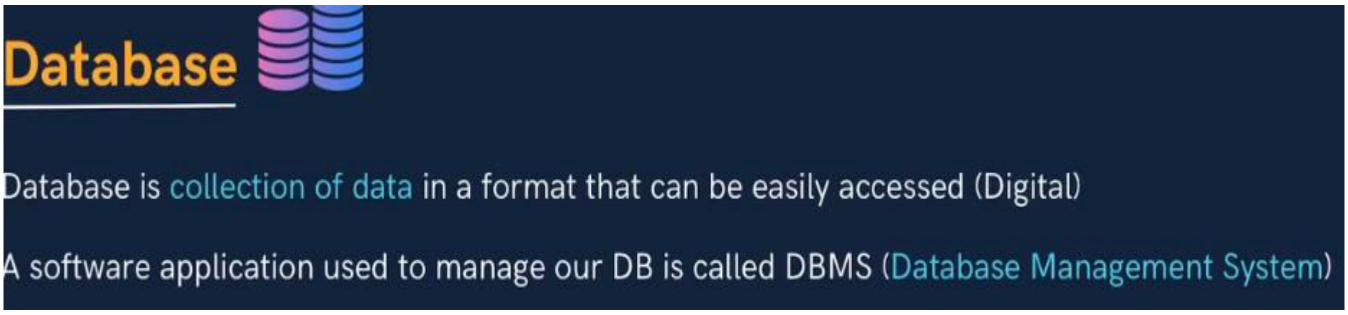
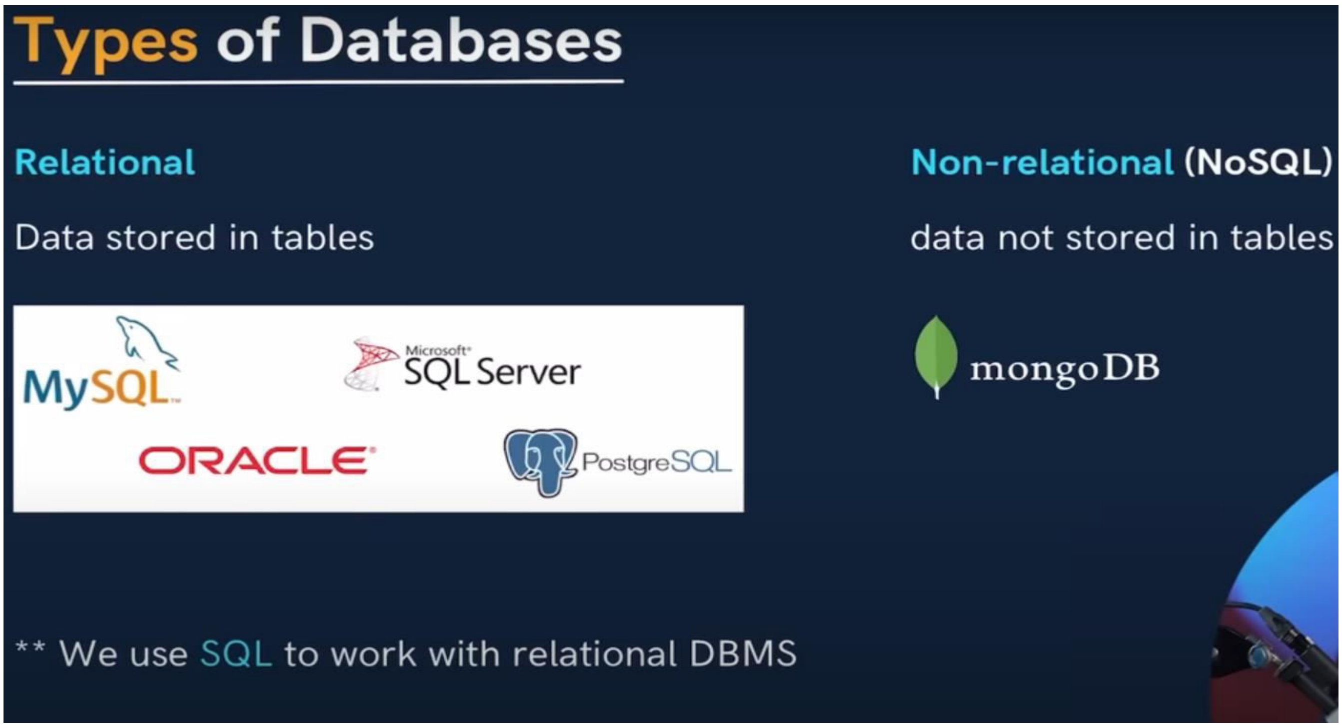
SQL BY DIDI



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)



Types of Databases

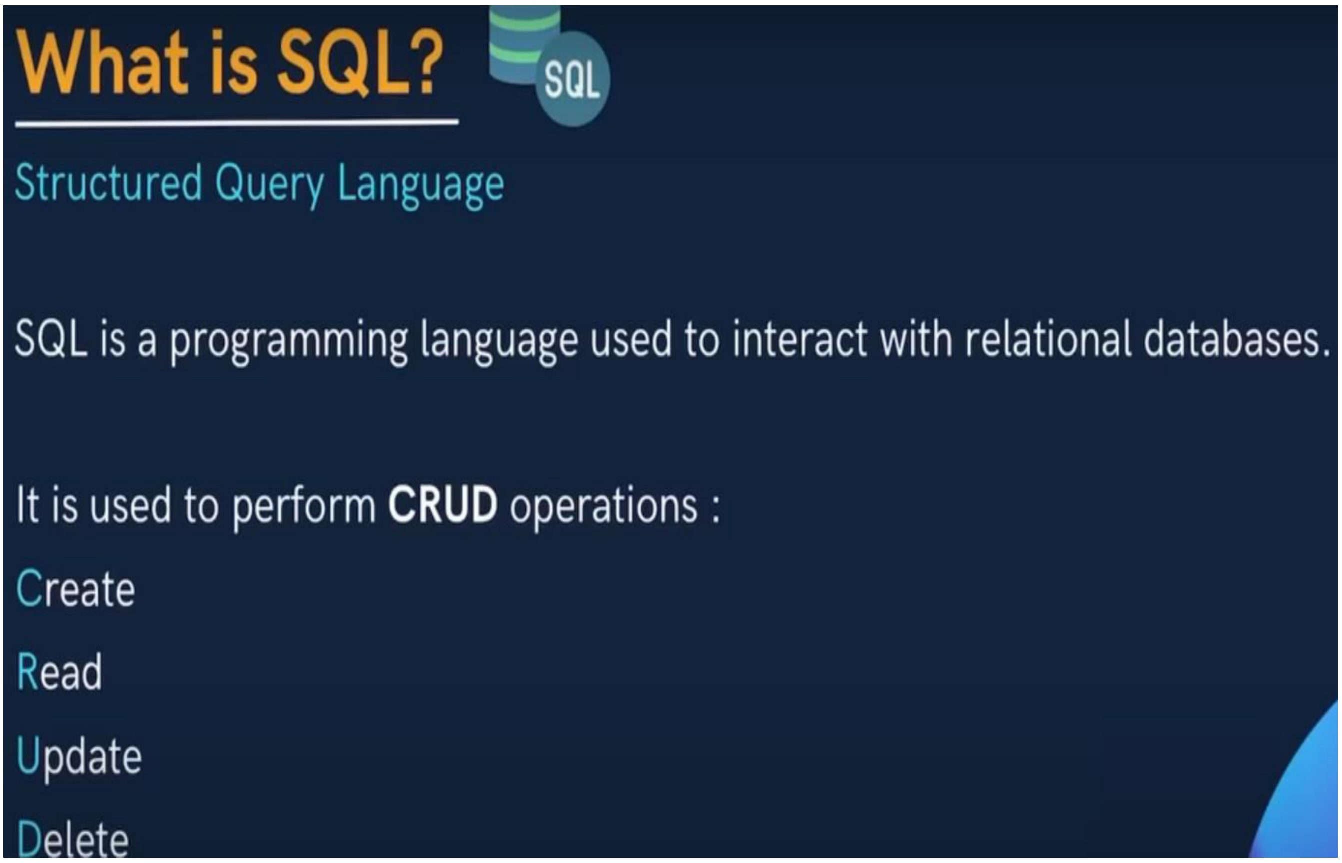
Relational

Data stored in tables

Non-relational (NoSQL)

data not stored in tables

** We use SQL to work with relational DBMS



What is SQL?

Structured Query Language

SQL is a programming language used to interact with relational databases.

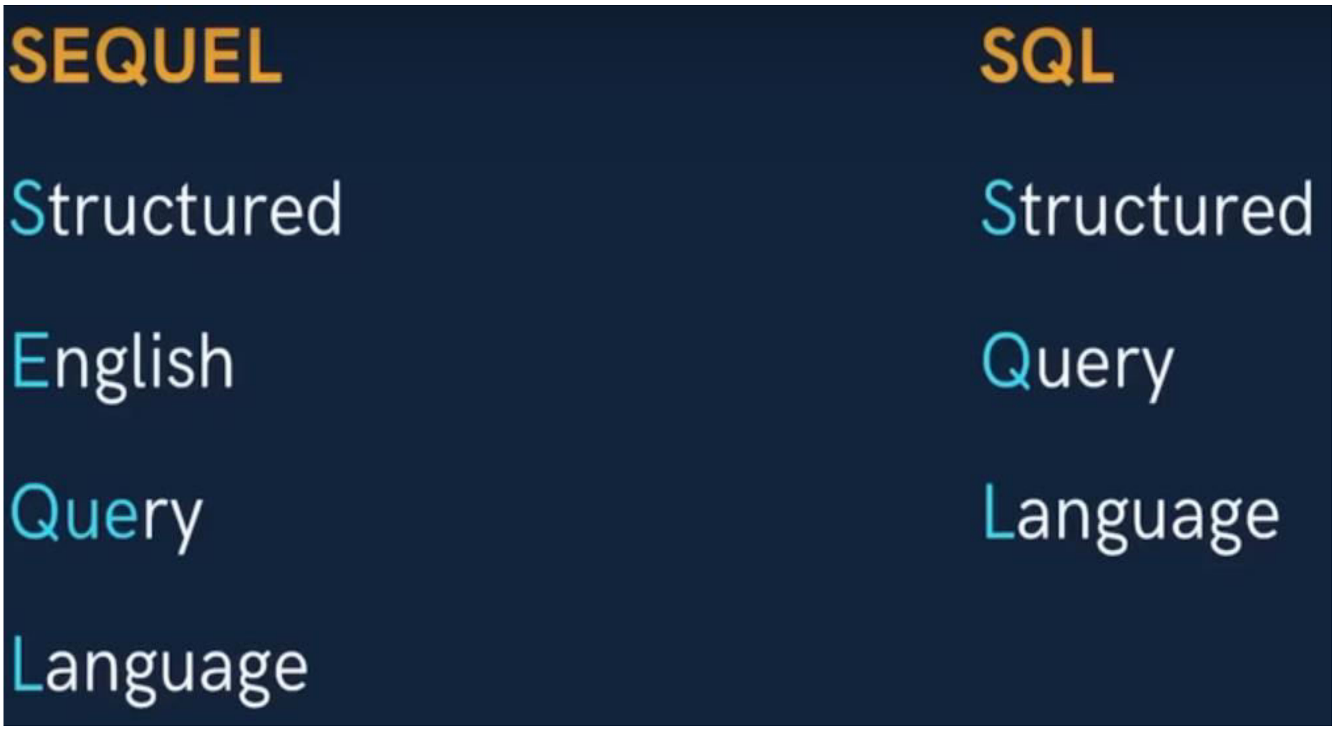
It is used to perform **CRUD** operations :

Create

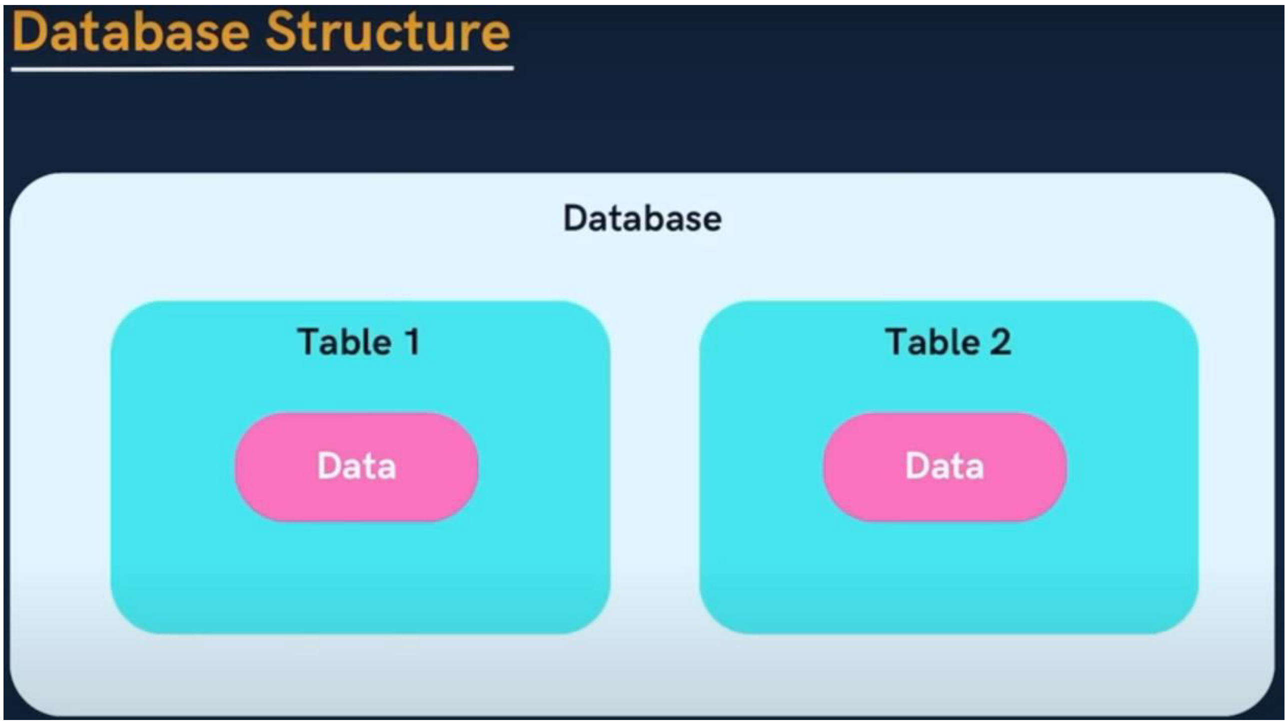
Read

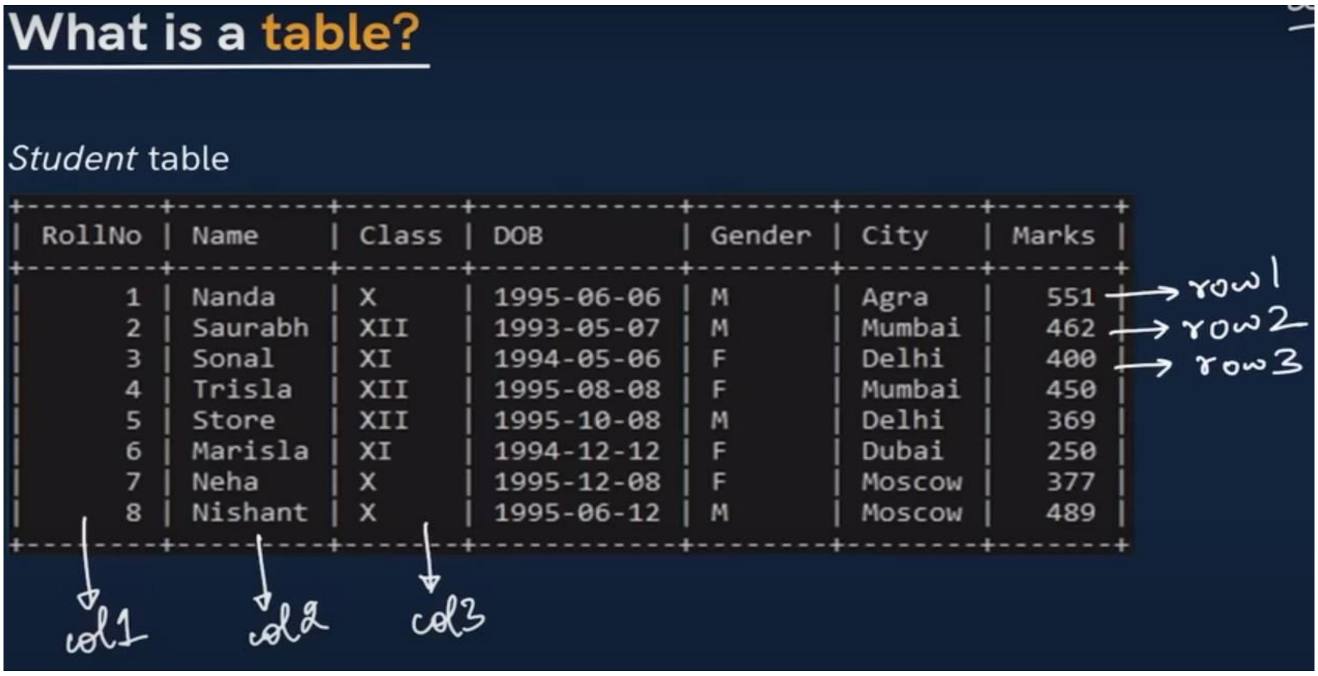
Update

Delete



SEQUEL	SQL
Structured	Structured
English	Query
Query	Language
Language	

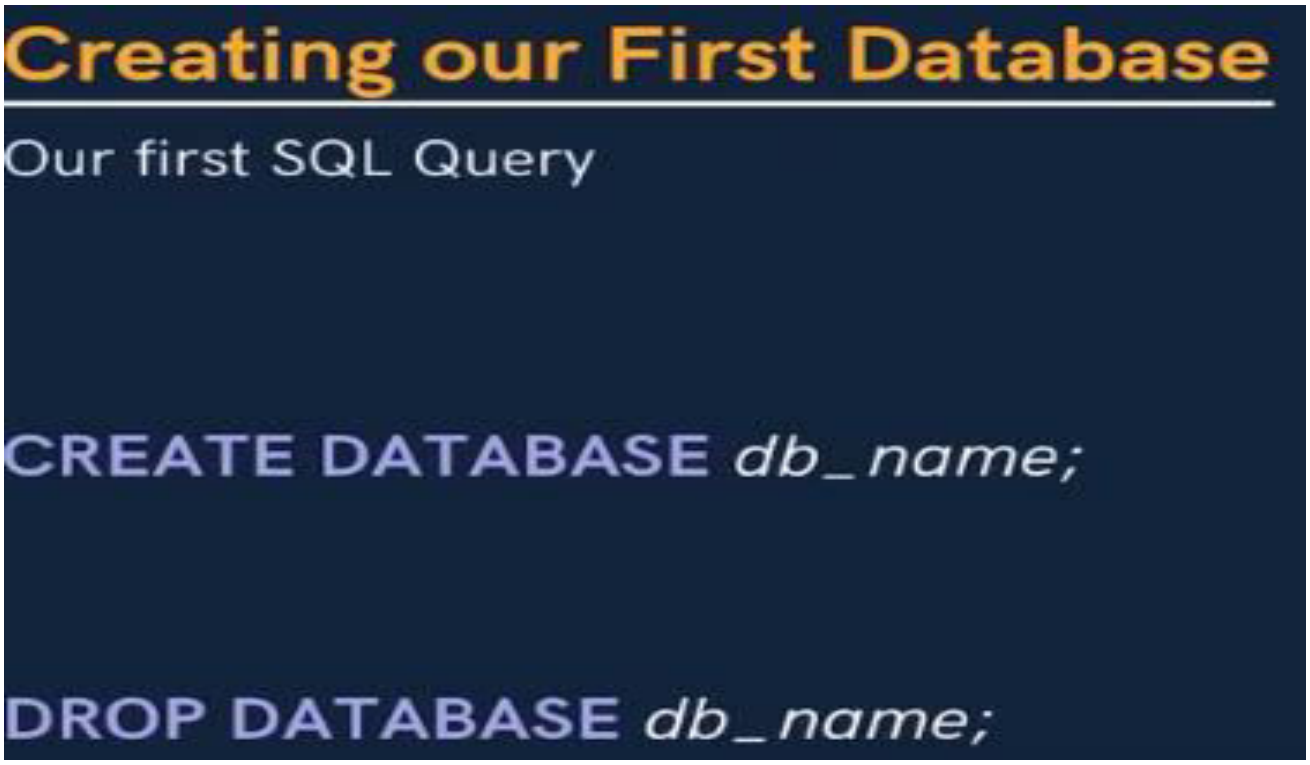



What is a table?

