/w a & b



```
SELECT *
FROM class
INNER JOIN vegetables
ON class.id = vegetables.number;
```

id	name	home	number	name
1	apple	kashmir	1	karela
2	banana	mumbai	2	potato
3	sugar	up	3	hero

UNION - will give unique common name
UNION ALL - saari values dega including duplicates

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

```sql
SELECT * FROM student;

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