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 |
+--------+---------+------+------------+--------+--------+-------+
```

Columns tell the schema/structure of a table.



Creating our First Database

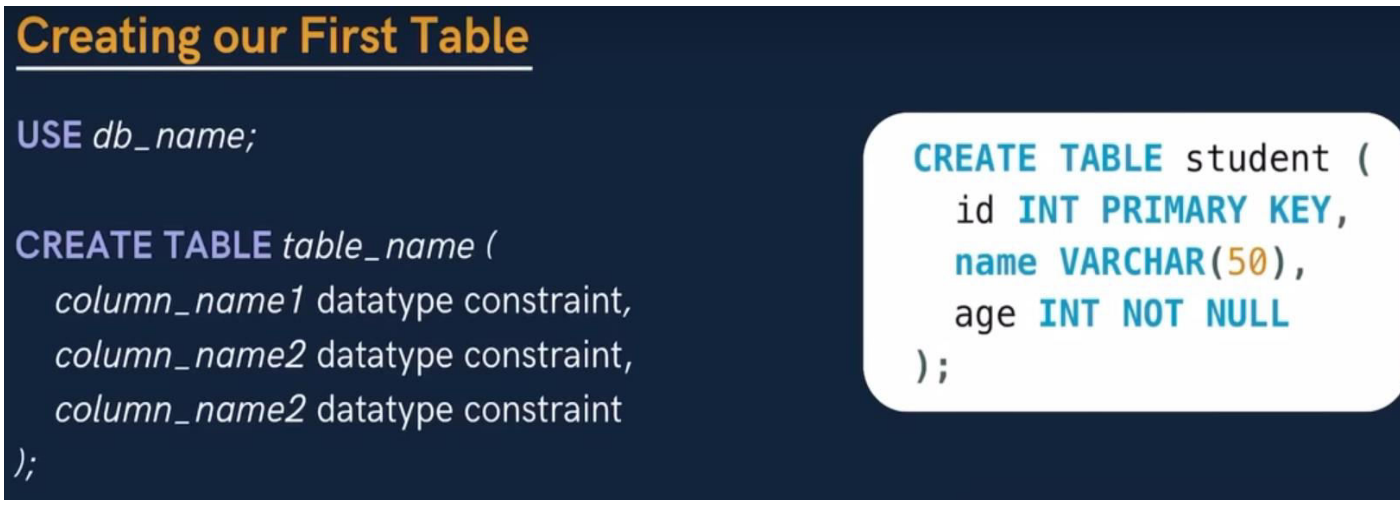
Our first SQL Query

```
CREATE DATABASE db_name;

DROP DATABASE db_name;
```

DROP will delete the database of the provided name.

Now we will see how we can create a table In SQL.



Creating our First Table

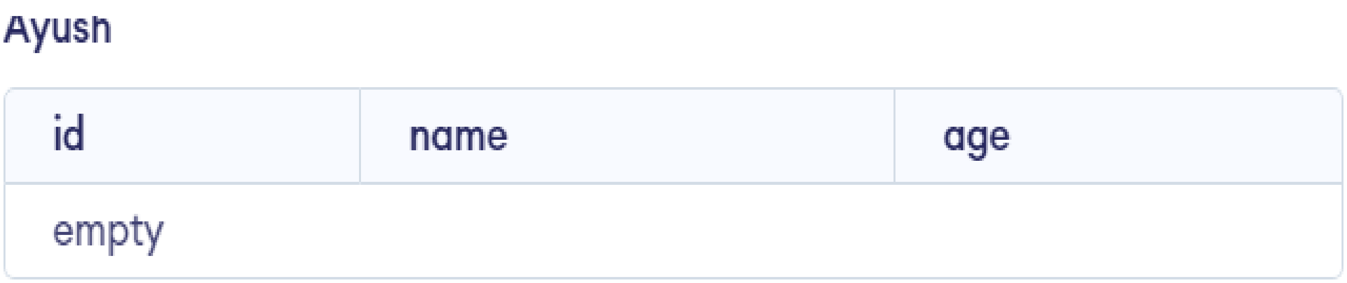
```
USE db_name;

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

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

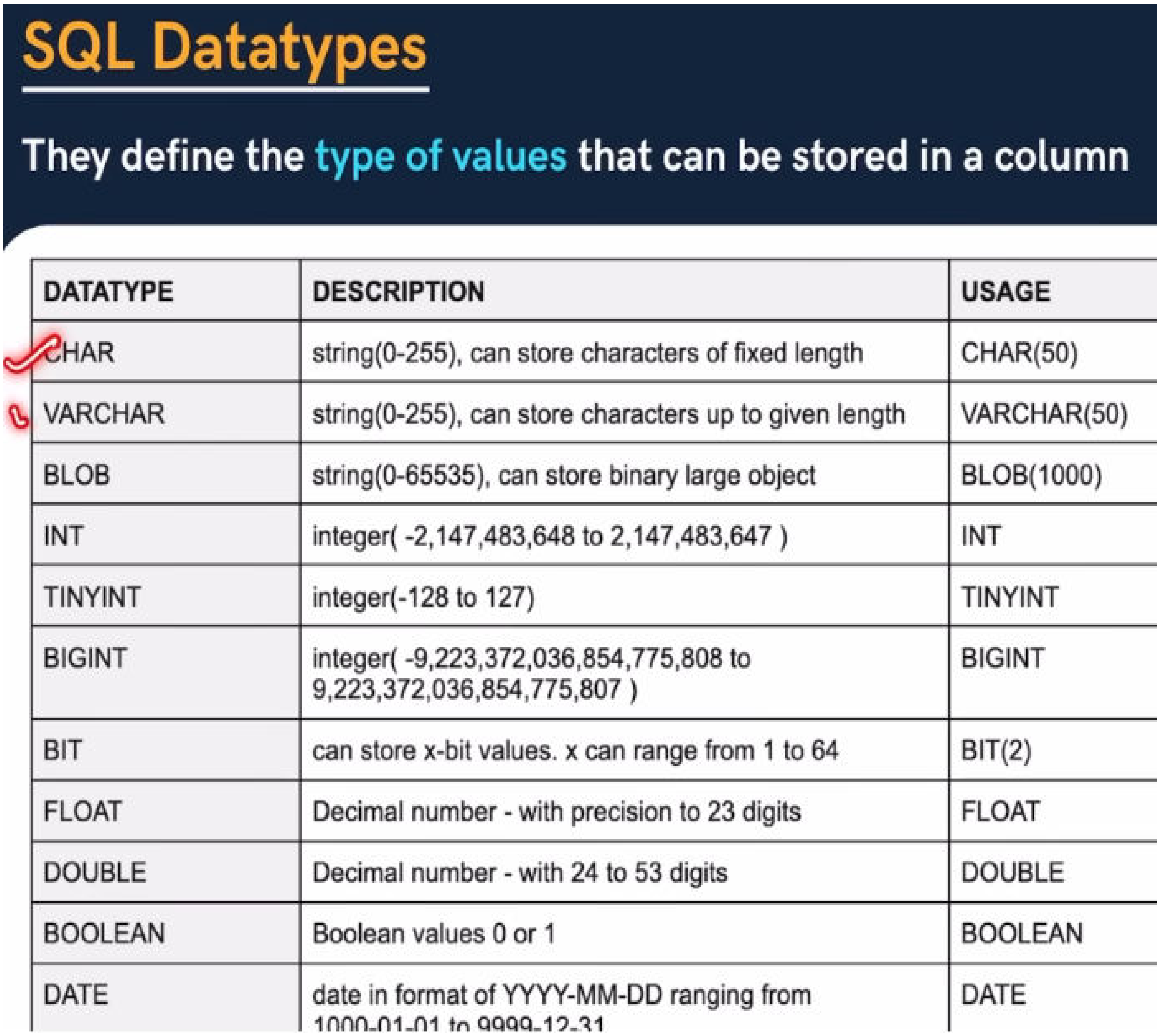
Ex:-

```
CREATE TABLE ayush( id INT PRIMARY kEY, name VARCHAR(20),age INT NOT NULL);
```



Ayush

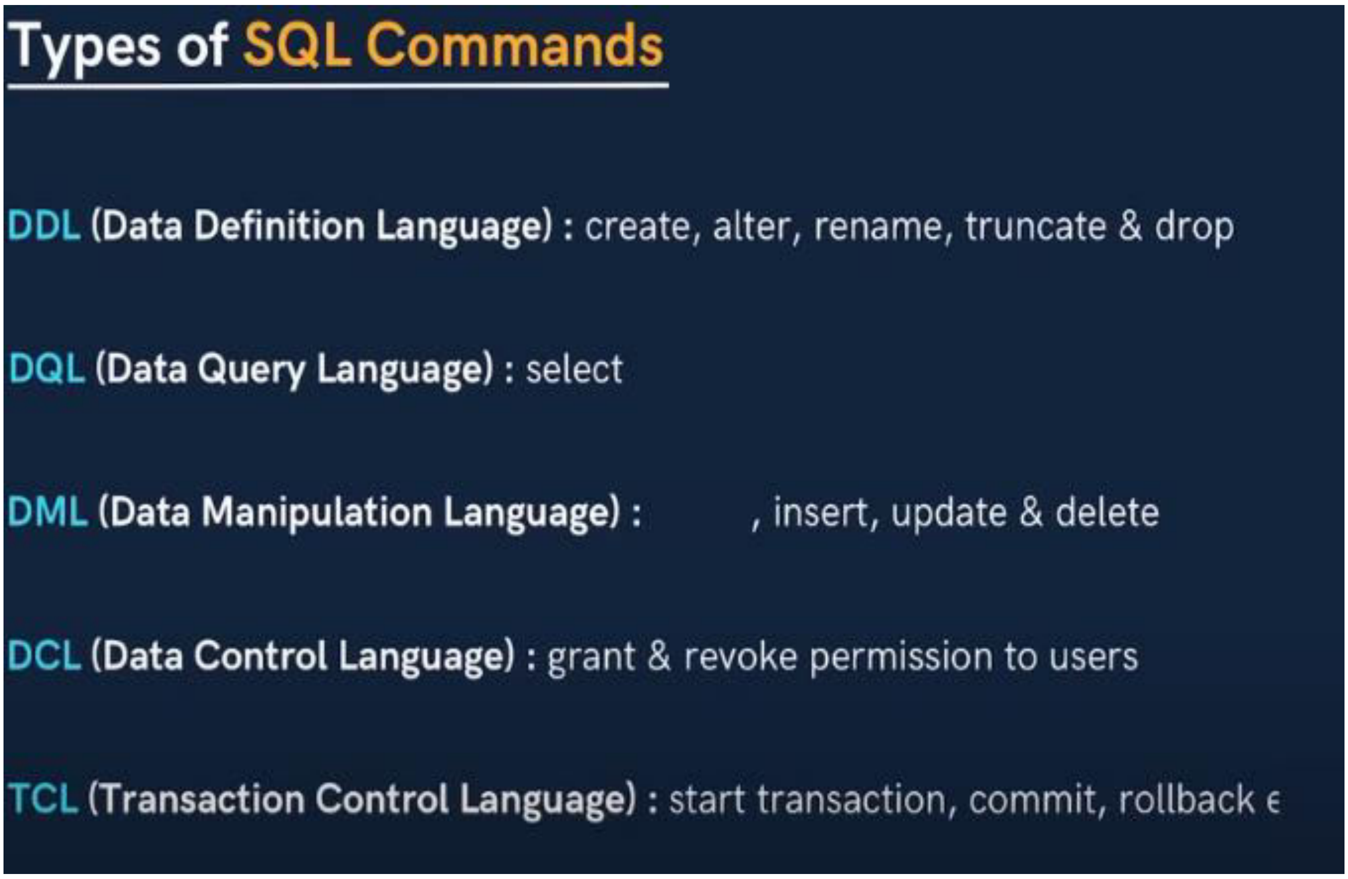
id	name	age
empty		



SQL Datatypes

They define the type of values that can be stored in a column

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



Types of SQL Commands

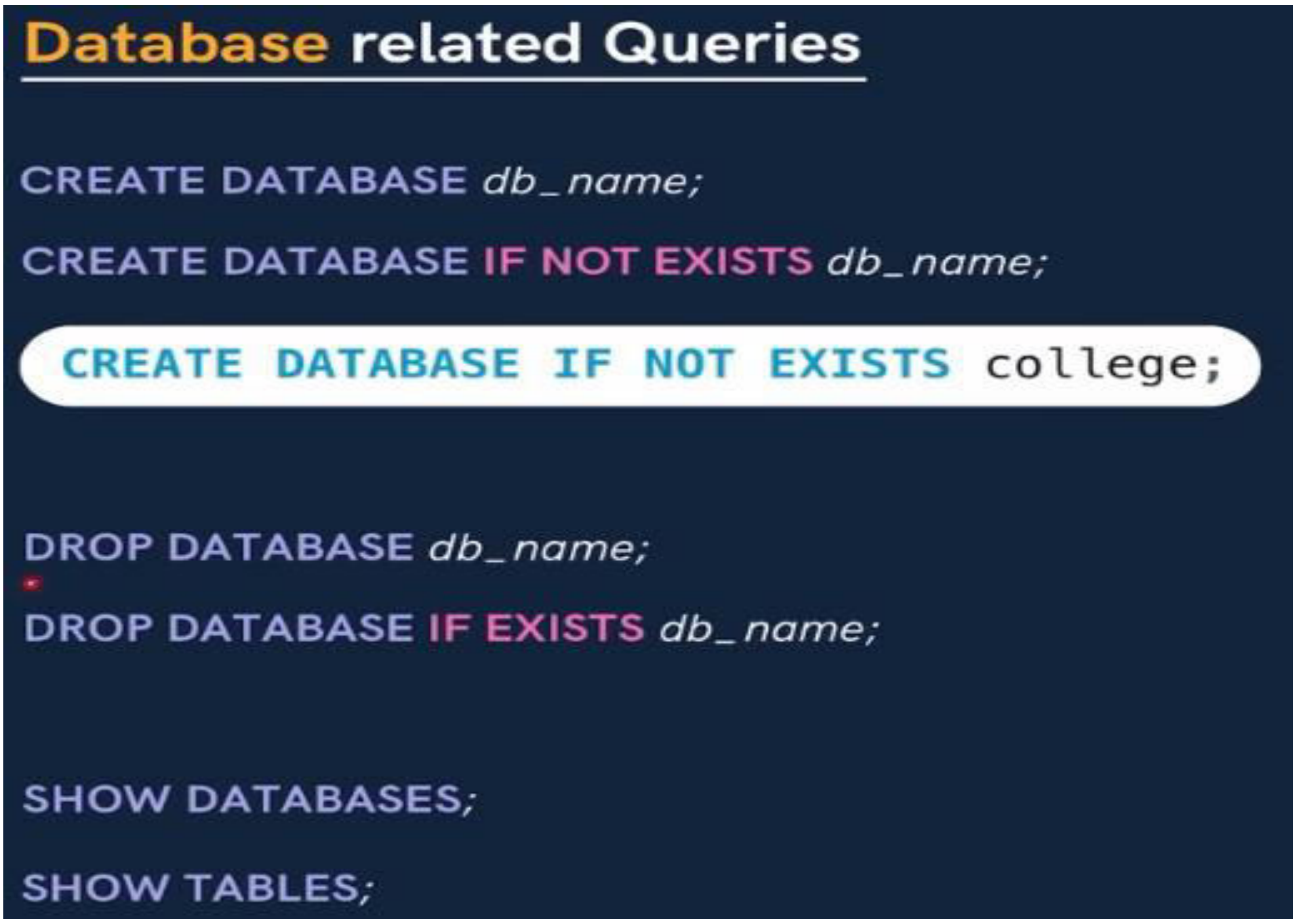
DDL (Data Definition Language) : create, alter, rename, truncate & drop

DQL (Data Query Language) : select

DML (Data Manipulation Language) : , insert, update & delete

DCL (Data Control Language) : grant & revoke permission to users

TCL (Transaction Control Language) : start transaction, commit, rollback



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;

SHOW TABLES;
```

Code works only if, when if exist/if not exist condition is true, otherwise an error will be thrown.

<u>TABLE RELATED QUERIES</u>:-

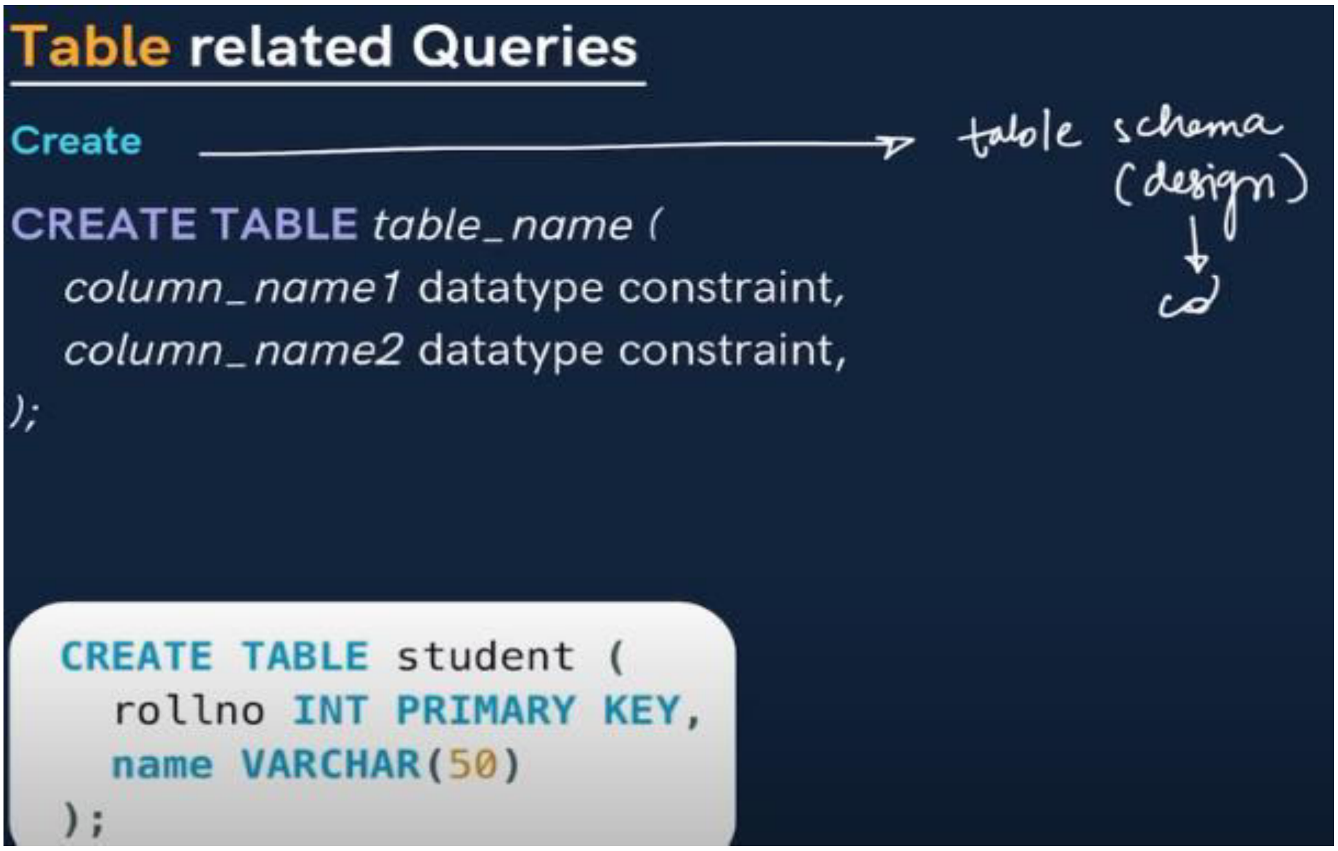
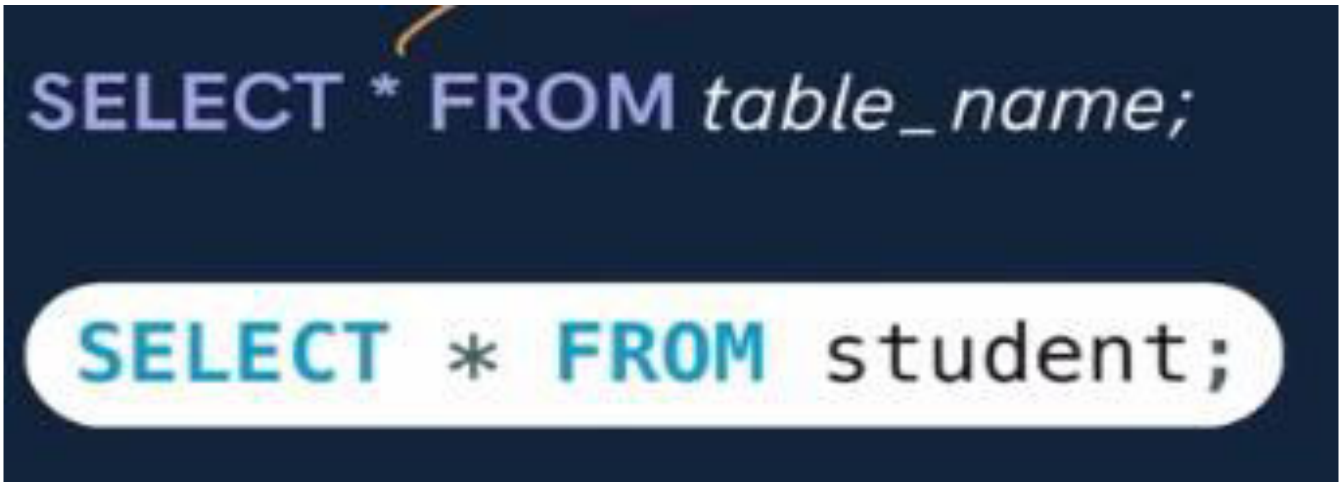


Table related Queries

Create → table schema (design)

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

```
CREATE TABLE student (
  rollno INT PRIMARY KEY,
  name VARCHAR(50)
);
```

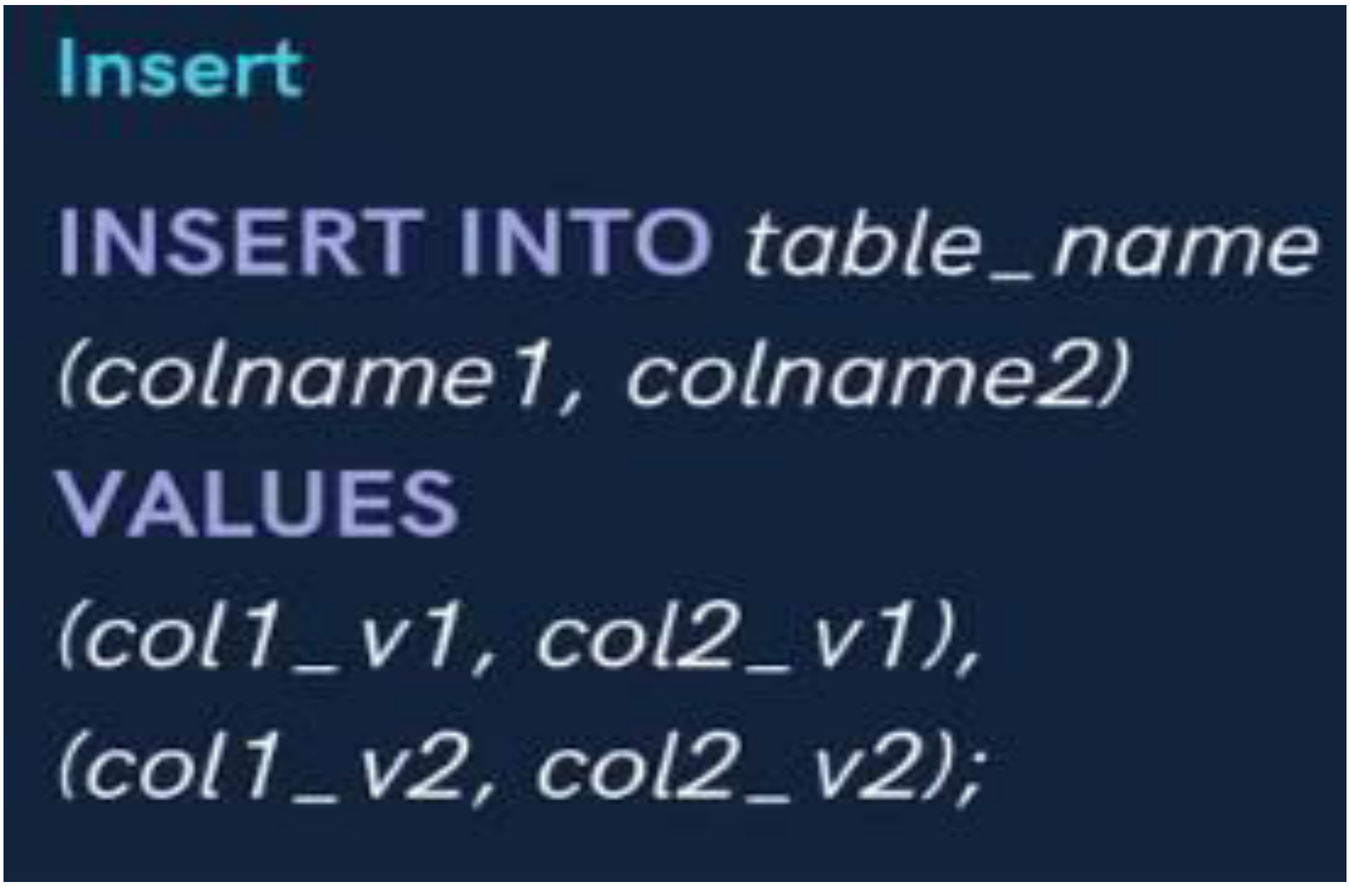


```
SELECT * FROM table_name;
```

```
SELECT * FROM student;
```

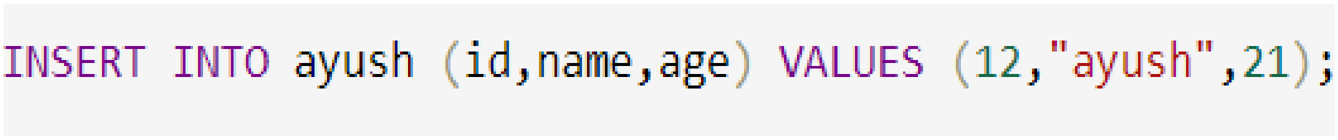
(*) means to select everything from the table.

Method to insert values into the table:-

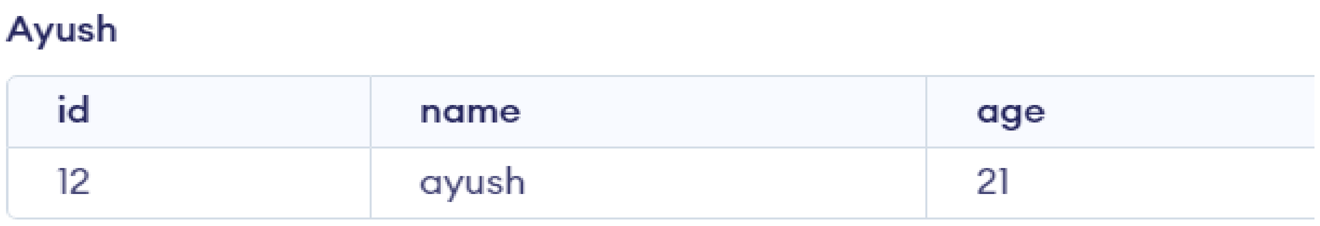


Insert

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

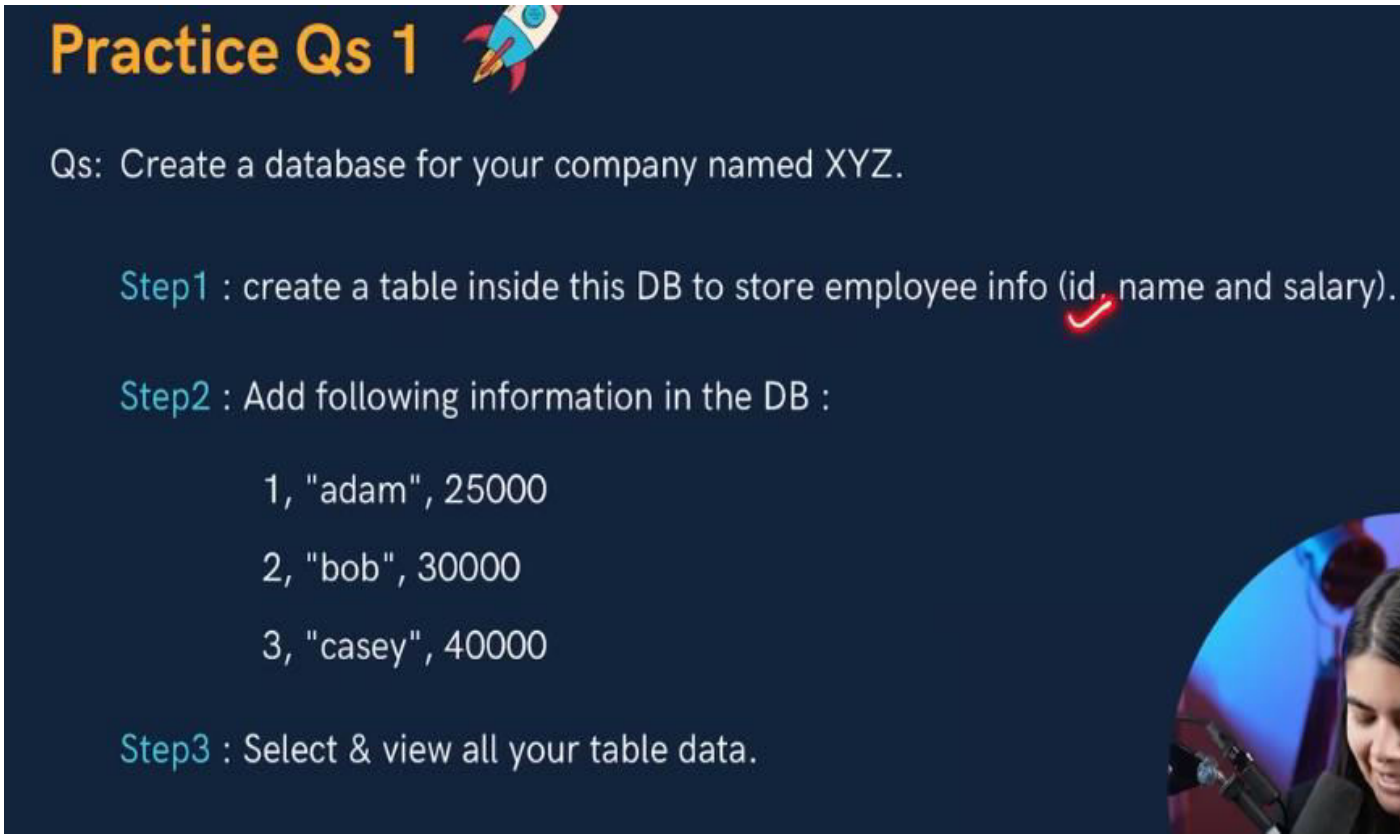


```
INSERT INTO ayush (id,name,age) VALUES (12,"ayush",21);
```



Ayush

id	name	age
12	ayush	21



Practice Qs 1

Qs: Create a database for your company named XYZ.

Step1 : create a table inside this DB to store employee info (id, name and salary).

Step2 : Add following information in the DB :

1, "adam", 25000

2, "bob", 30000

3, "casey", 40000

Step3 : Select & view all your table data.

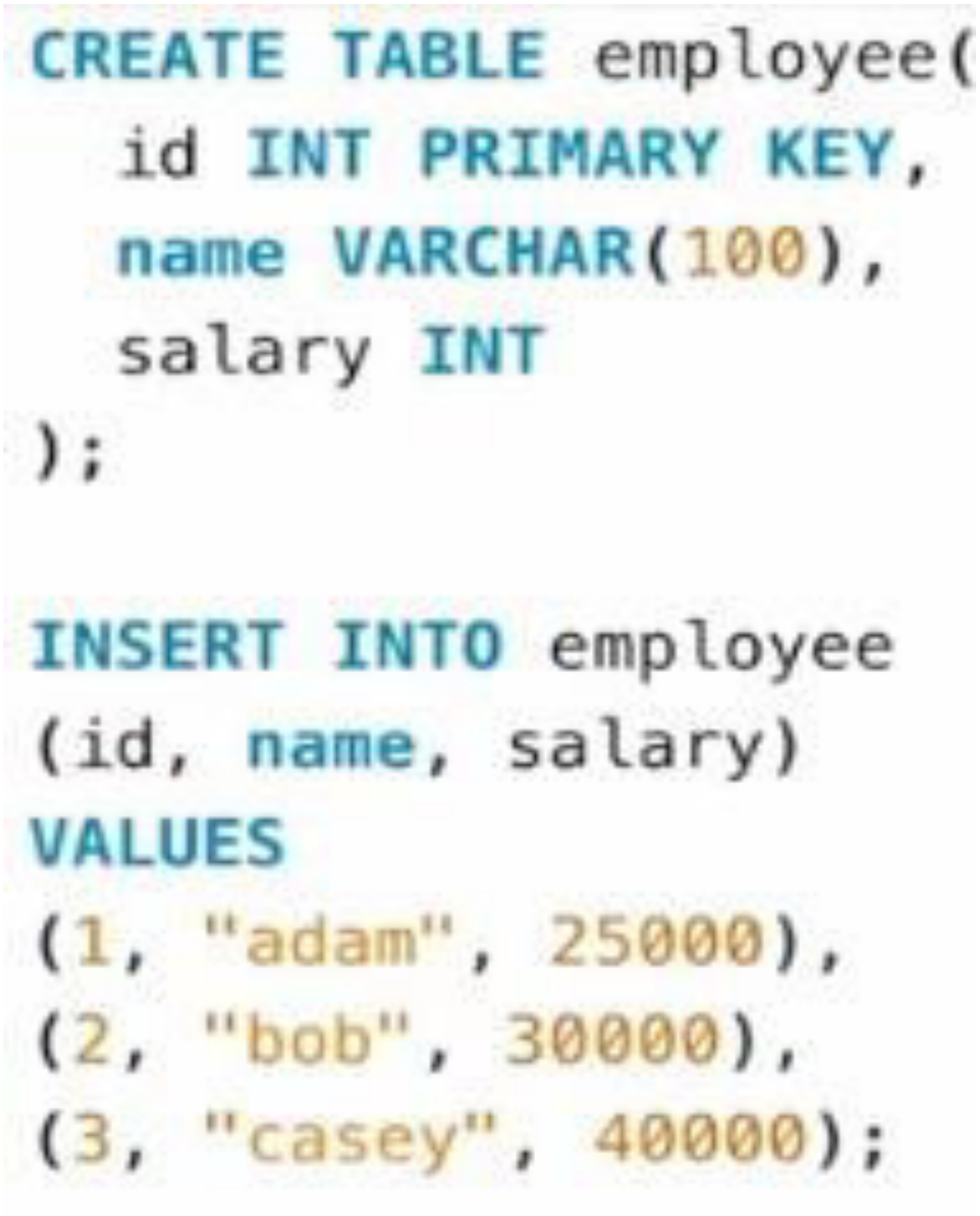
```
CREATE TABLE employee (
  id INT PRIMARY KEY,
  name VARCHAR(50),
  salary INT NOT NULL);

INSERT INTO employee (id,name,salary) VALUES (1,"adam",25000);
INSERT INTO employee (id,name,salary) VALUES (2,"bob",30000);
INSERT INTO employee (id,name,salary) VALUES (3,"casey",40000);
```

Employee

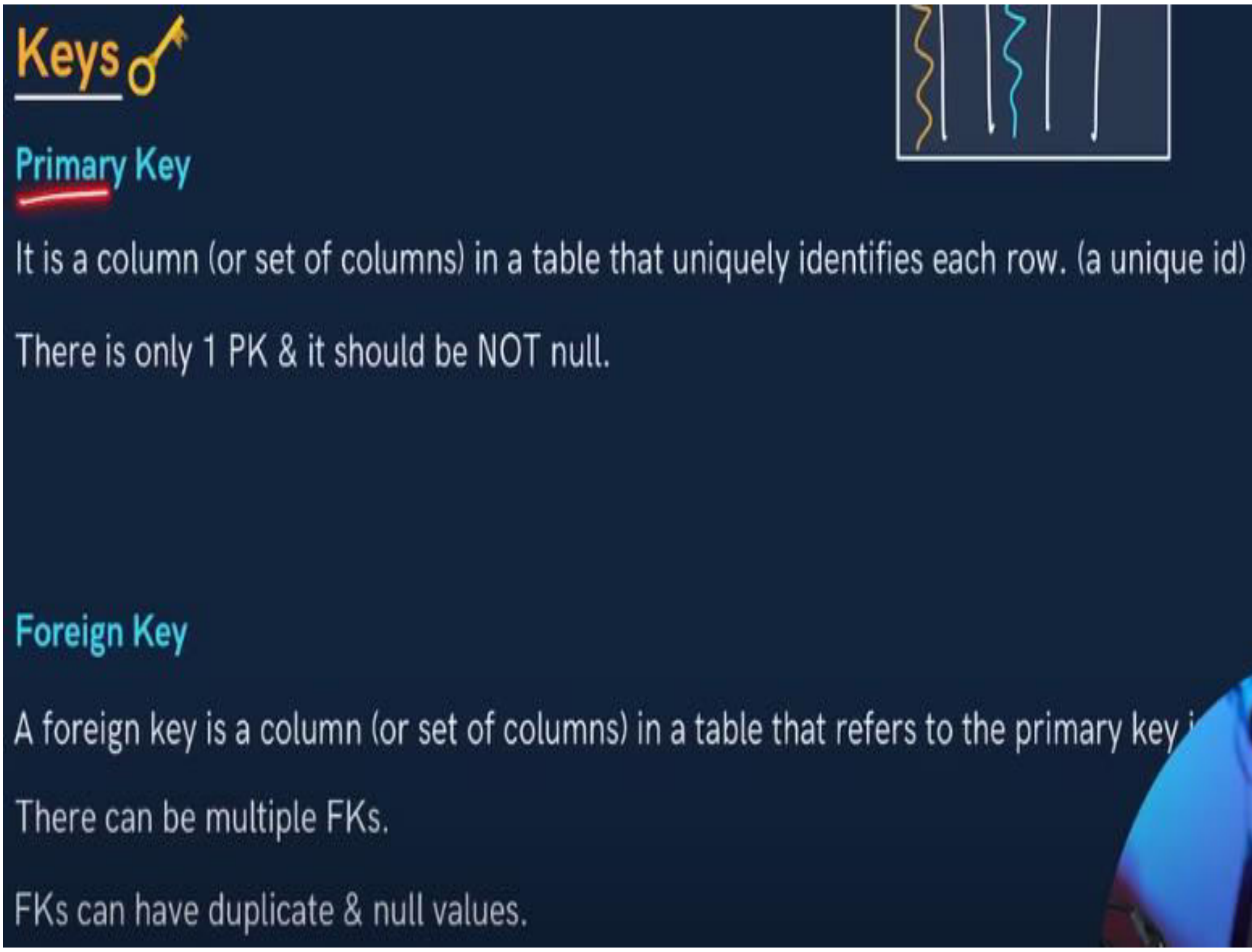
id	name	salary
1	adam	25000
2	bob	30000
3	casey	40000

You can use this syntax too:-



```
CREATE TABLE employee(
  id INT PRIMARY KEY,
  name VARCHAR(100),
  salary INT
);

INSERT INTO employee
(id, name, salary)
VALUES
(1, "adam", 25000),
(2, "bob", 30000),
(3, "casey", 40000);
```



Keys

Primary Key

It is a column (or set of columns) in a table that uniquely identifies each row. (a unique id)

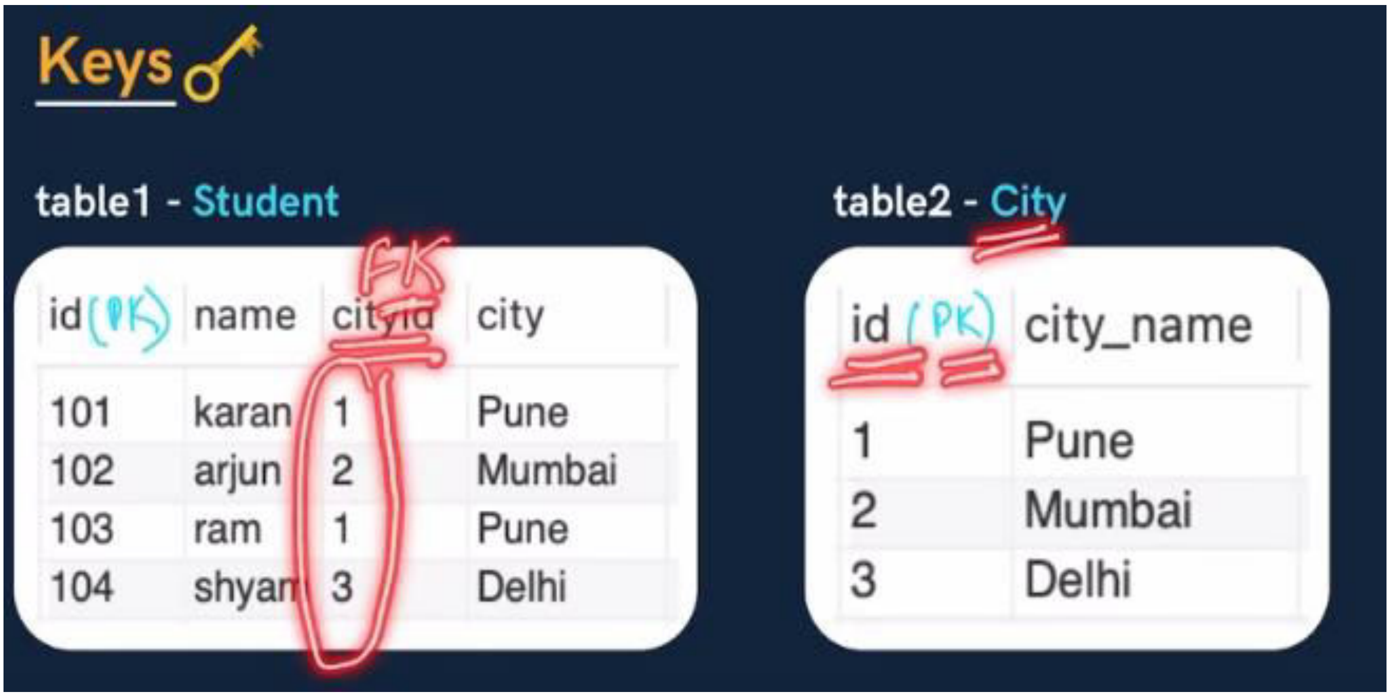
There is only 1 PK & it should be NOT null.

Foreign Key

A foreign key is a column (or set of columns) in a table that refers to the primary key i

There can be multiple FKs.

FKs can have duplicate & null values.



Keys

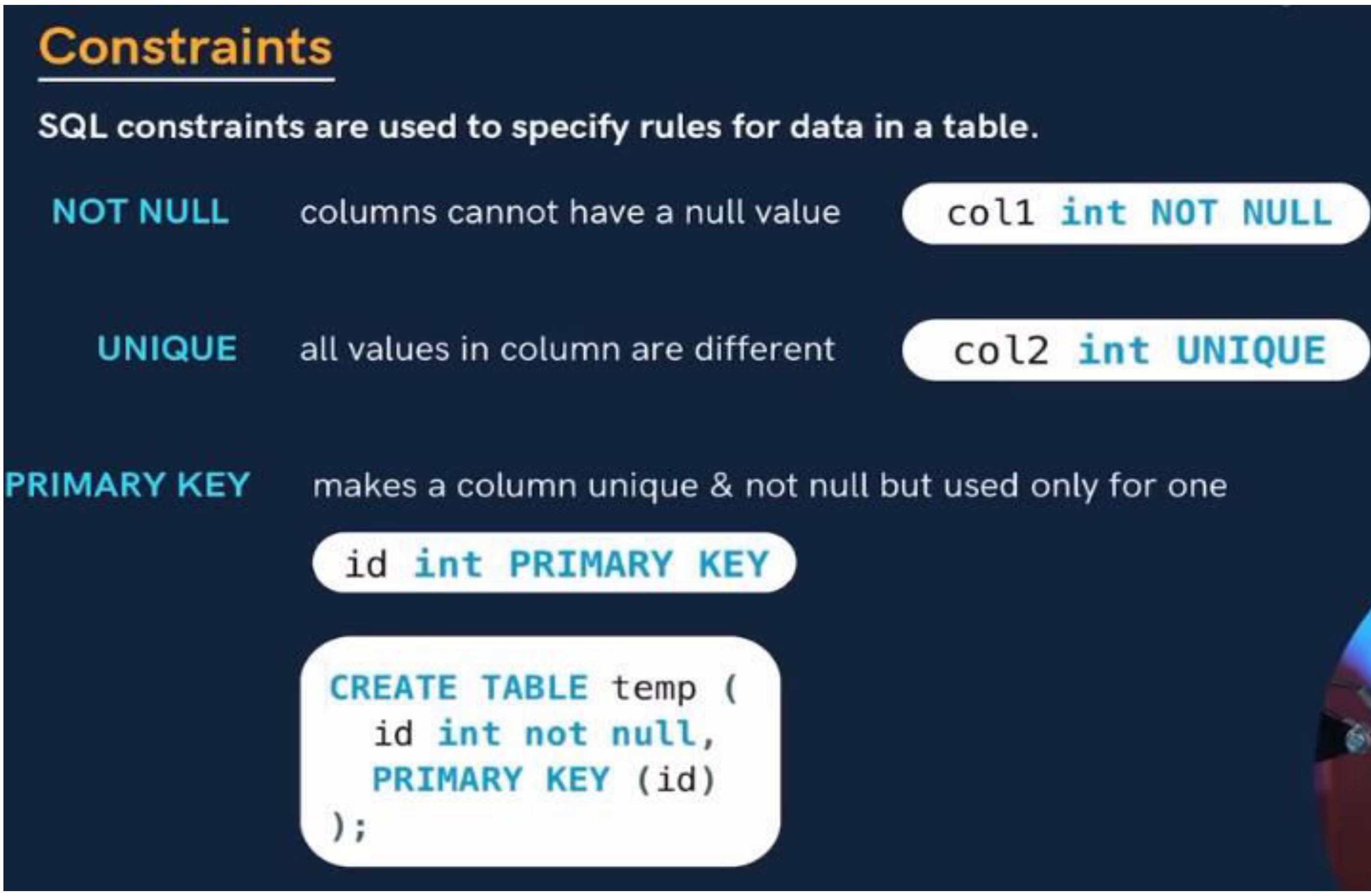
table1 - Student

id(PK)	name	cityid (FK)	city
101	karan	1	Pune
102	arjun	2	Mumbai
103	ram	1	Pune
104	shyam	3	Delhi

table2 - City

id (PK)	city_name
1	Pune
2	Mumbai
3	Delhi

Primary key wo h jo hr row ko uniquely identify karwae, pr foreign key wo h joki kisi aur table ka primary key column h. for example table2 ke id colum ke andar jo values h wo unique h and wo values table1 ke cityid se liye gae h, to cityid wala column foreign Key hua becoz wo kisi aur table ka primary key h. foreign key khud null store krskta h, pr use liye hue values jo kisi aur ka table ke primary key column me jayenge wo null nhi hoskte, bacoz primary key cant tolerate null values.



Constraints

SQL constraints are used to specify rules for data in a table.

NOT NULL columns cannot have a null value

```
col1 int NOT NULL
```

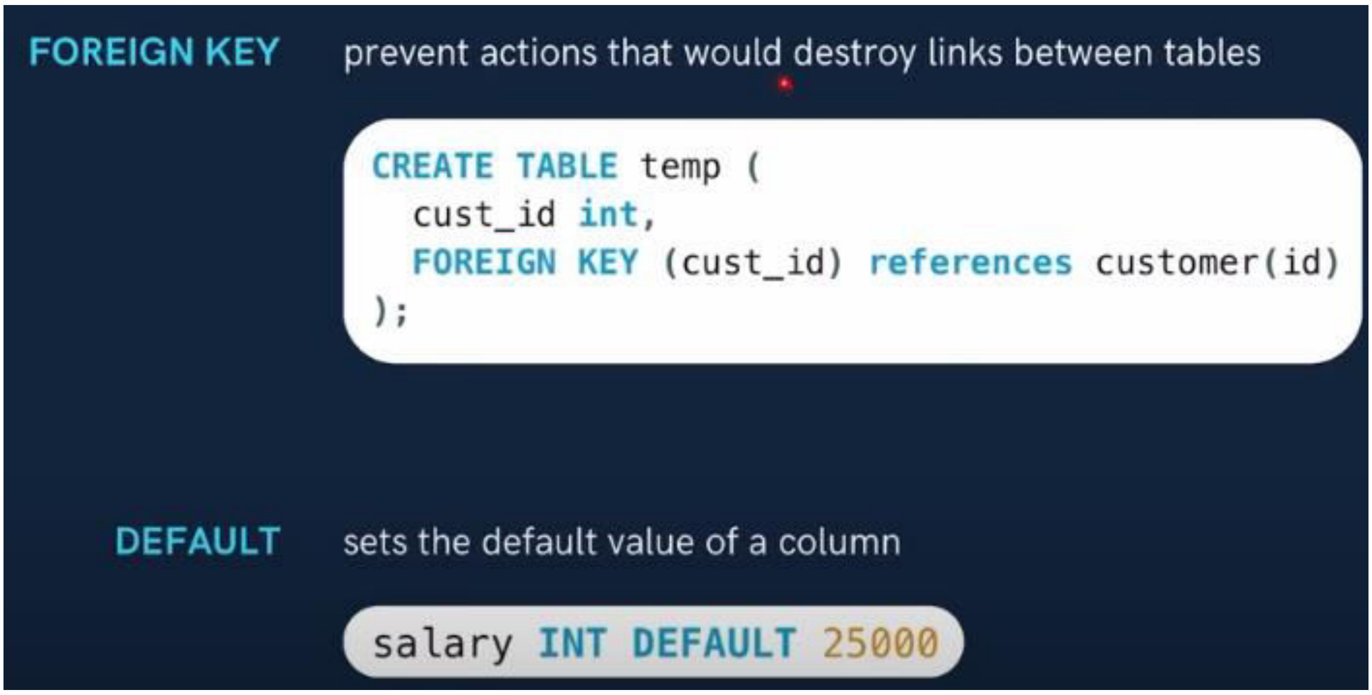
UNIQUE all values in column are different

```
col2 int UNIQUE
```

PRIMARY KEY makes a column unique & not null but used only for one

```
id int PRIMARY KEY
```

```
CREATE TABLE temp (
  id int not null,
  PRIMARY KEY (id)
);
```



FOREIGN KEY prevent actions that would destroy links between tables

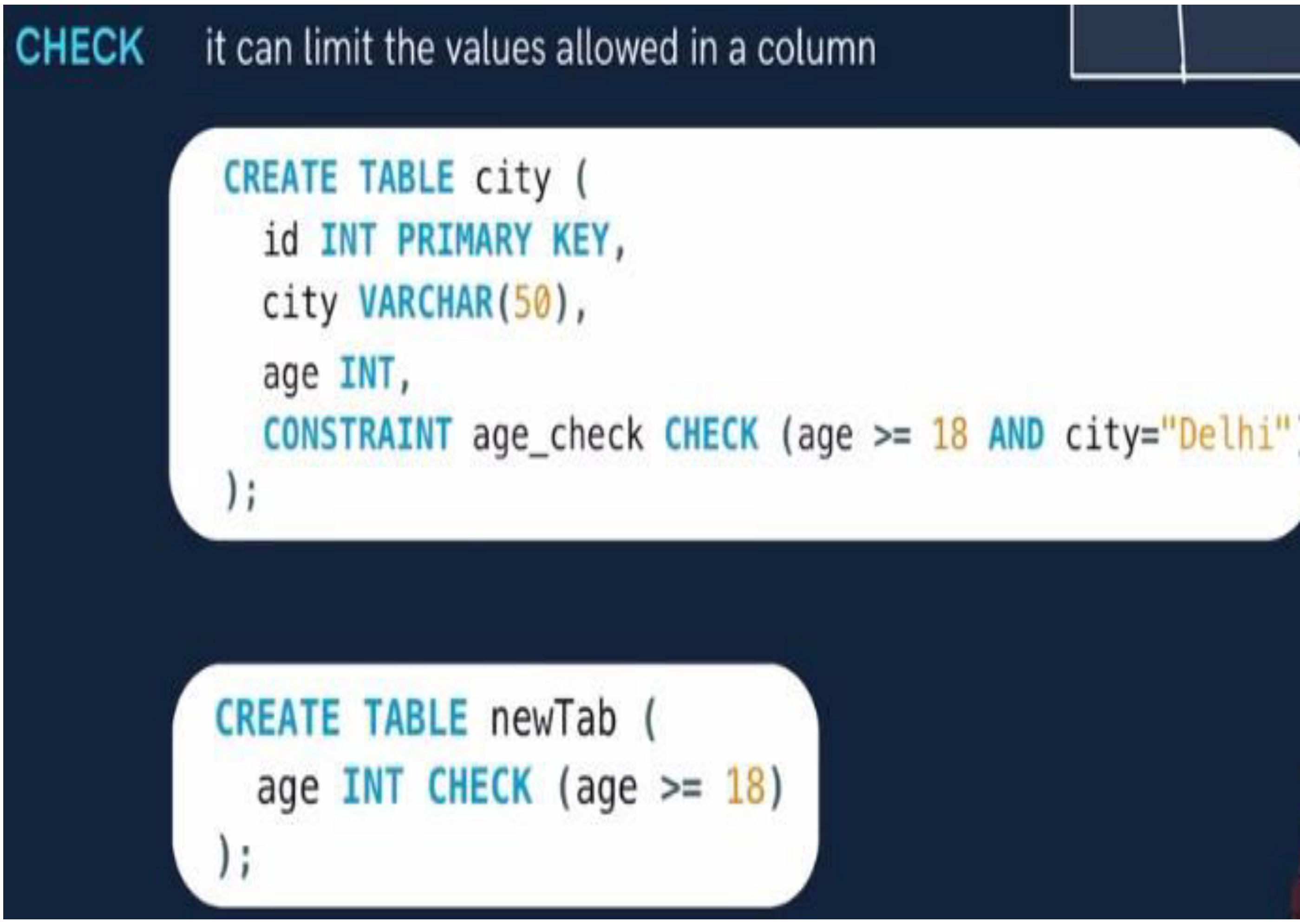
```
CREATE TABLE temp (
  cust_id int,
  FOREIGN KEY (cust_id) references customer(id)
);
```

DEFAULT sets the default value of a column

```
salary INT DEFAULT 25000
```

Here we declare cust_id a foreign key, and values will be taken from the table of customer in which it has the primary key column named (id).

Default will put the default value in the entire column.

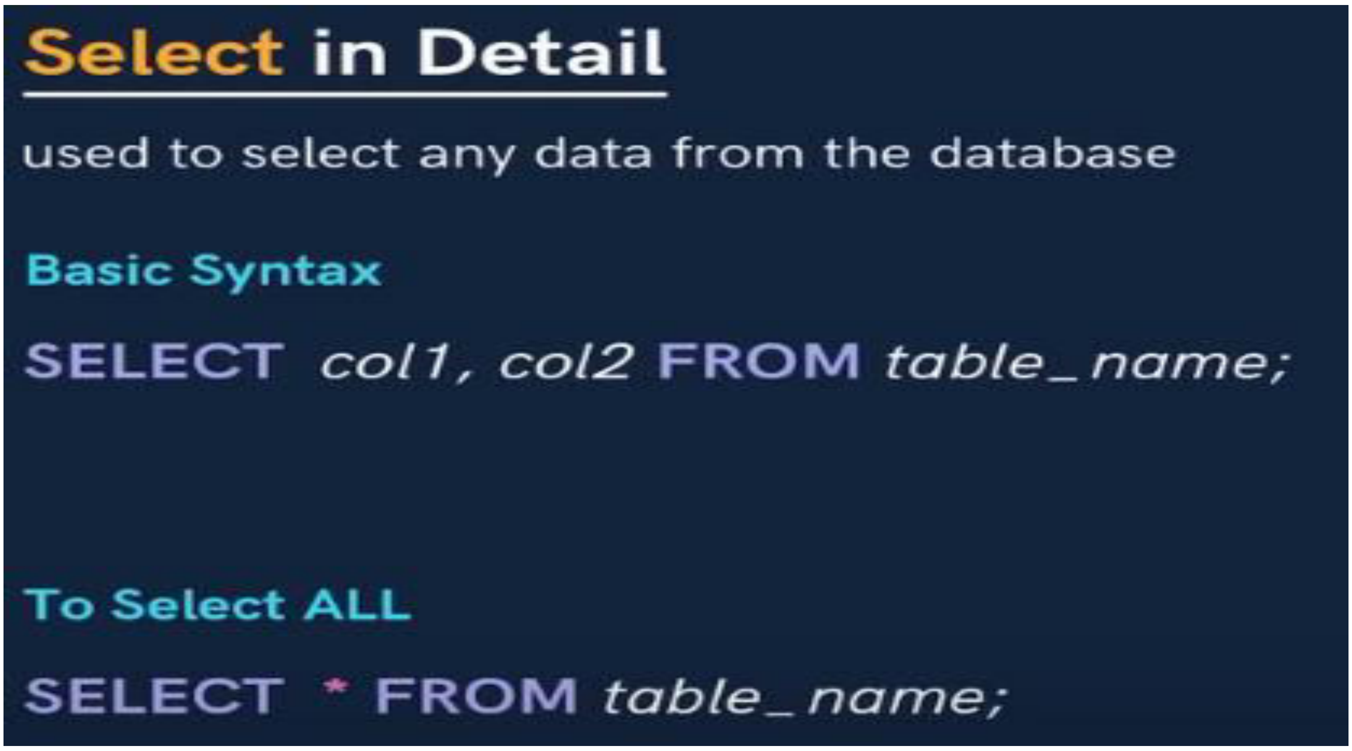

CHECK it can limit the values allowed in a column

```
CREATE TABLE city (
  id INT PRIMARY KEY,
  city VARCHAR(50),
  age INT,
  CONSTRAINT age_check CHECK (age >= 18 AND city="Delhi"
);
```

```
CREATE TABLE newTab (
  age INT CHECK (age >= 18)
);
```

Values are only allowed according to the check constraints.

Now we will learn about selection in the table:-



Select in Detail

used to select any data from the database

Basic Syntax

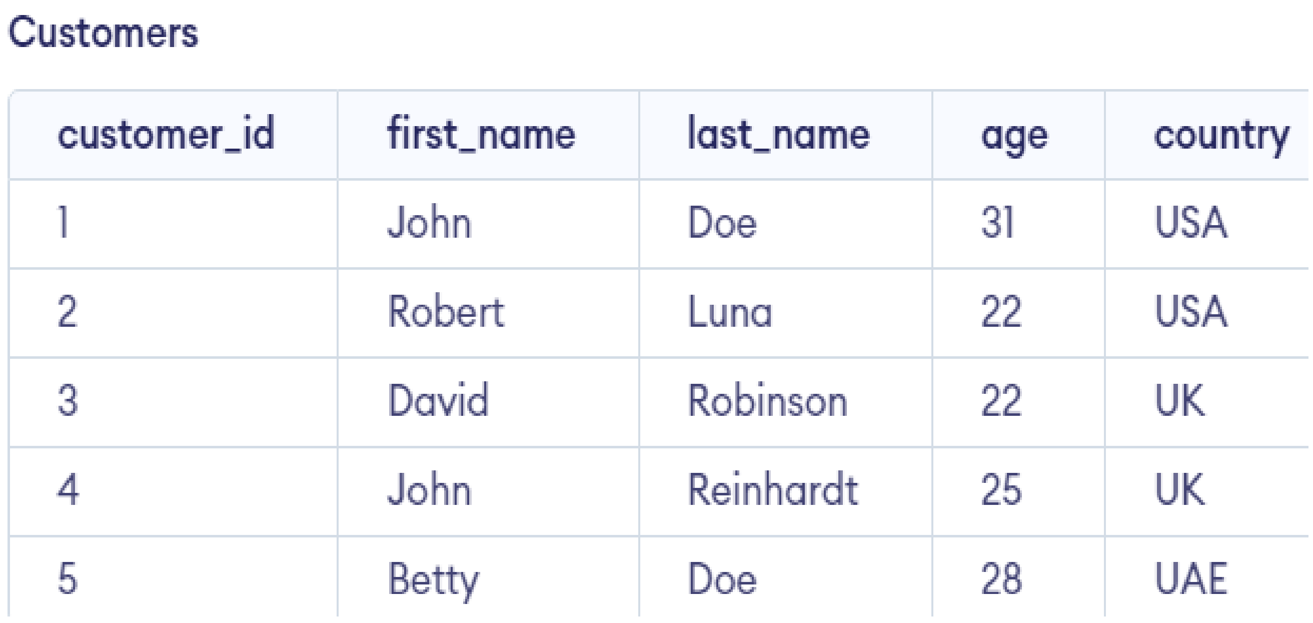
```
SELECT col1, col2 FROM table_name;
```

To Select ALL

```
SELECT * FROM table_name;
```

Ex:-

Table:-



Customers

customer_id	first_name	last_name	age	country
1	John	Doe	31	USA
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE

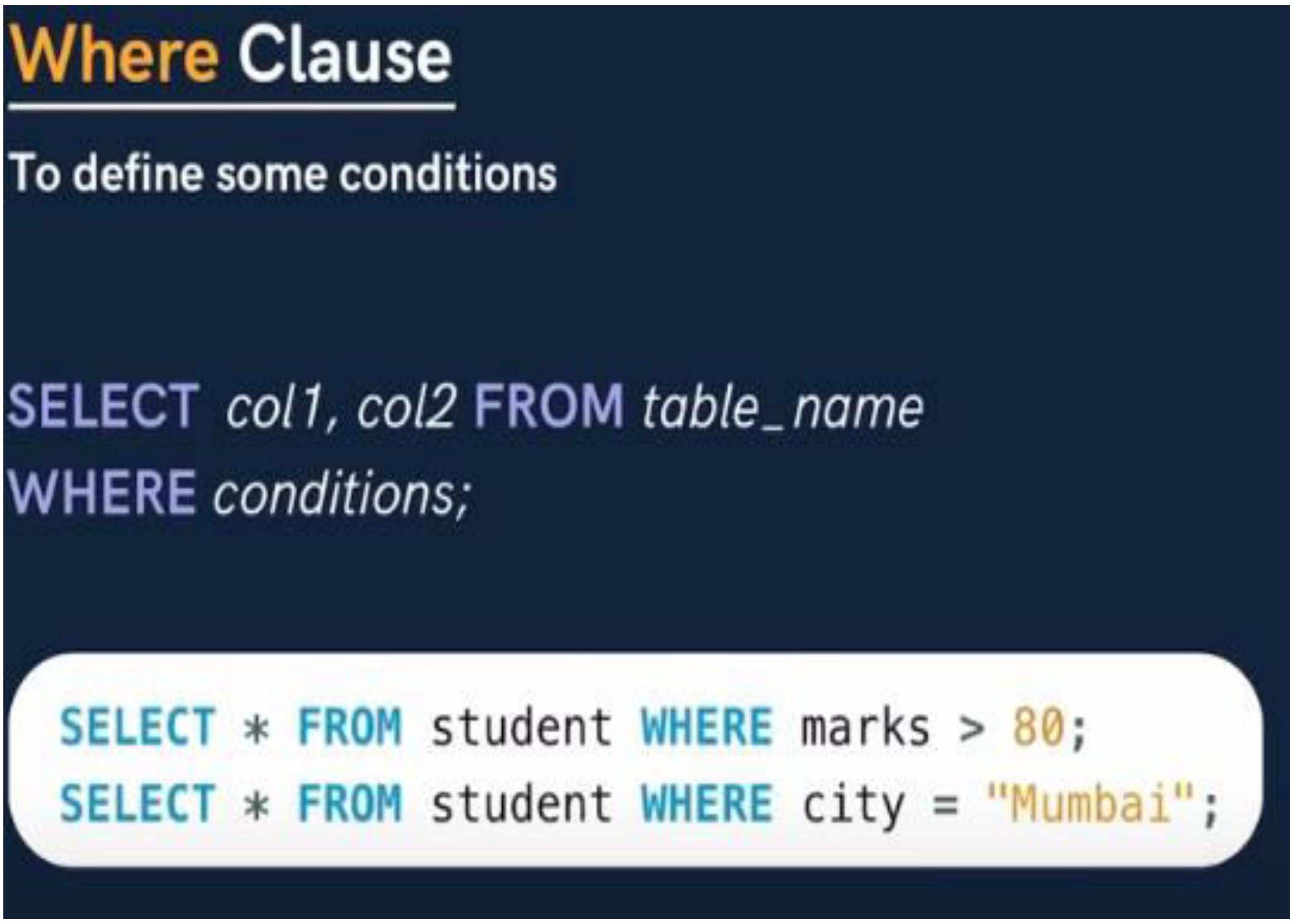
Query:-

```
SELECT first_name,last_name FROM Customers;
```

Output:-

first_name	last_name
John	Doe
Robert	Luna
David	Robinson
John	Reinhardt
Betty	Doe

Now we will see about Where Clause:-



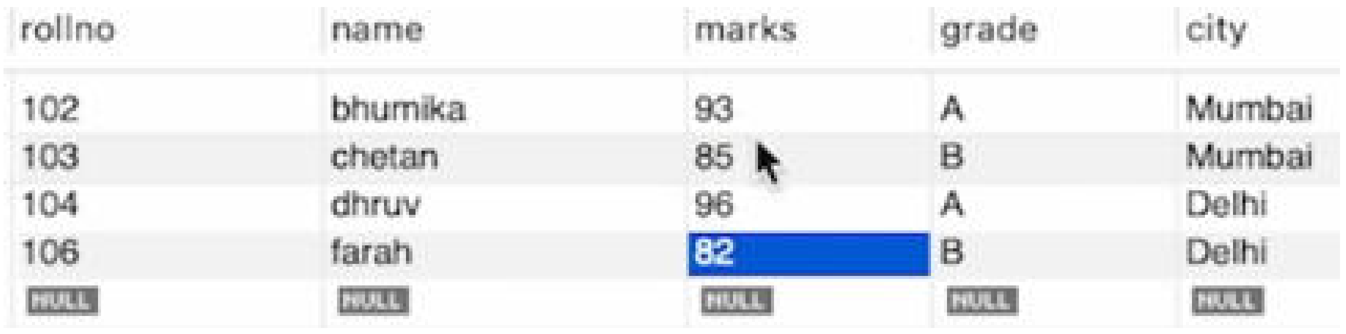
Where Clause

To define some conditions

```
SELECT col1, col2 FROM table_name
WHERE conditions;
```

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

It will select the row of that particular column like here are marks and city, whose values are greater than 80 and the city name is Mumbai respectively.

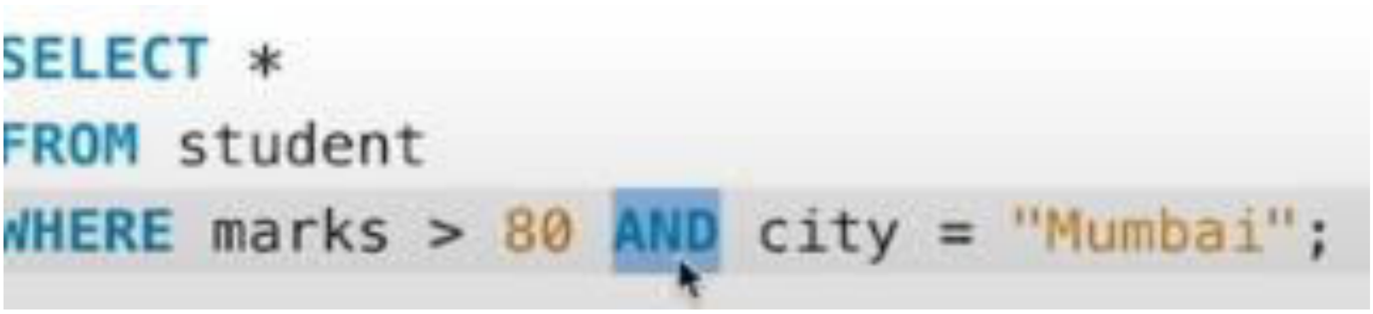


rollno	name	marks	grade	city
102	bhumika	93	A	Mumbai
103	chetan	85	B	Mumbai
104	dhruv	96	A	Delhi
106	farah	82	B	Delhi
NULL	NULL	NULL	NULL	NULL

Only marks>80 is applied.

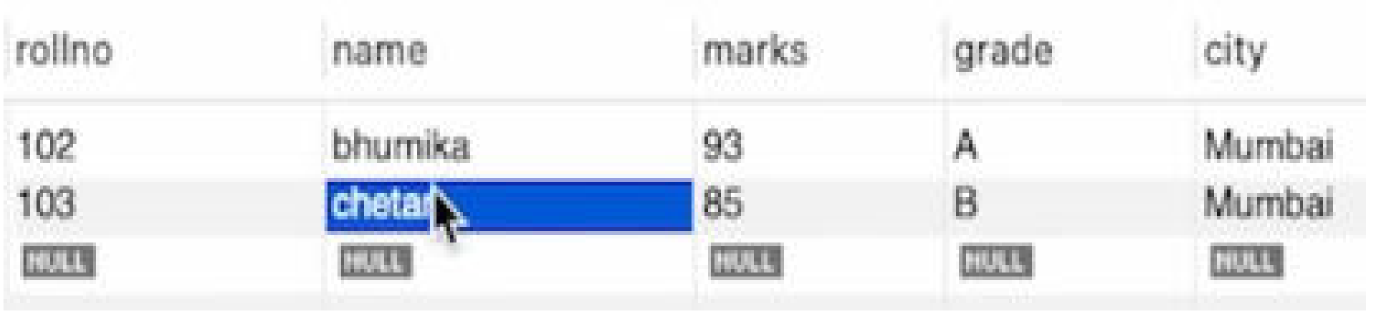
As you can see you have selected (*) where marks > 80, so it shows the table accordingly.

We club these clause like :-



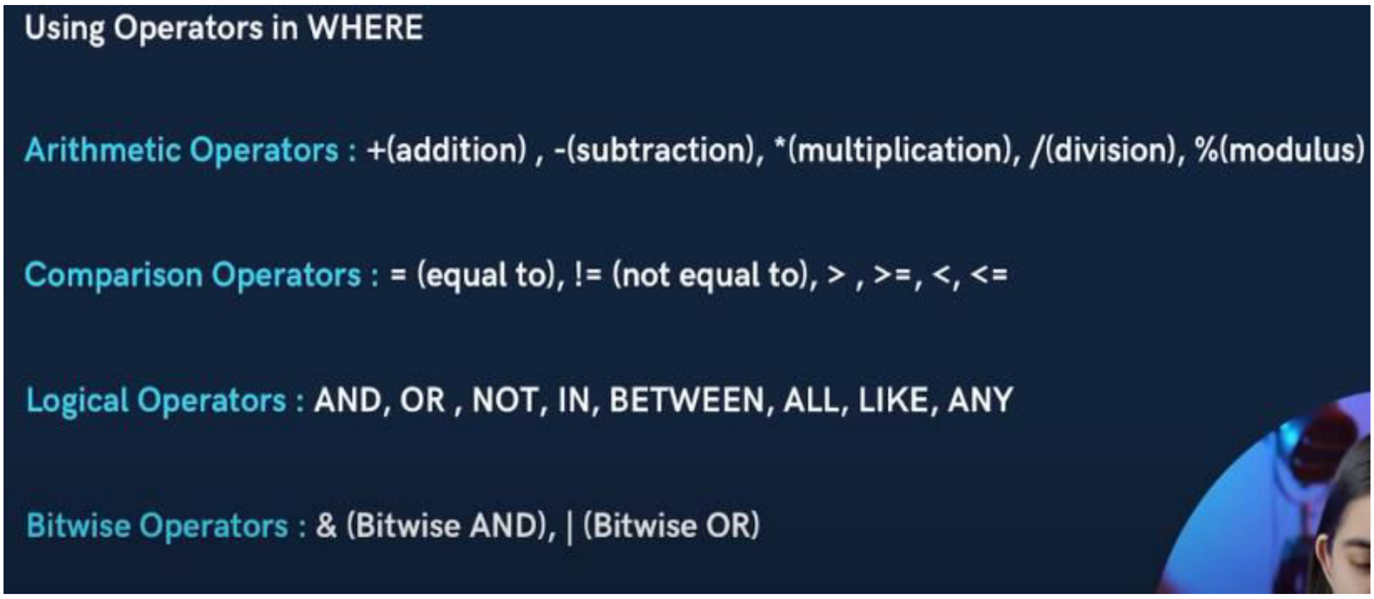
```
SELECT *
FROM student
WHERE marks > 80 AND city = "Mumbai";
```

So the table will look like this:-



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

So this AND was the logical operator here, now we will see all the operators here:-



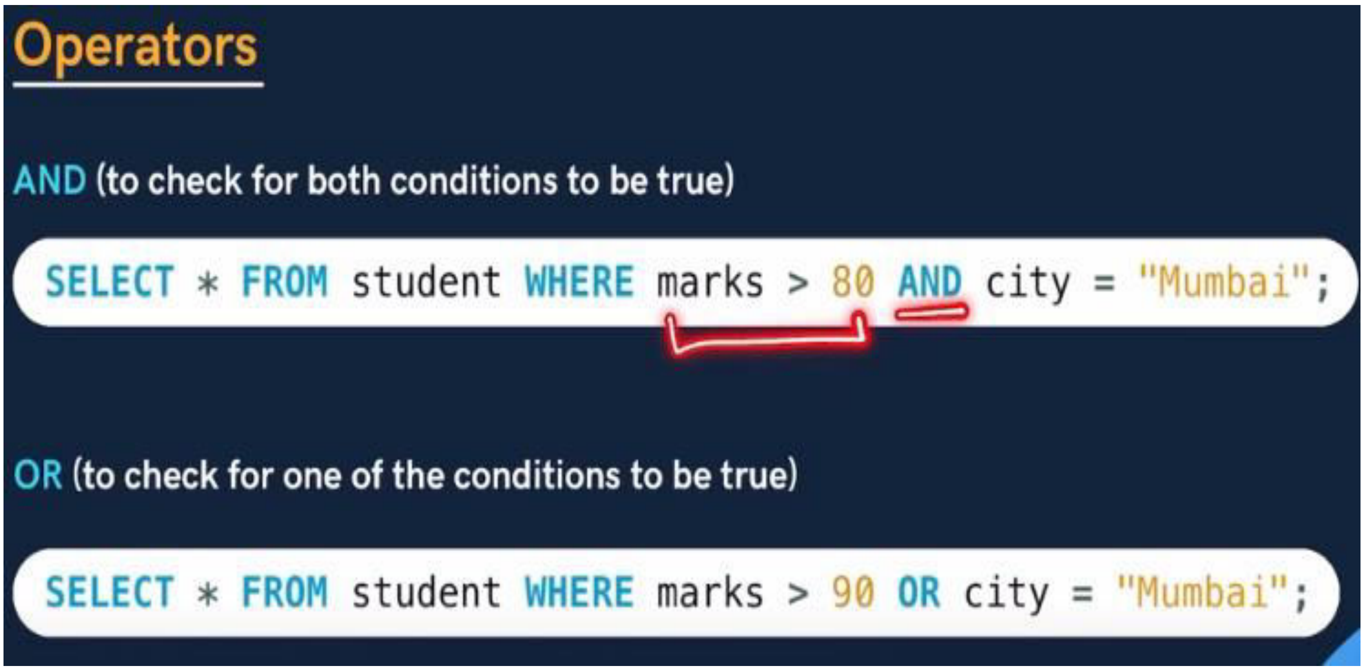
Using Operators in WHERE

Arithmetic Operators : +(addition) , -(subtraction), *(multiplication), /(division), %(modulus)

Comparison Operators : = (equal to), != (not equal to), > , >=, <, <=

Logical Operators : AND, OR , NOT, IN, BETWEEN, ALL, LIKE, ANY

Bitwise Operators : & (Bitwise AND), | (Bitwise OR)



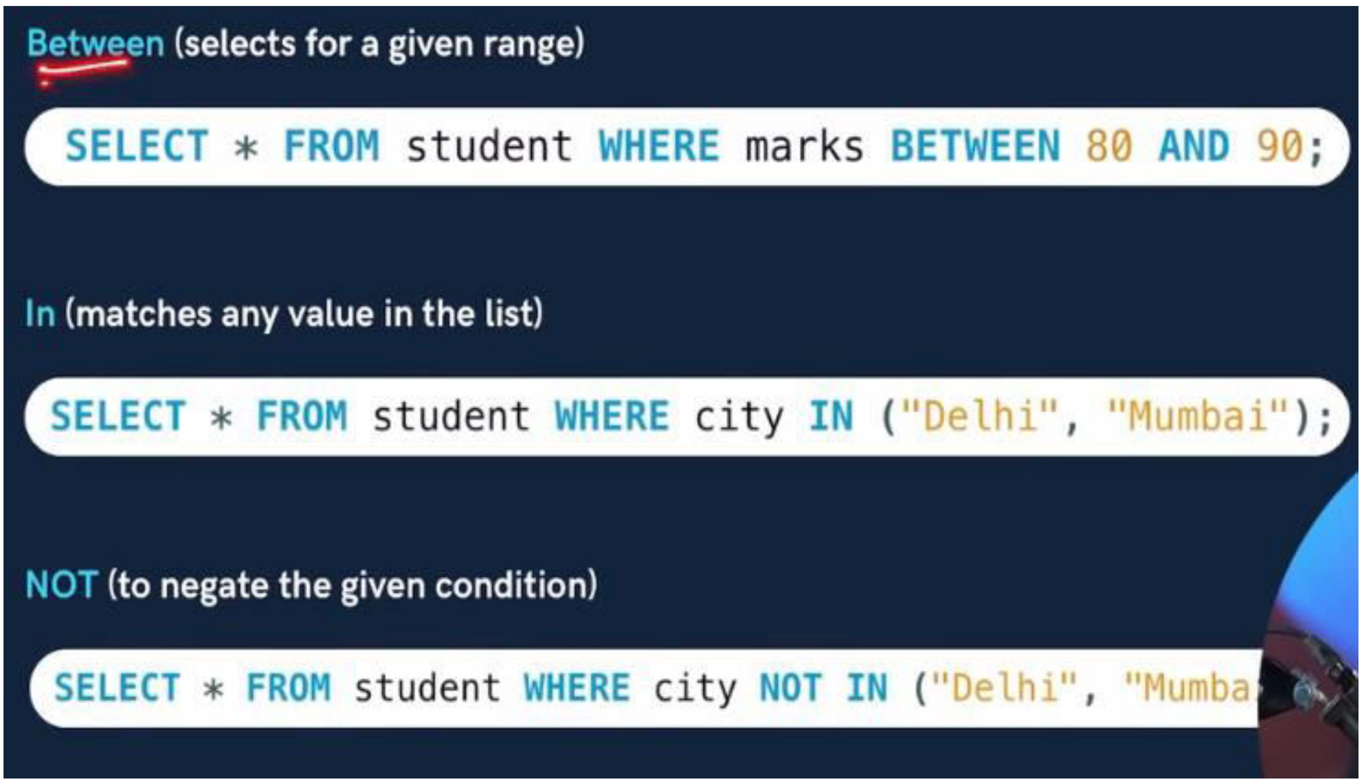
Operators

AND (to check for both conditions to be true)

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

OR (to check for one of the conditions to be true)

```
SELECT * FROM student WHERE marks > 90 OR city = "Mumbai";
```



Between (selects for a given range)

```
SELECT * FROM student WHERE marks BETWEEN 80 AND 90;
```

In (matches any value in the list)

```
SELECT * FROM student WHERE city IN ("Delhi", "Mumbai");
```

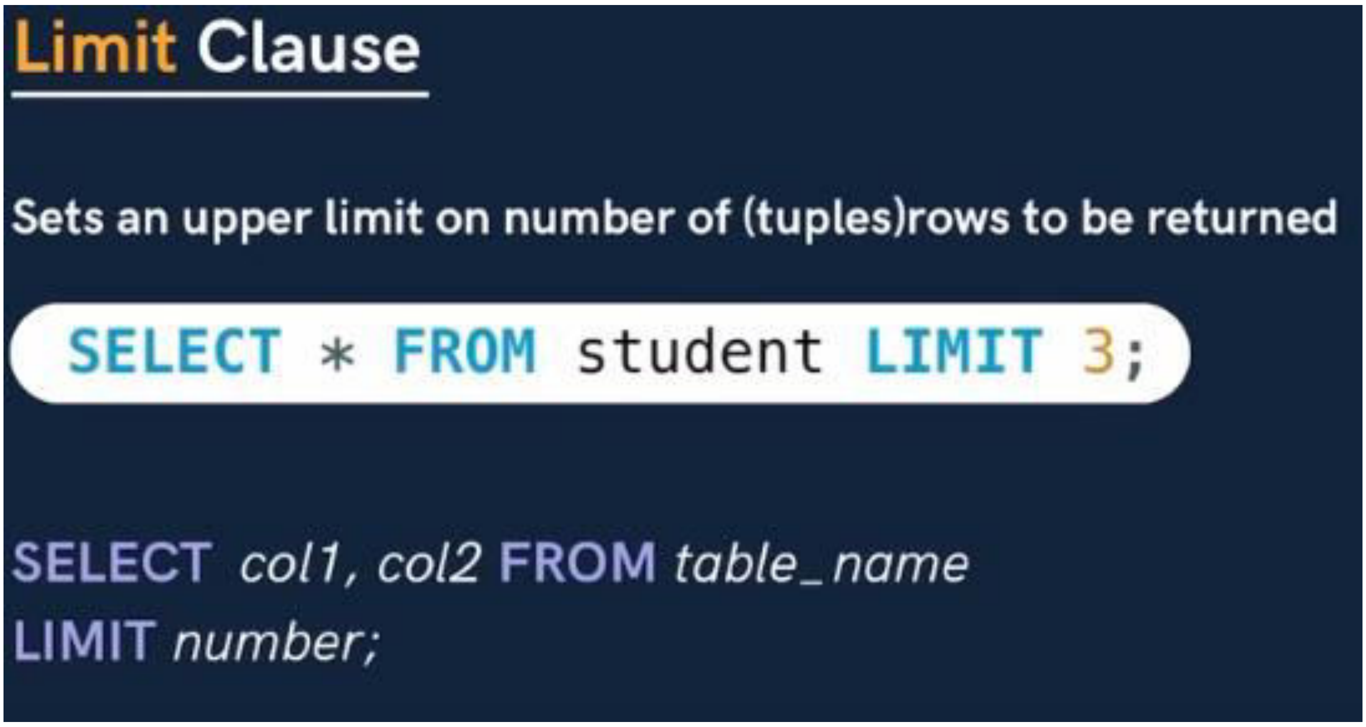
NOT (to negate the given condition)

```
SELECT * FROM student WHERE city NOT IN ("Delhi", "Mumba
```

In between, 80 and 90 are inclusive.

The IN responds with only those tuples whose city columns are mentioned inside the IN (“a”,…”b”) code.

We also limit the no. of rows to be returned as a response, by using limit clause:-



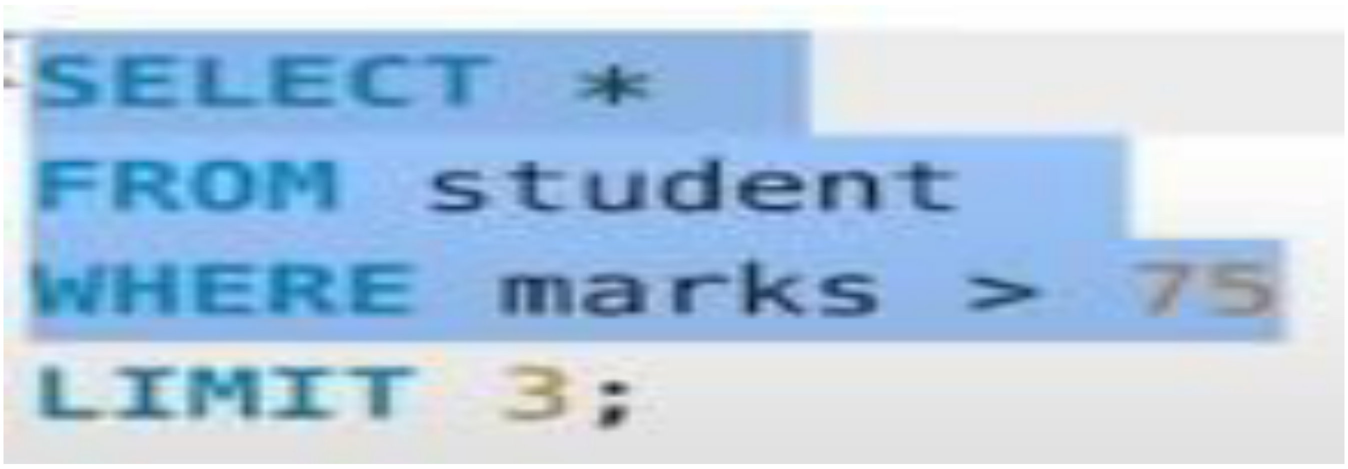
Limit Clause

Sets an upper limit on number of (tuples)rows to be returned

```
SELECT * FROM student LIMIT 3;
```

SELECT *col1, col2* FROM *table_name*
LIMIT *number;*

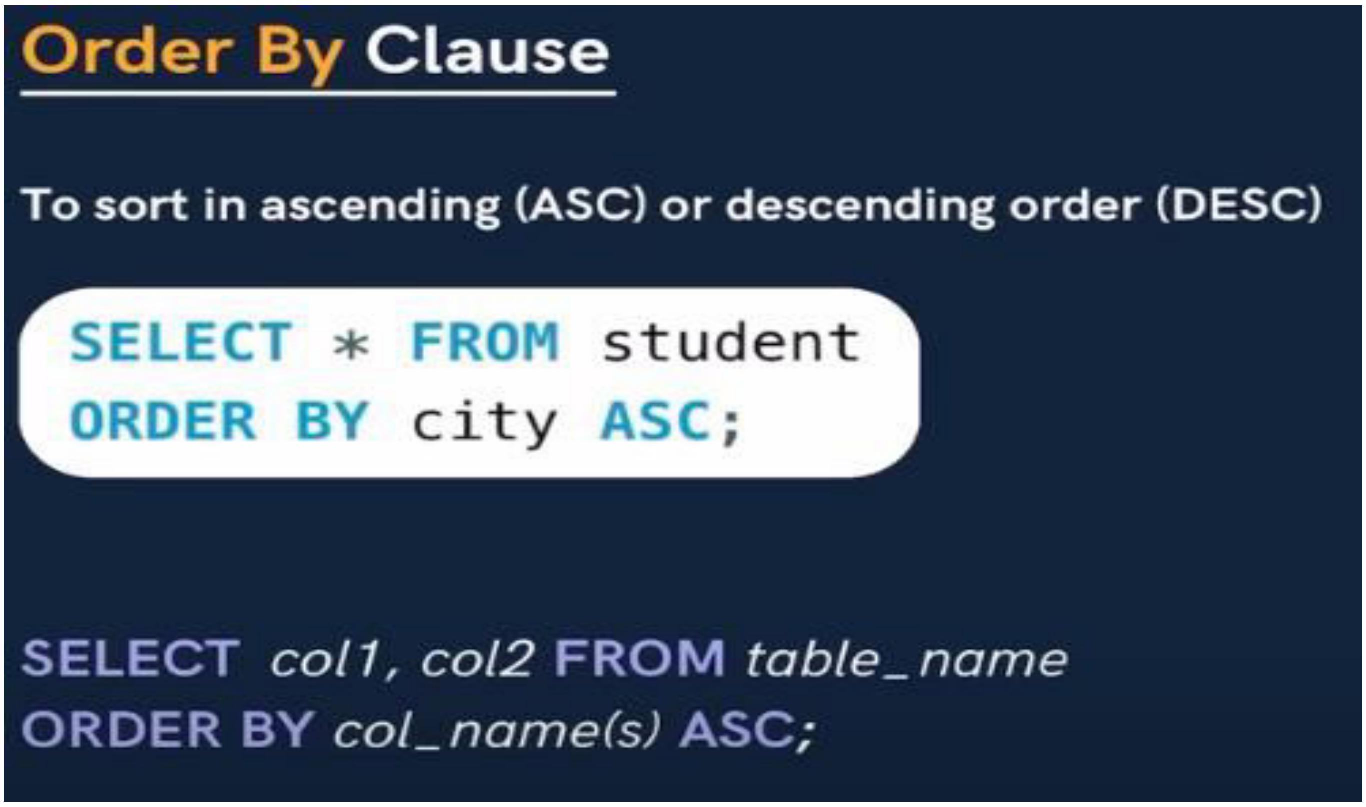
Ex:-



```
SELECT *
FROM student
WHERE marks > 75
LIMIT 3;
```

If there are 5 tuples with marks >72, the response will contain only the top 3 tuples.

We can also sort our tuples based upon any column by using order by clause:-



Order By Clause

To sort in ascending (ASC) or descending order (DESC)

```sql
SELECT * FROM student
ORDER BY city ASC;
```

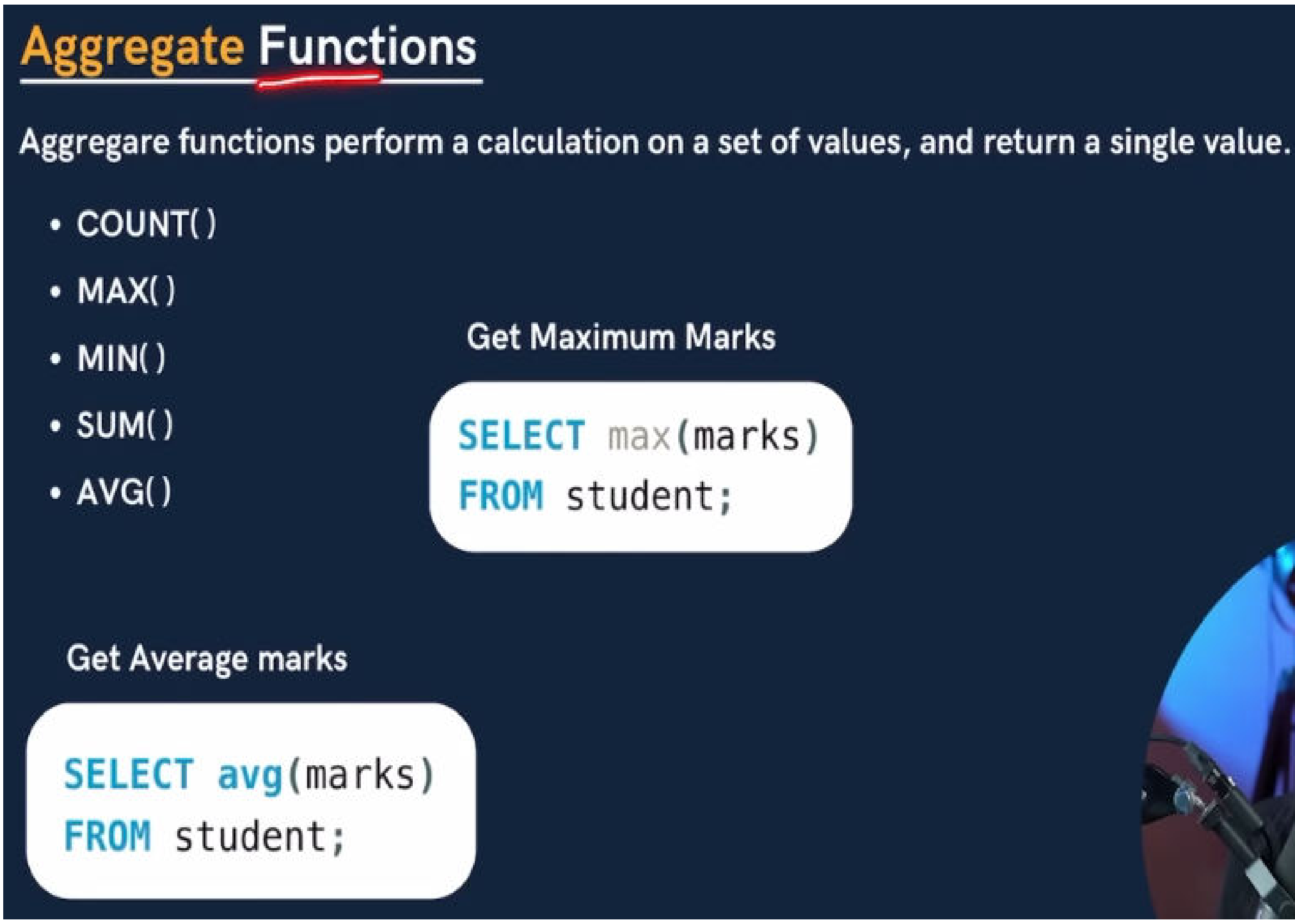
```sql
SELECT col1, col2 FROM table_name
ORDER BY col_name(s) ASC;
```

Ex:-

```sql
select * FROM customers
ORDER BY age;
```

customer_id	first_name	last_name	age	country
2	Robert	Luna	22	USA
3	David	Robinson	22	UK
4	John	Reinhardt	25	UK
5	Betty	Doe	28	UAE
1	John	Doe	31	USA

Now we will study some aggregate functions:-



Aggregate Functions

Aggregare functions perform a calculation on a set of values, and return a single value.

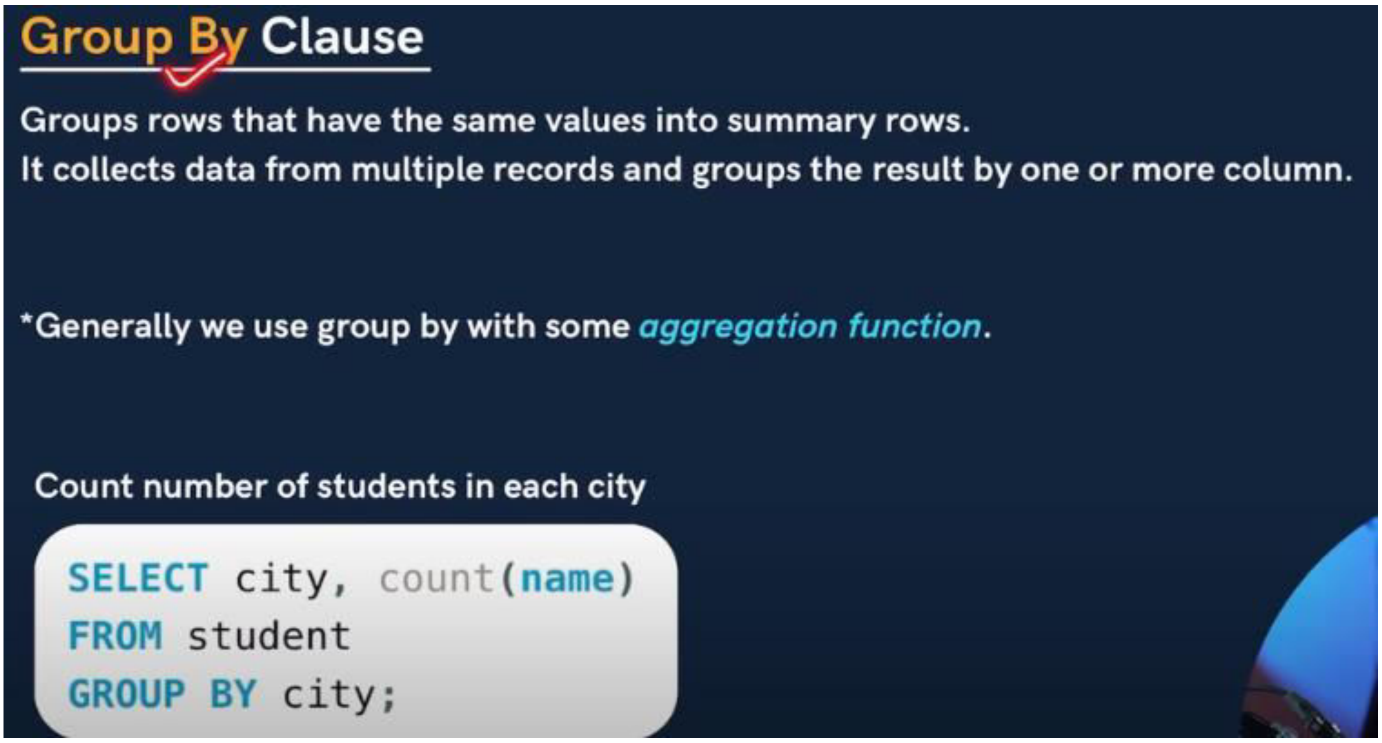
- COUNT( )
- MAX( )
- MIN( )
- SUM( )
- AVG( )

Get Maximum Marks

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

Get Average marks

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



Group By Clause

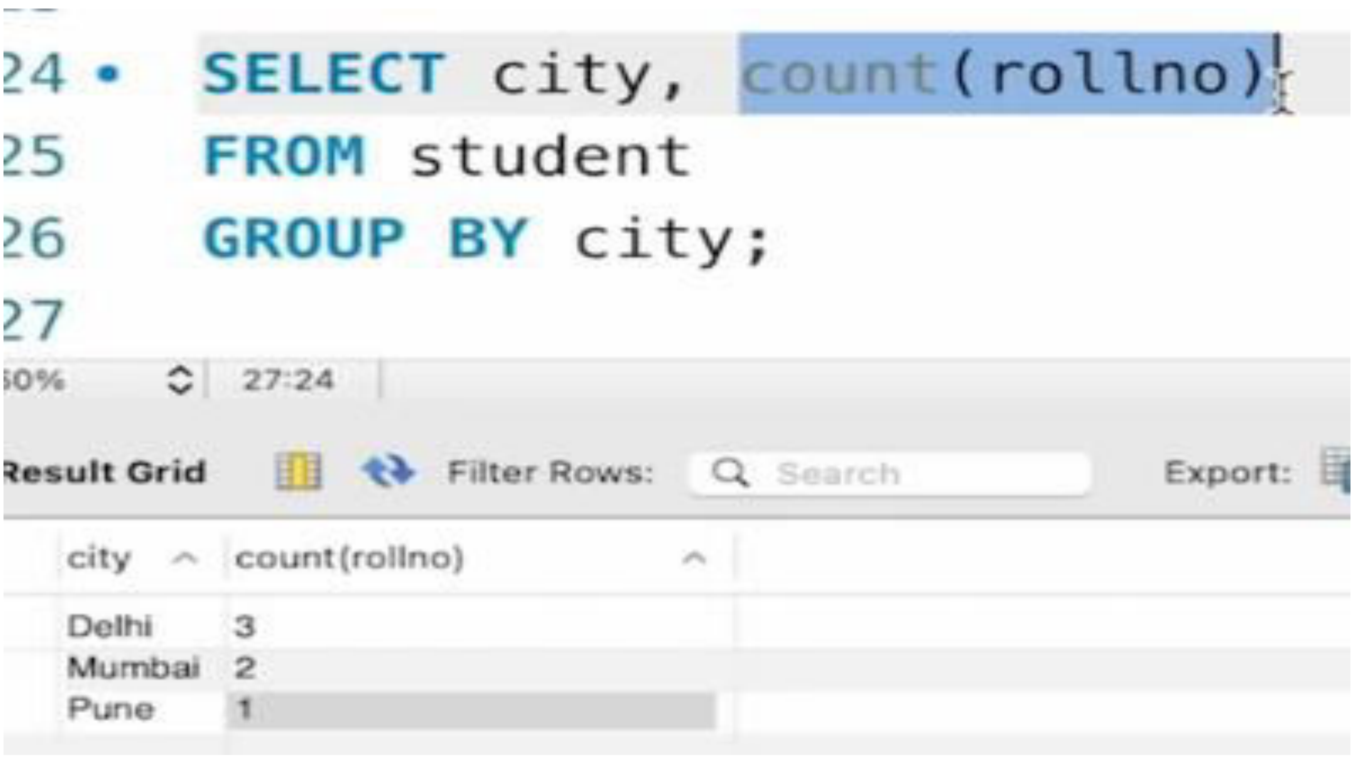
Groups rows that have the same values into summary rows.
It collects data from multiple records and groups the result by one or more column.

*Generally we use group by with some *aggregation function*.

Count number of students in each city

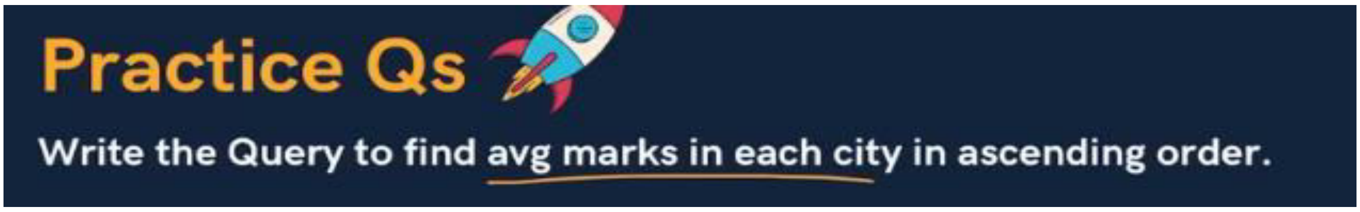
```
SELECT city, count(name)
FROM student
GROUP BY city;
```

Ex:-



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

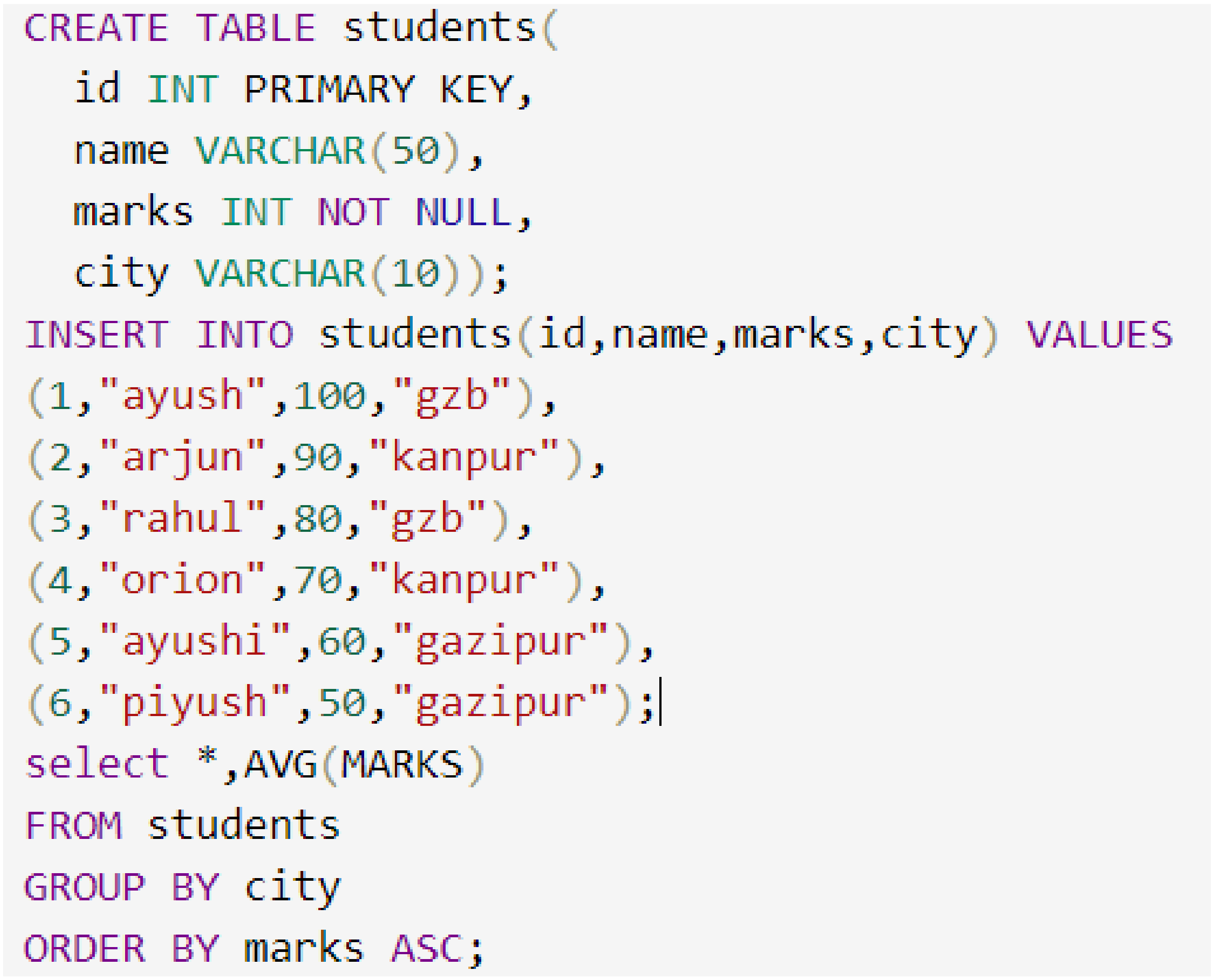
```
city     count(rollno)
Delhi    3
Mumbai   2
Pune     1
```



Practice Qs

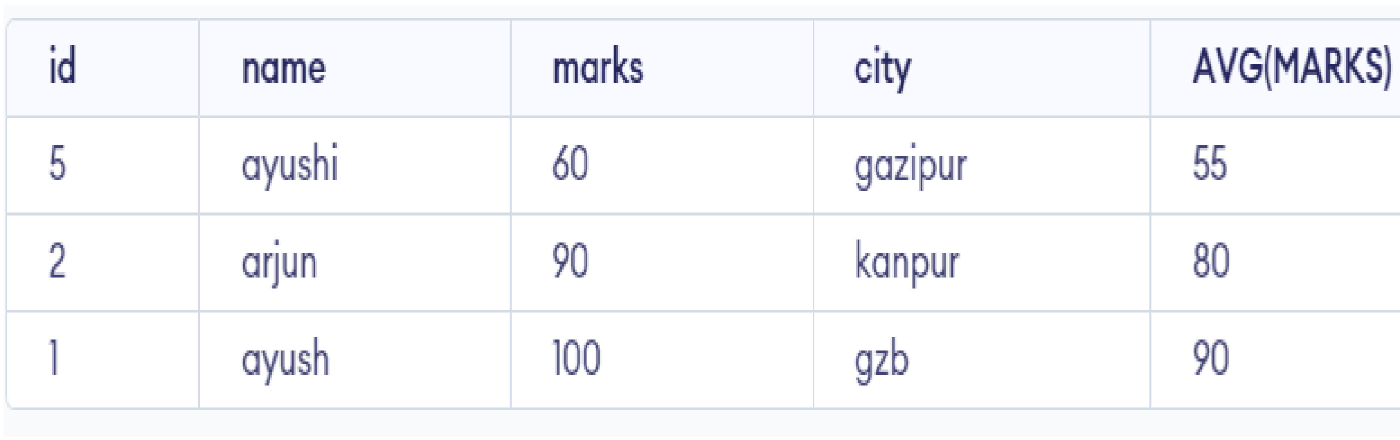
Write the Query to find avg marks in each city in ascending order.

Code:-

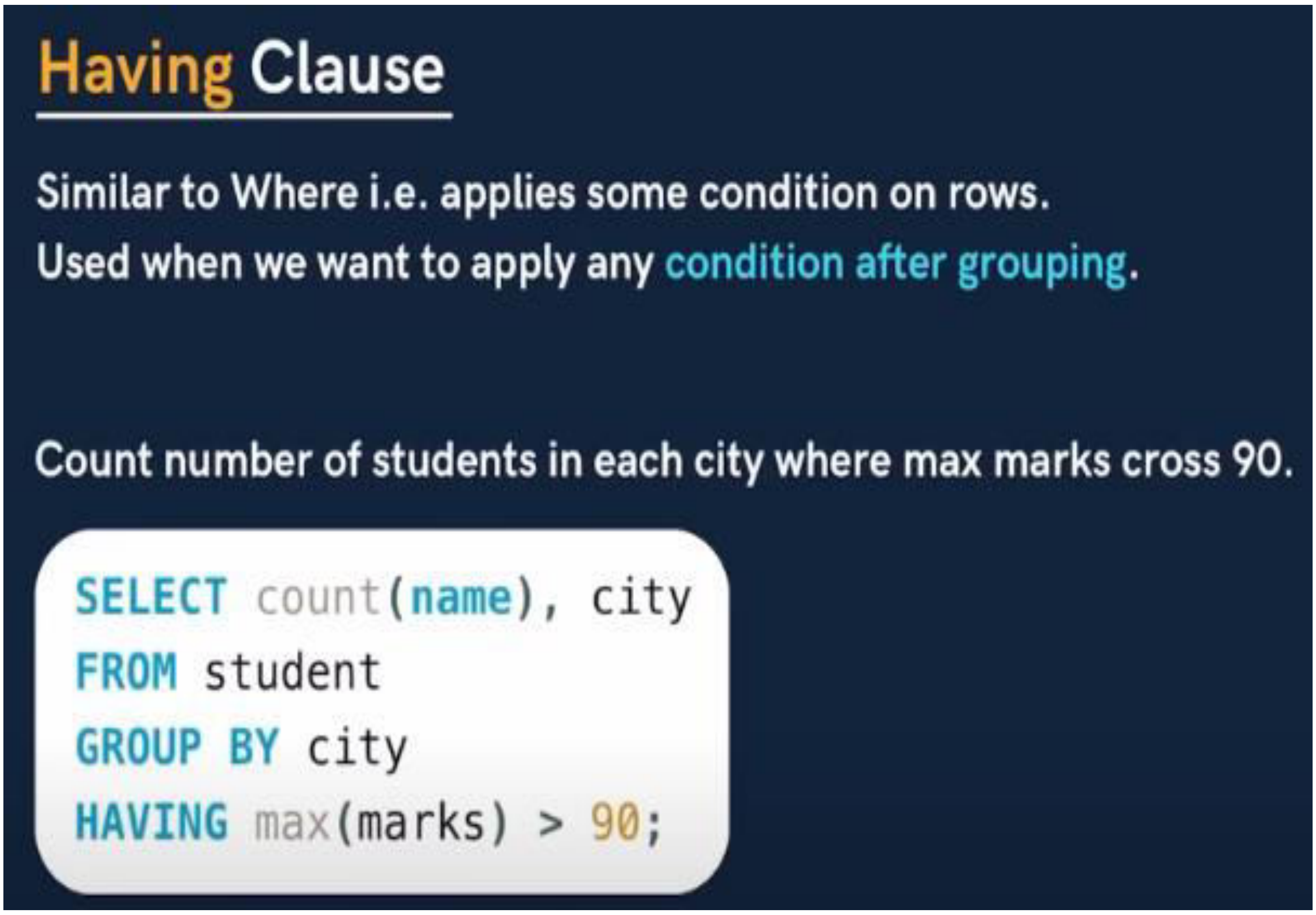


```
CREATE TABLE students(
  id INT PRIMARY KEY,
  name VARCHAR(50),
  marks INT NOT NULL,
  city VARCHAR(10));
INSERT INTO students(id,name,marks,city) VALUES
(1,"ayush",100,"gzb"),
(2,"arjun",90,"kanpur"),
(3,"rahul",80,"gzb"),
(4,"orion",70,"kanpur"),
(5,"ayushi",60,"gazipur"),
(6,"piyush",50,"gazipur");
select *,AVG(MARKS)
FROM students
GROUP BY city
ORDER BY marks ASC;
```

Output:-



id	name	marks	city	AVG(MARKS)
5	ayushi	60	gazipur	55
2	arjun	90	kanpur	80
1	ayush	100	gzb	90



Having Clause

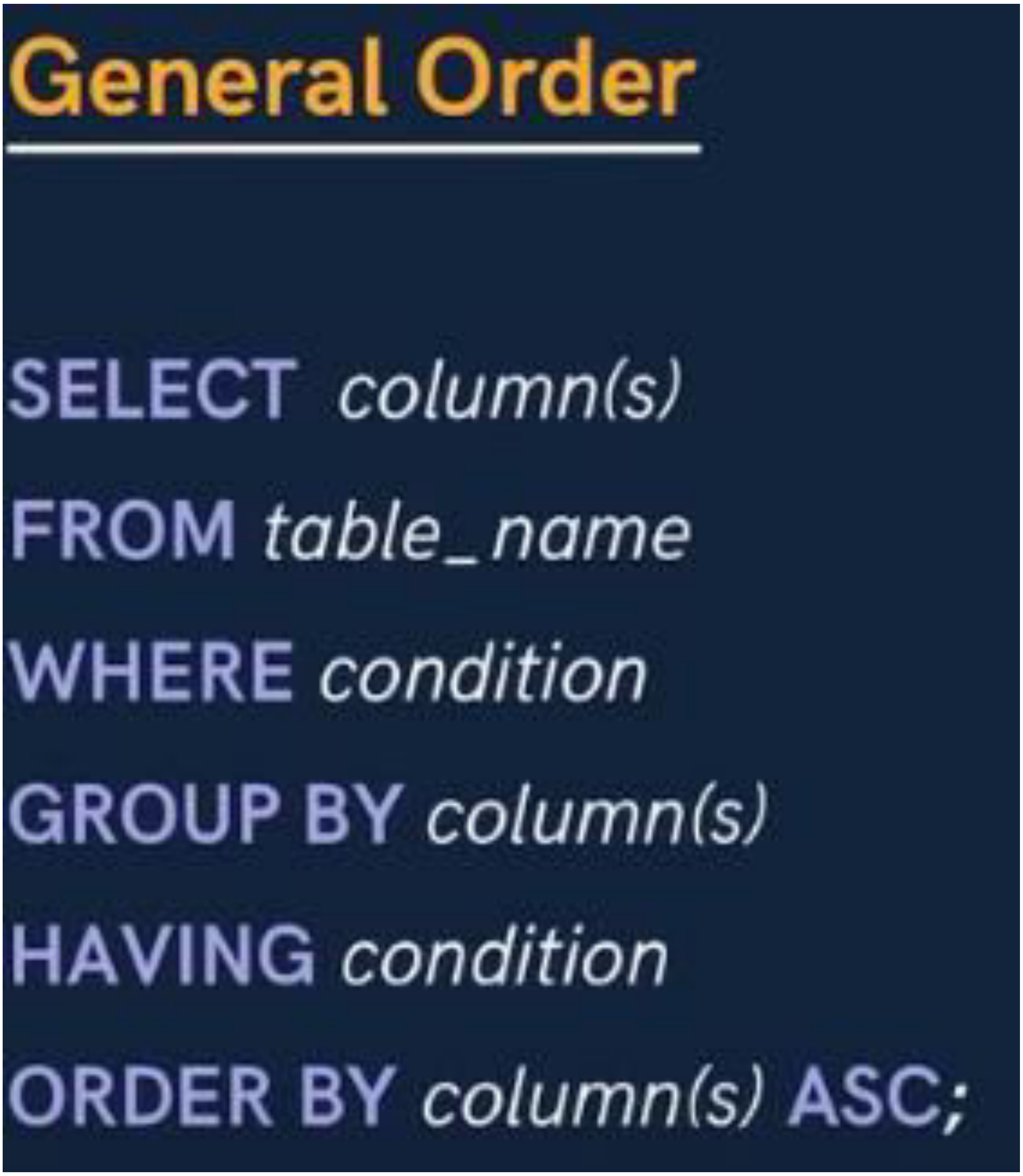
Similar to Where i.e. applies some condition on rows.
Used when we want to apply any condition after grouping.

Count number of students in each city where max marks cross 90.

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

WHERE clause is applied to a single tuple, but the HAVING clause can be applied to a group of tuples.

We should follow some order to write the clauses:-



General Order

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

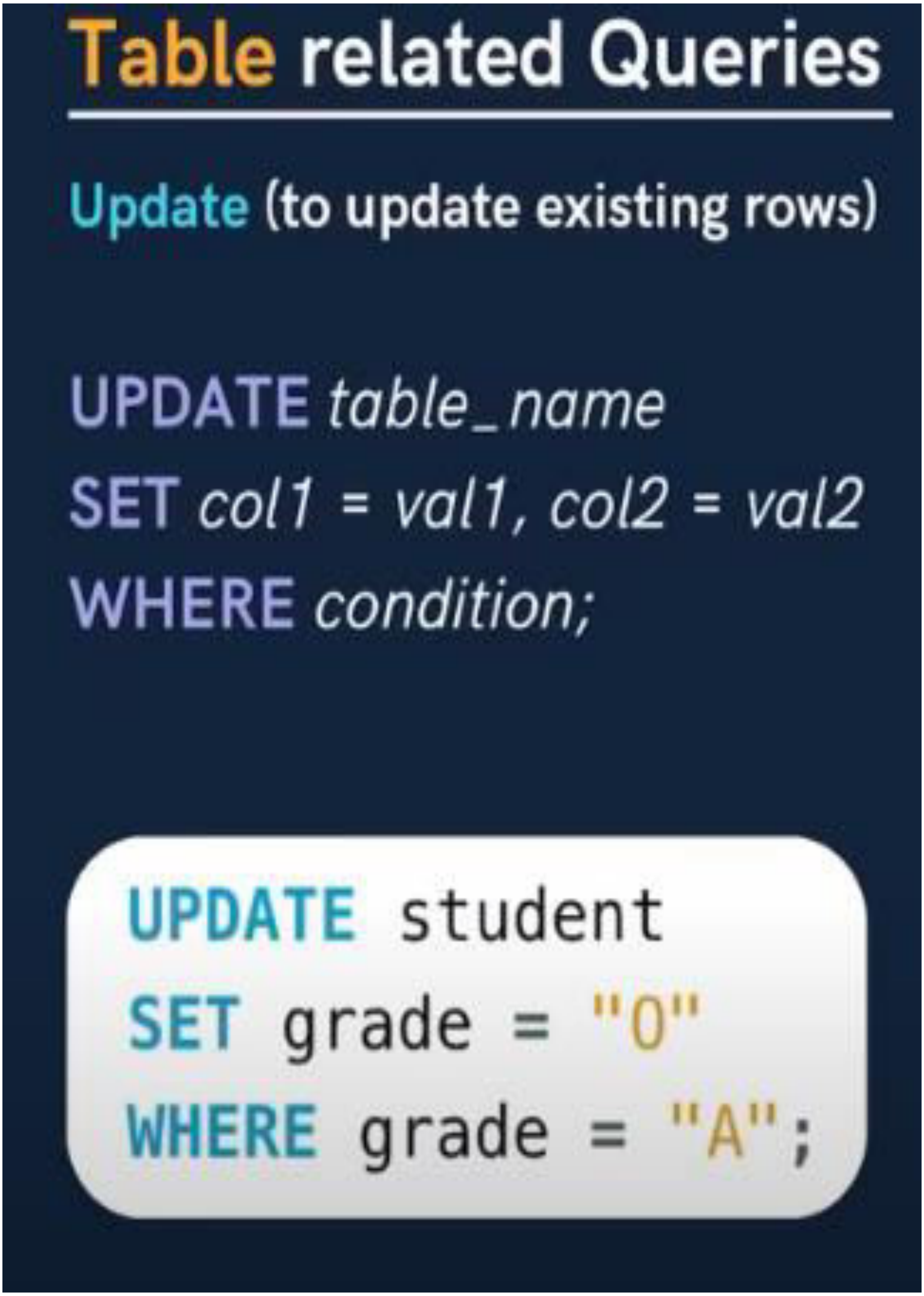
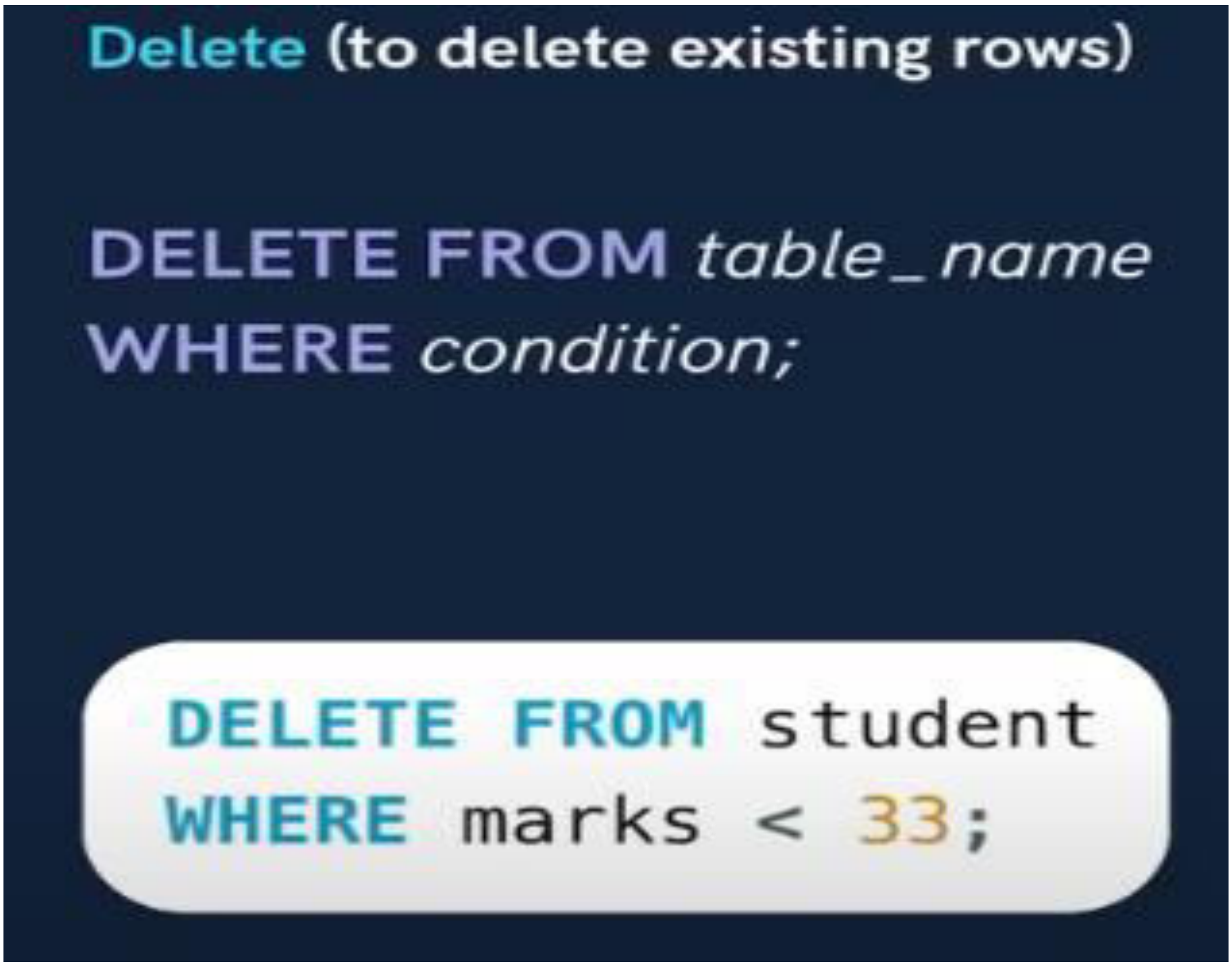


Table related Queries

Update (to update existing rows)

```
UPDATE table_name
SET col1 = val1, col2 = val2
WHERE condition;
```

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

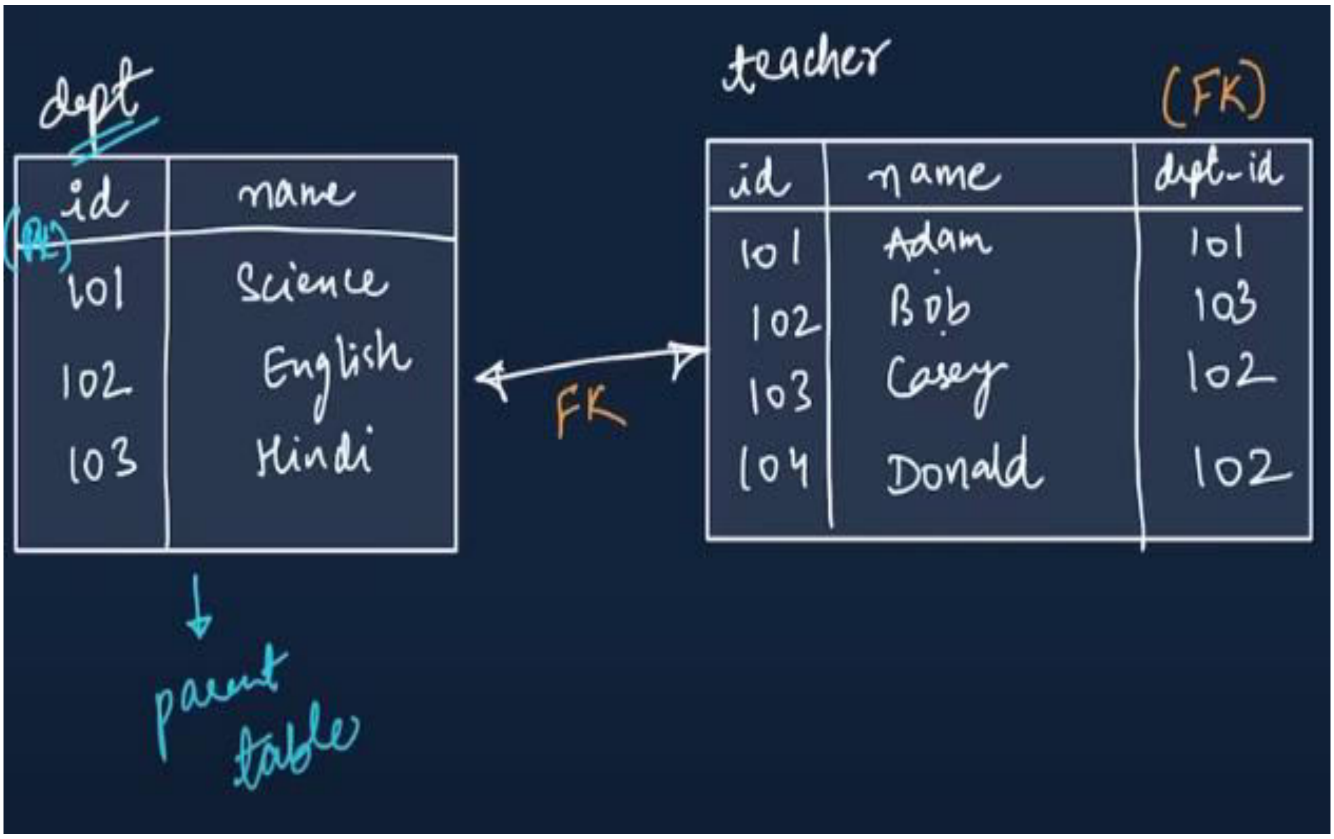


Delete (to delete existing rows)

```
DELETE FROM table_name
WHERE condition;
```

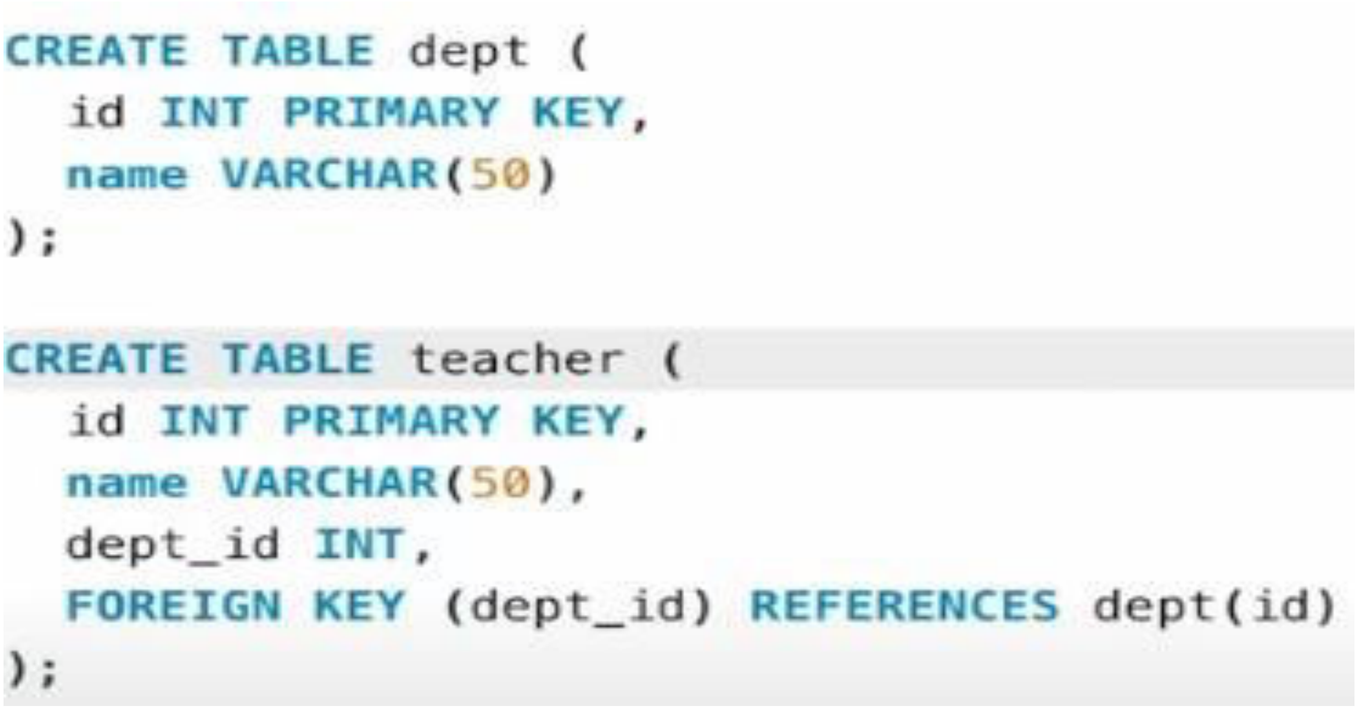
```
DELETE FROM student
WHERE marks < 33;
```

Now. We will declare relationship between two tables by using foreign key:-



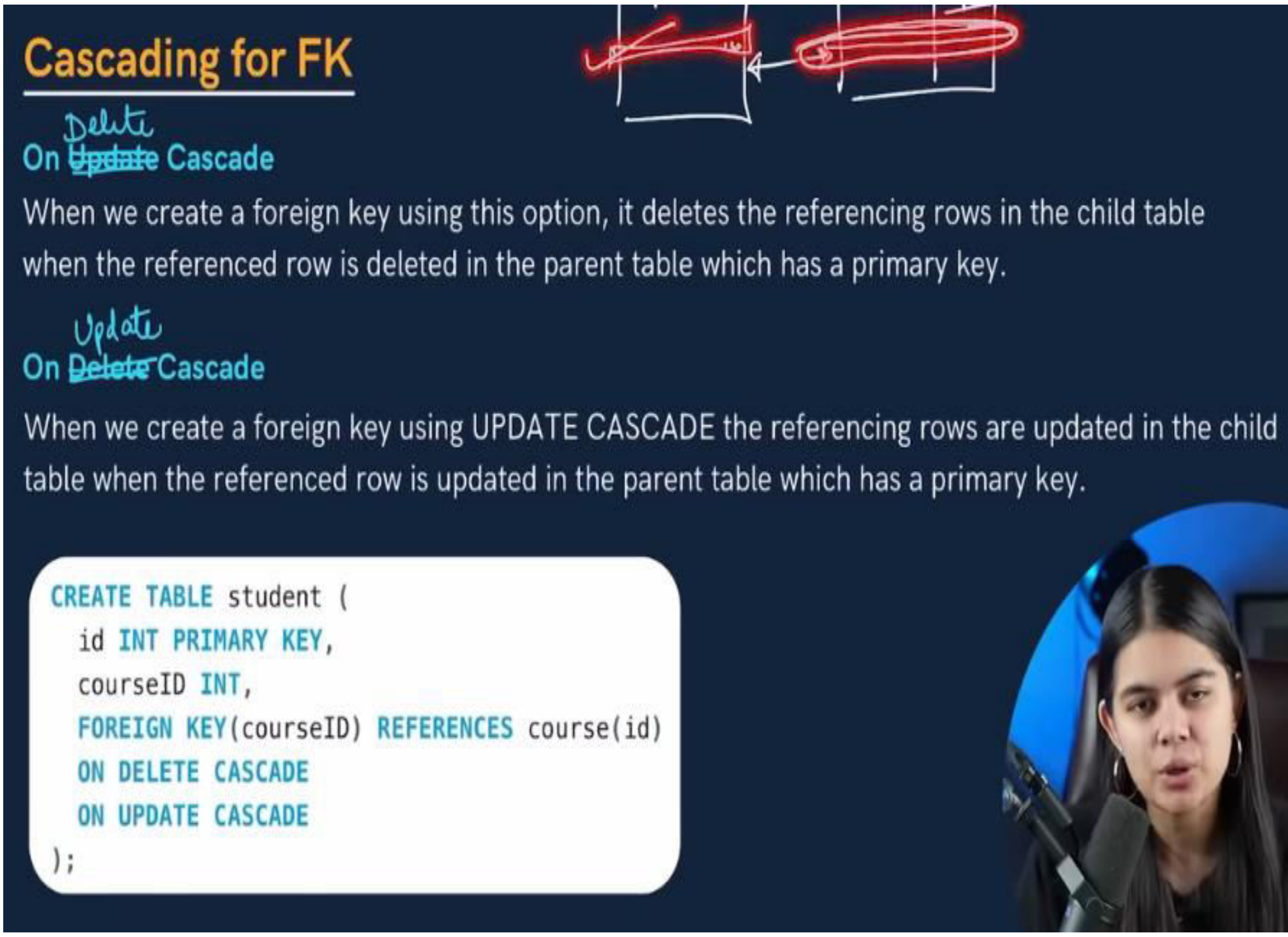
id	name
101	Science
102	English
103	Hindi

id	name	dept_id
101	Adam	101
102	Bob	103
103	Casey	102
104	Donald	102



```
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)
);
```



Cascading for FK

On ~~Update~~ Delete Cascade

When we create a foreign key using this option, it deletes the referencing rows in the child table when the referenced row is deleted in the parent table which has a primary key.

On ~~Delete~~ Update Cascade

When we create a foreign key using UPDATE CASCADE the referencing rows are updated in the child table when the referenced row is updated in the parent table which has a primary key.

```
CREATE TABLE student (
  id INT PRIMARY KEY,
  courseID INT,
  FOREIGN KEY(courseID) REFERENCES course(id)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);
```

Age parent table(jisme primary key jo kisi aur table ka foreign key h) ,me kuch change hua to us table me bhi change hoga jhn pe foreign key h.

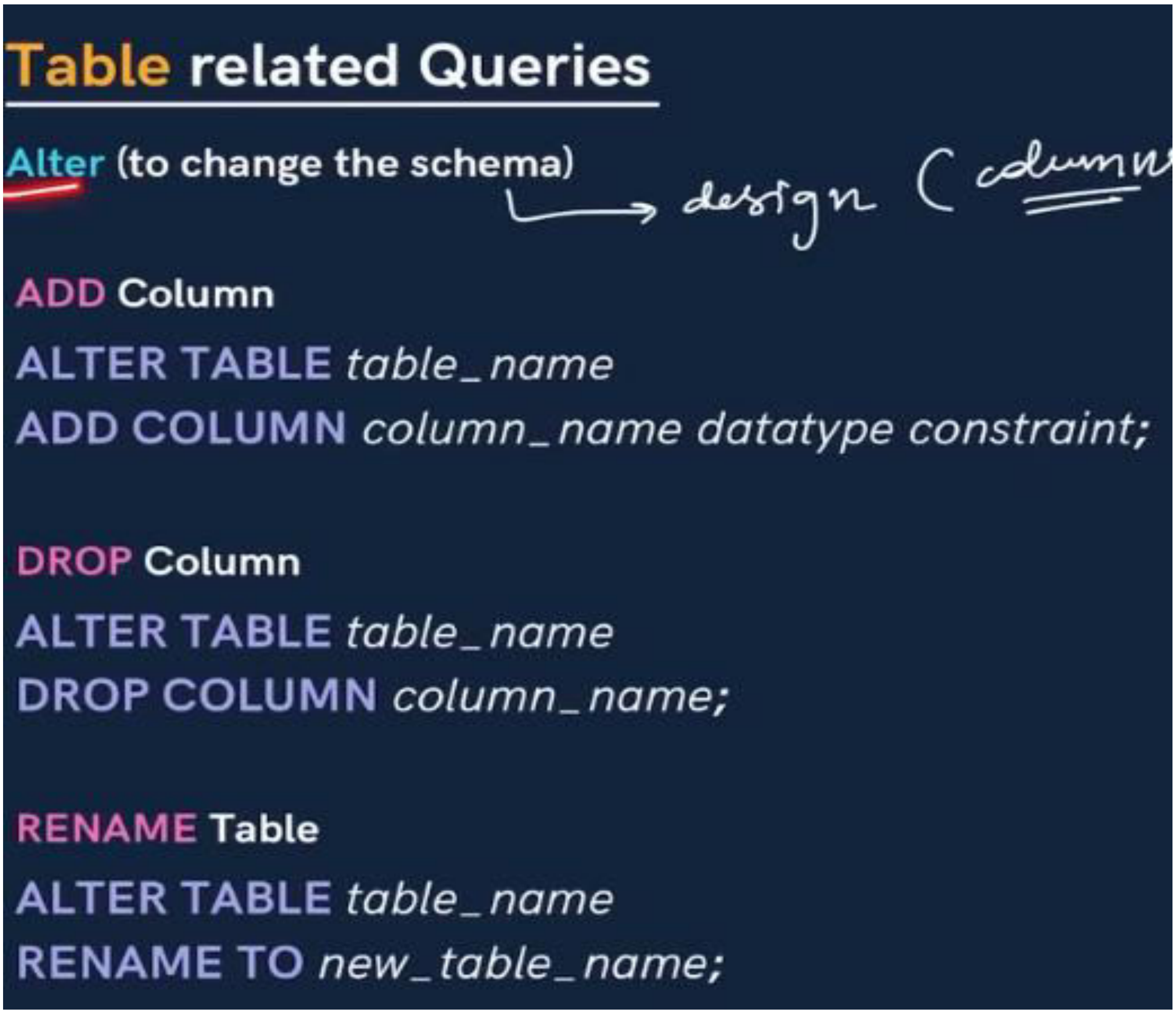


Table related Queries

Alter (to change the schema) $\longrightarrow$ design (column

ADD Column

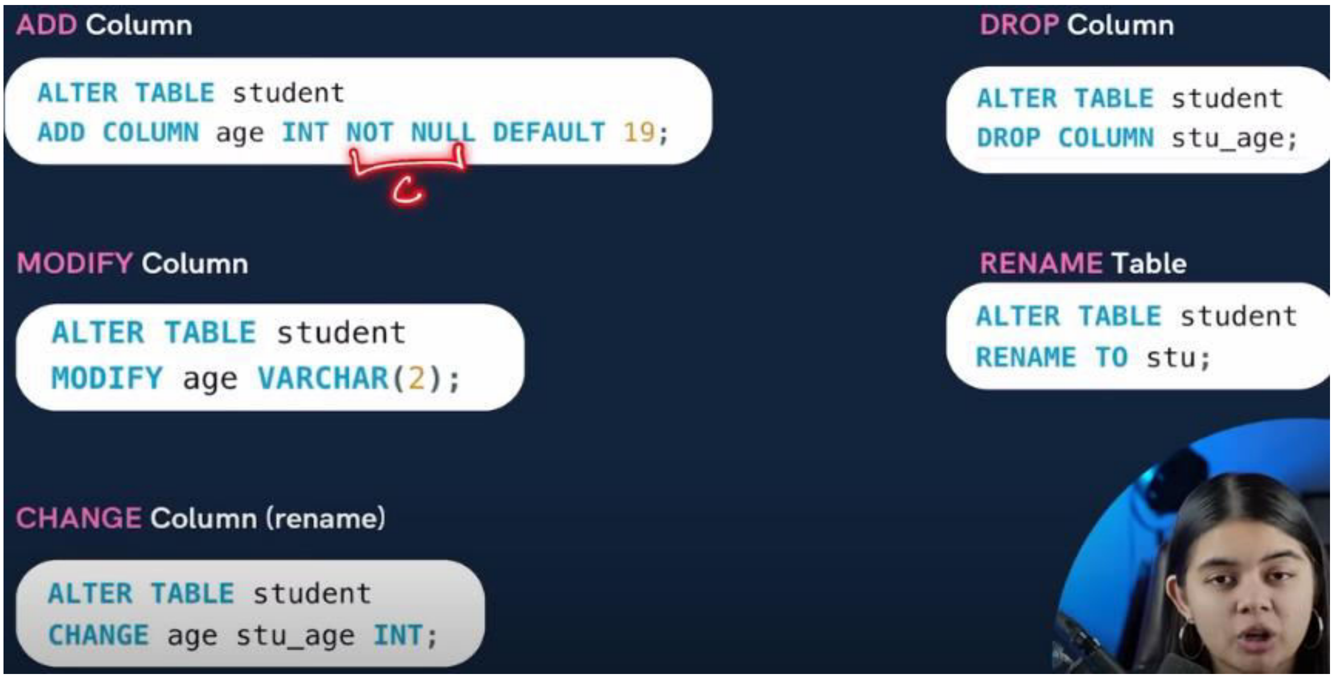
```
ALTER TABLE table_name
ADD COLUMN column_name datatype constraint;
```

DROP Column

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

RENAME Table

```
ALTER TABLE table_name
RENAME TO new_table_name;
```



ADD Column

```
ALTER TABLE student
ADD COLUMN age INT NOT NULL DEFAULT 19;
                      C
```

DROP Column

```
ALTER TABLE student
DROP COLUMN stu_age;
```

MODIFY Column

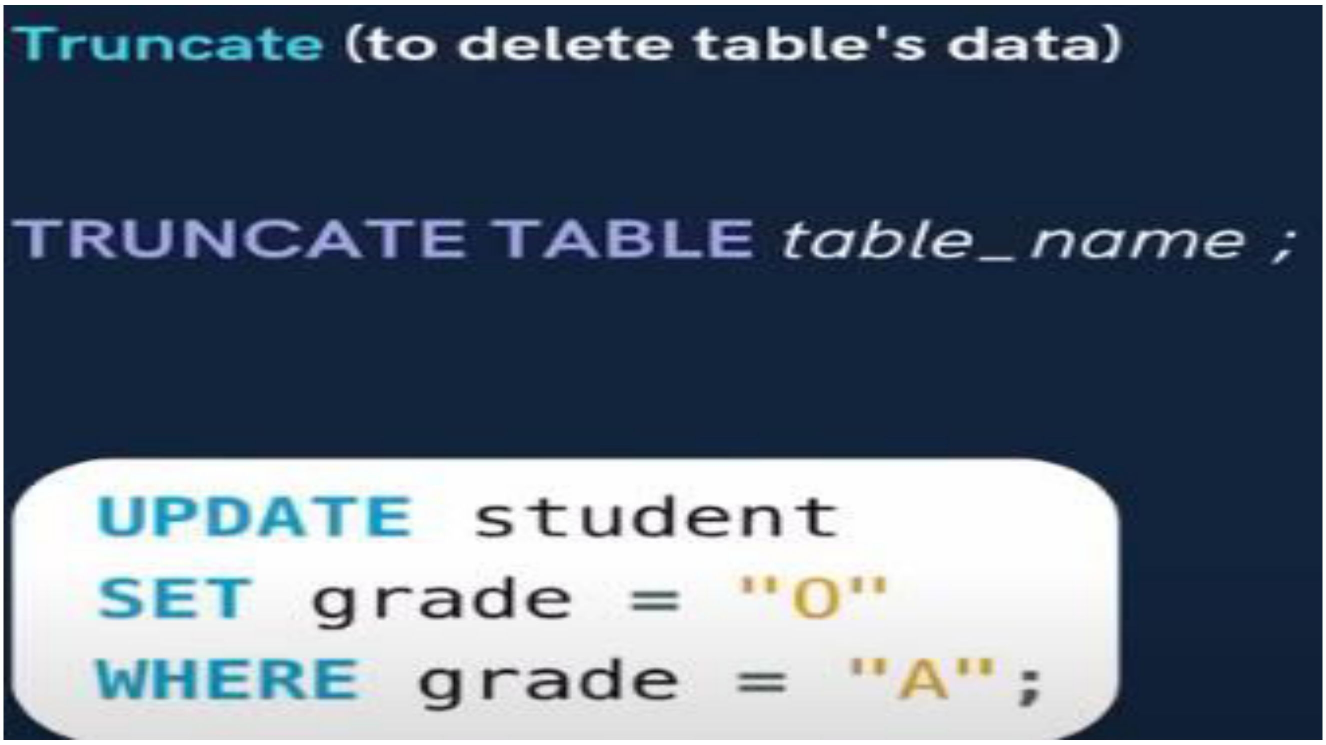
```
ALTER TABLE student
MODIFY age VARCHAR(2);
```

RENAME Table

```
ALTER TABLE student
RENAME TO stu;
```

CHANGE Column (rename)

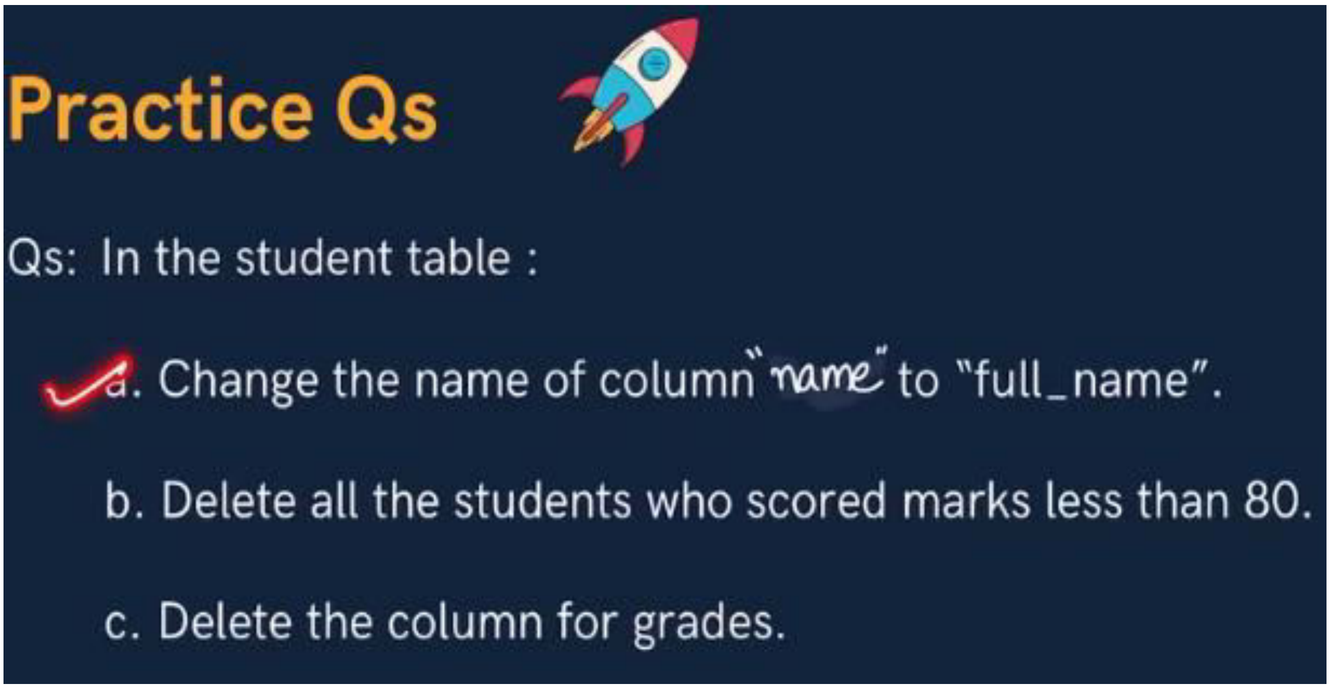
```
ALTER TABLE student
CHANGE age stu_age INT;
```


Truncate (to delete table's data)

```
TRUNCATE TABLE table_name ;
```

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

Truncate only deletes the data while drop deletes the entire table.


Practice Qs

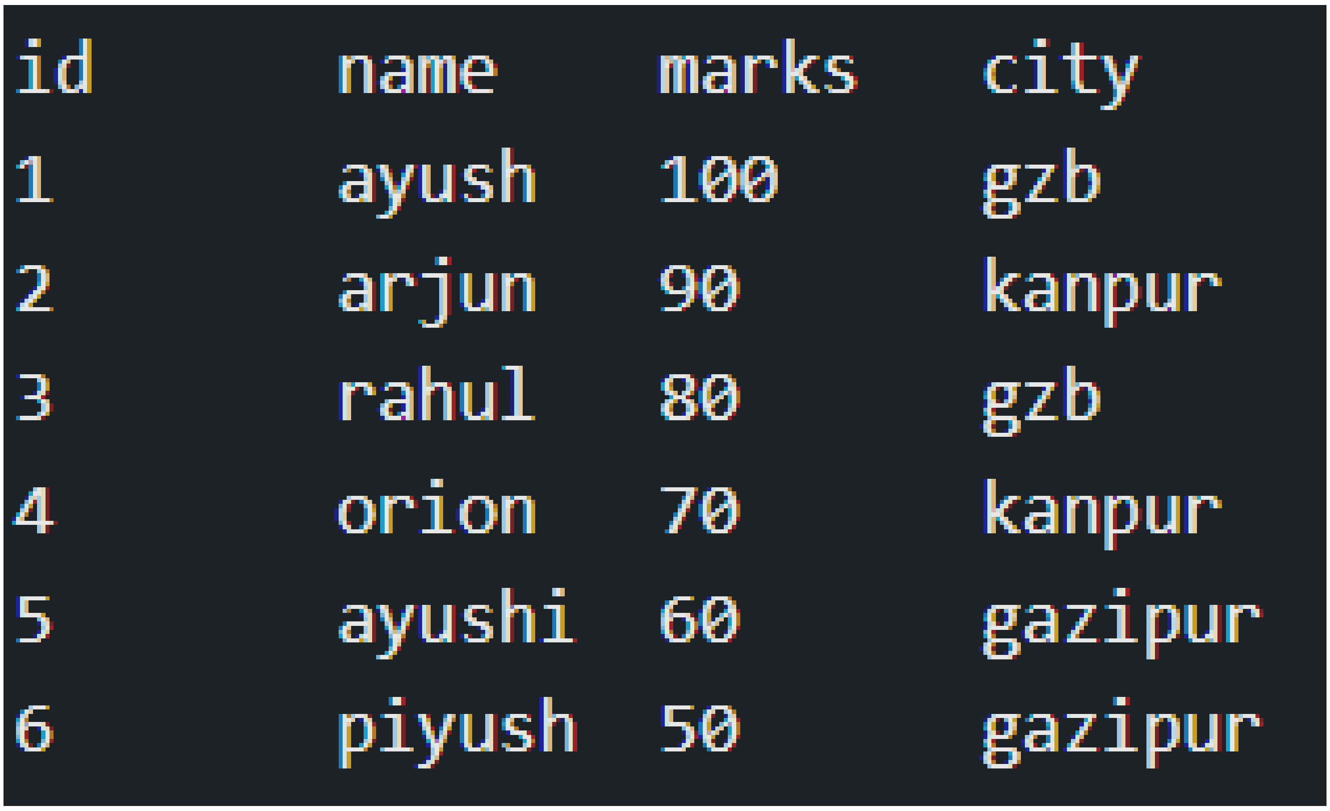
Qs: In the student table :

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

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

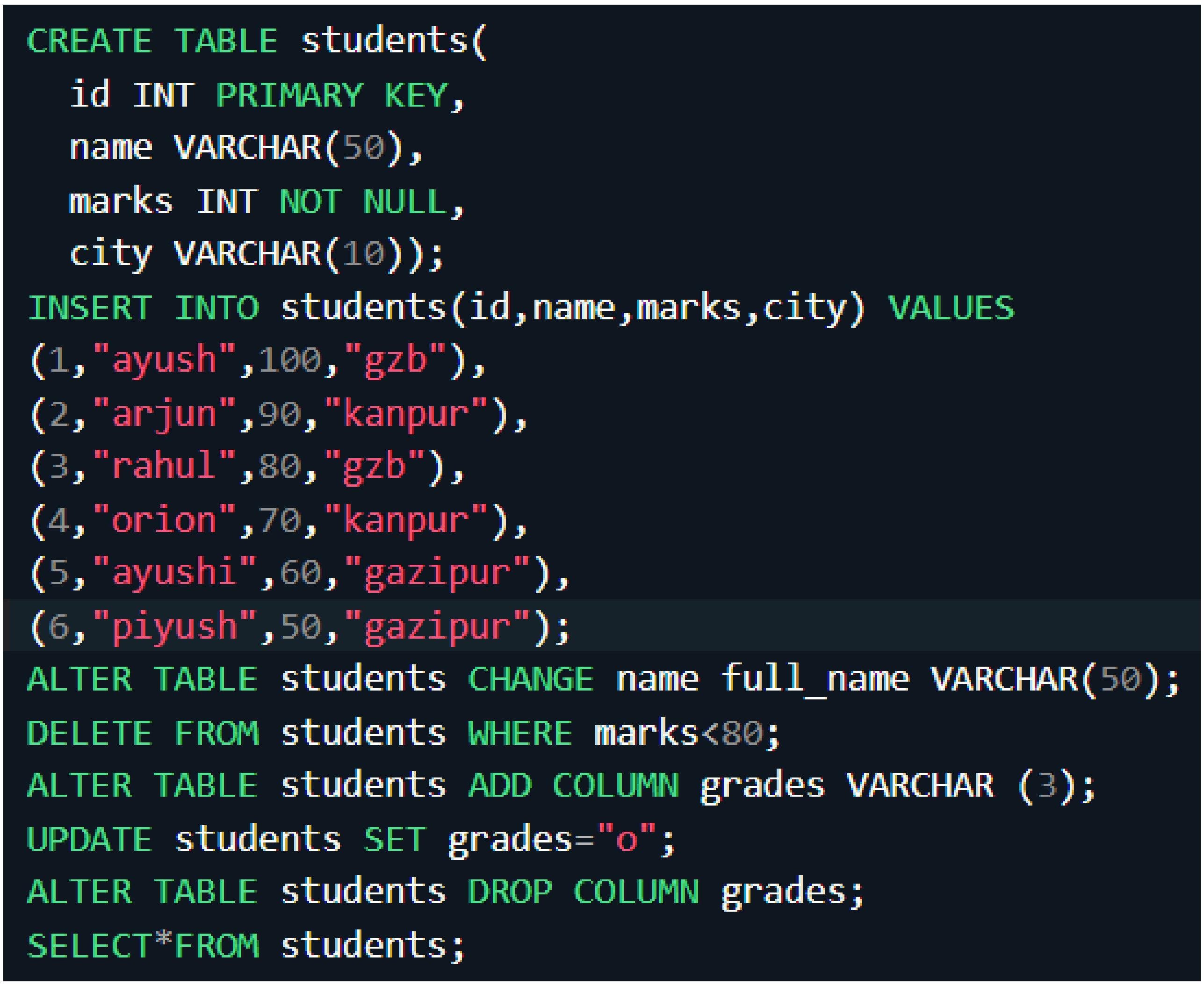
c. Delete the column for grades.

Original table:-



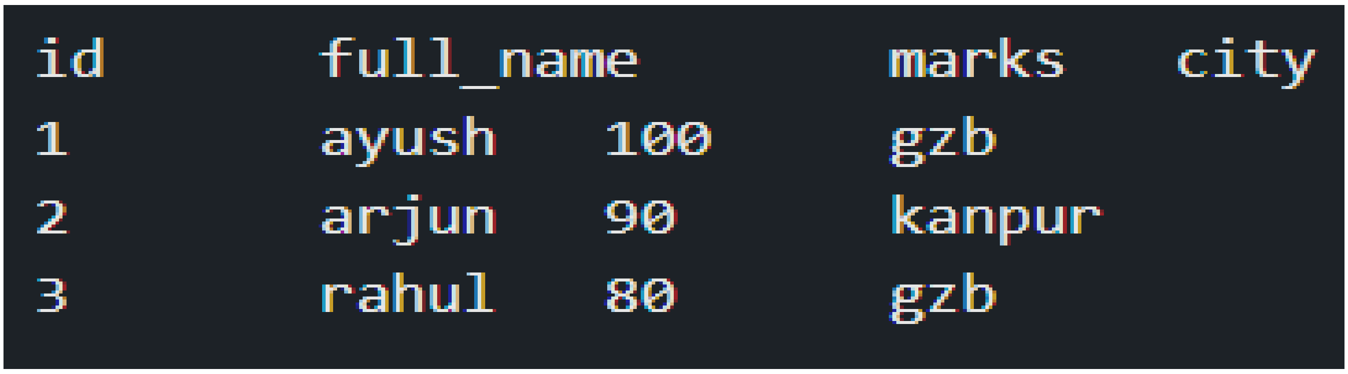
```
id      name     marks    city
1       ayush    100      gzb
2       arjun    90       kanpur
3       rahul    80       gzb
4       orion    70       kanpur
5       ayushi   60       gazipur
6       piyush   50       gazipur
```

Code:-



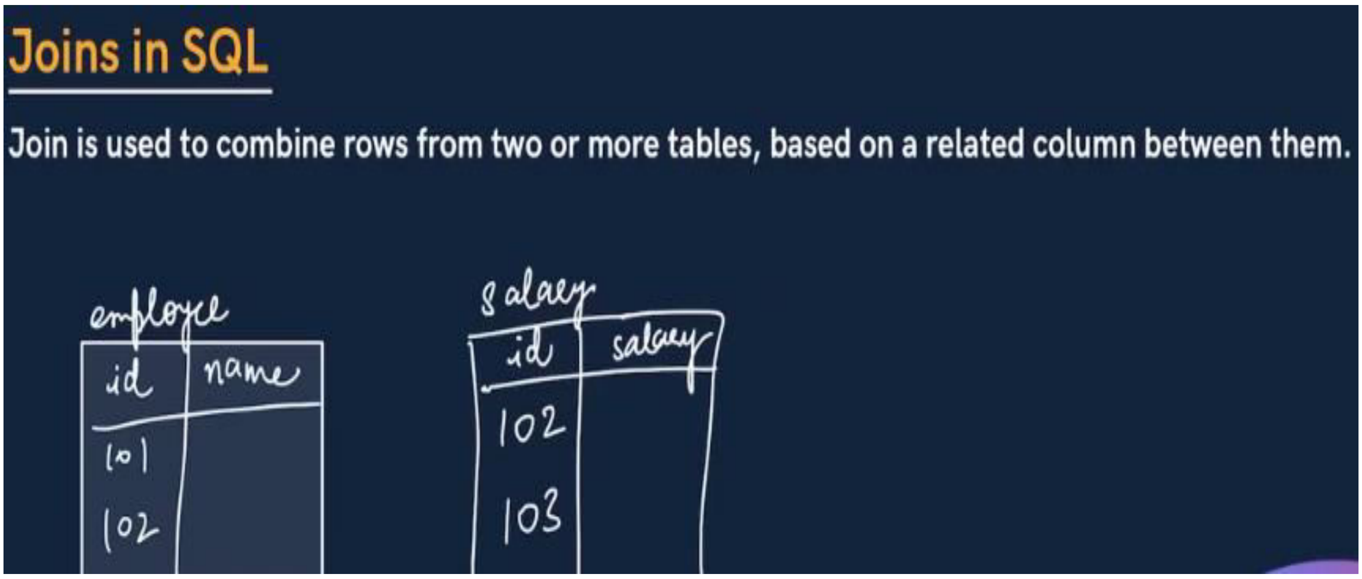
```sql
CREATE TABLE students(
  id INT PRIMARY KEY,
  name VARCHAR(50),
  marks INT NOT NULL,
  city VARCHAR(10));
INSERT INTO students(id,name,marks,city) VALUES
(1,"ayush",100,"gzb"),
(2,"arjun",90,"kanpur"),
(3,"rahul",80,"gzb"),
(4,"orion",70,"kanpur"),
(5,"ayushi",60,"gazipur"),
(6,"piyush",50,"gazipur");
ALTER TABLE students CHANGE name full_name VARCHAR(50);
DELETE FROM students WHERE marks<80;
ALTER TABLE students ADD COLUMN grades VARCHAR (3);
UPDATE students SET grades="o";
ALTER TABLE students DROP COLUMN grades;
SELECT*FROM students;
```

Table:-



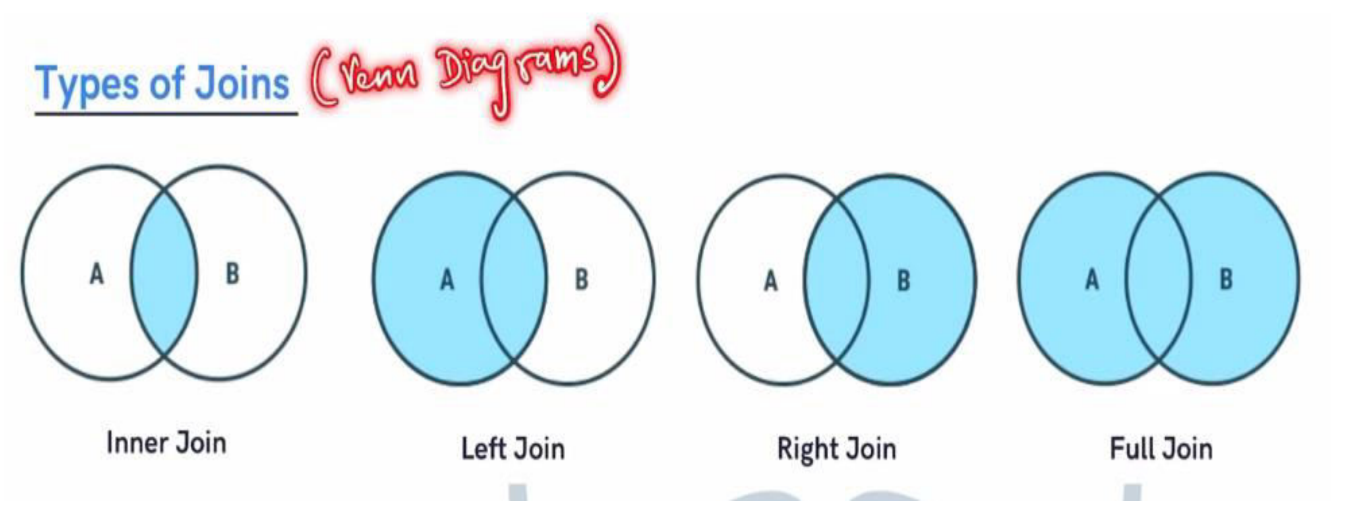
```
id      full_name           marks     city
1       ayush     100       gzb
2       arjun     90        kanpur
3       rahul     80        gzb
```

JOINS



Joins in SQL

Join is used to combine rows from two or more tables, based on a related column between them.



Types of Joins (Venn Diagrams)

Inner Join

Returns records that have matching values in both tables

Syntax

```
SELECT column(s)
FROM tableA
INNER JOIN tableB
ON tableA.col_name = tableB.col_name;
```

Example

student

student_id	name
101	adam
102	bob
103	casey

course

student_id	course
102	english
105	math
103	science
107	computer science

Result

student_id	name	course
102	bob	english
103	casey	science

Left Join

Returns all records from the left table, and the matched records from the right table

Syntax

```
SELECT column(s)
FROM tableA
LEFT JOIN tableB
ON tableA.col_name = tableB.col_name;
```

Example

student (left)

student_id	name
101 ✓	adam
102 ✓	bob
103 ✓	casey

course (Right)

student_id	course
102	english
105	math
103	science
107	computer science

Result

student_id	name	course
101 ✓	adam	*null*
102 ✓	bob	english
103 ✓	casey	science

Right Join

Returns all records from the right table, and the matched records from the left table

Syntax

```
SELECT column(s)
FROM tableA
RIGHT JOIN tableB
ON tableA.col_name = tableB.col_name;
```

Example

student

student_id	name
101	adam
102	bob
103	casey

course

student_id	course
102	english
105	math
103	science
107	computer science

Result

student_id	course	name
102	english	bob
105	math	*null*
103	science	casey
107	computer science	*null*

Full Join

FULL OUTER JOIN

Returns all records when there is a match in either left or right table

Syntax in MySQL

```
SELECT * FROM student as a
LEFT JOIN course as b
ON a.id = b.id
UNION
SELECT * FROM student as a
RIGHT JOIN course as b
ON a.id = b.id;
```

LEFT JOIN
UNION
RIGHT JOIN

Example

student

student_id	name
101	adam
102	bob
103	casey

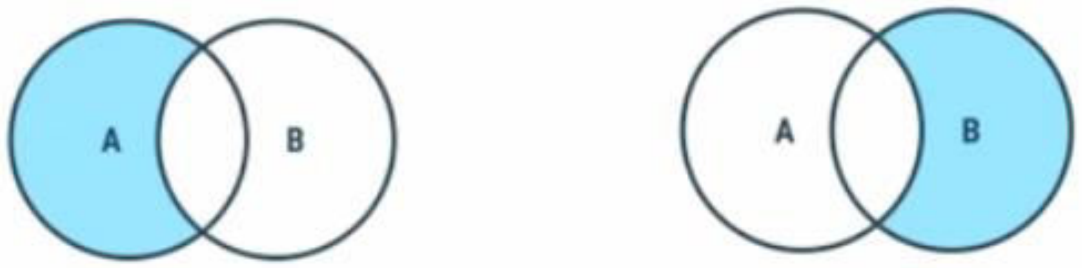
course

student_id	course
102	english
105	math
103	science
107	computer science

Result

student_id	name	course
101	adam	*null*
102	bob	english
103	casey	science
105	*null*	math
107	*null*	computer science

Qs: Write SQL commands to display the right exclusive join :



Left Exclusive Join Right Exclusive Join

```
SELECT *
FROM student as a
LEFT JOIN course as b
ON a.id = b.id
WHERE b.id IS NULL;
```

The white one is the selected part.

Self Join

It is a regular join but the table is joined with itself.

Syntax

```
SELECT column(s)
FROM table as a
JOIN table as b
ON a.col_name = b.col_name;
```

Example

Employee

id	name	manager_id
101	adam	103
102	bob	104
103	casey	*null*
104	donald	103

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

Union

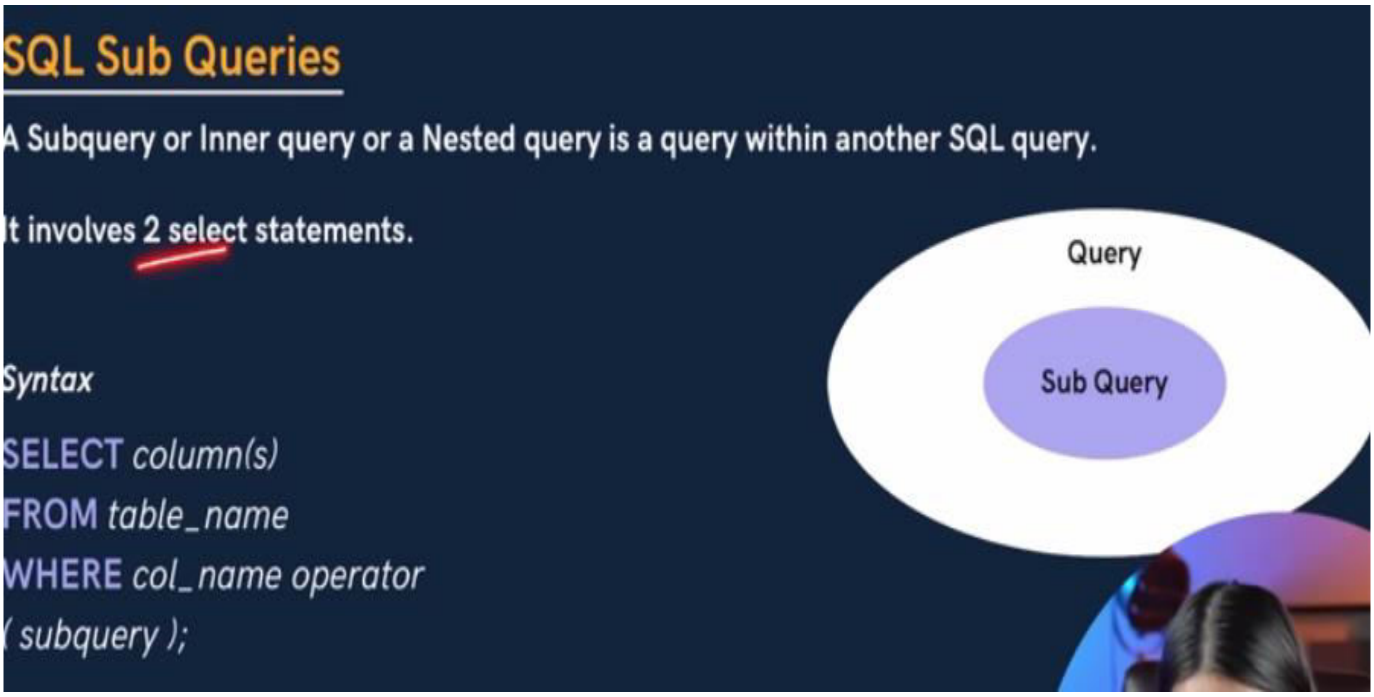
It is used to combine the result-set of two or more SELECT statements.
Gives UNIQUE records.

To use it :

- every SELECT should have same no. of columns
- columns must have similar data types
- columns in every SELECT should be in same order

Syntax

SELECT *column(s)* FROM ***tableA***
UNION
SELECT *column(s)* FROM **tableB**



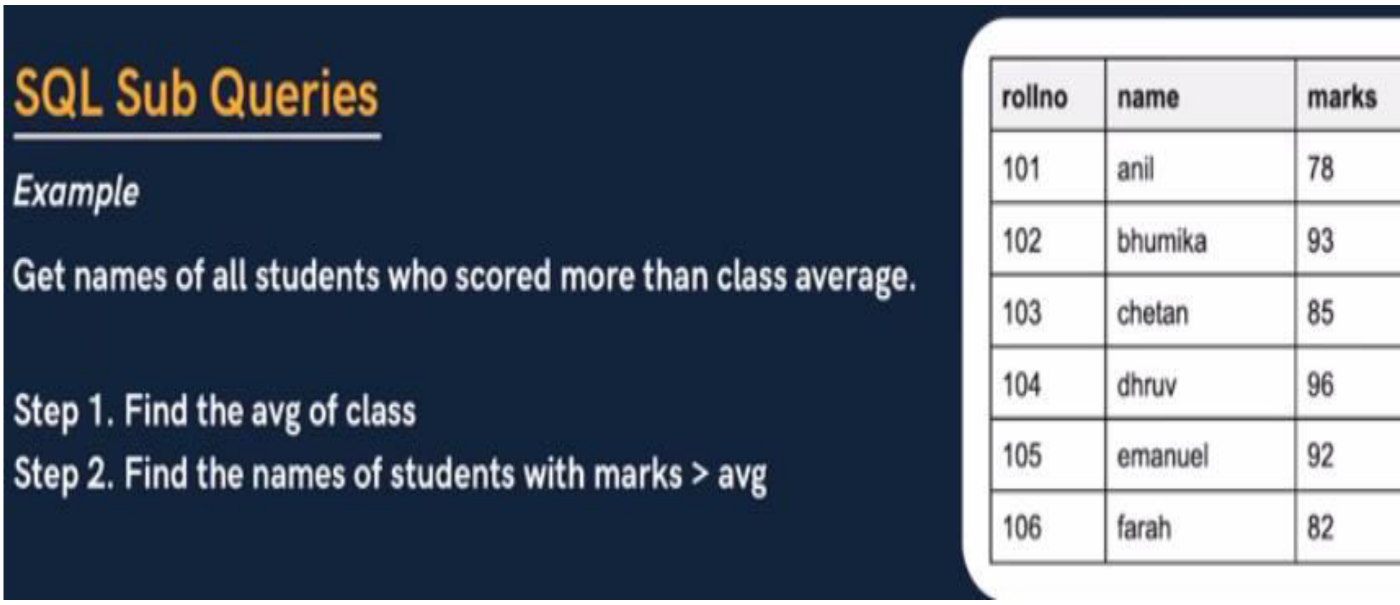
SQL Sub Queries

A Subquery or Inner query or a Nested query is a query within another SQL query.

It involves 2 select statements.

Syntax

```
SELECT column(s)
FROM table_name
WHERE col_name operator
( subquery );
```



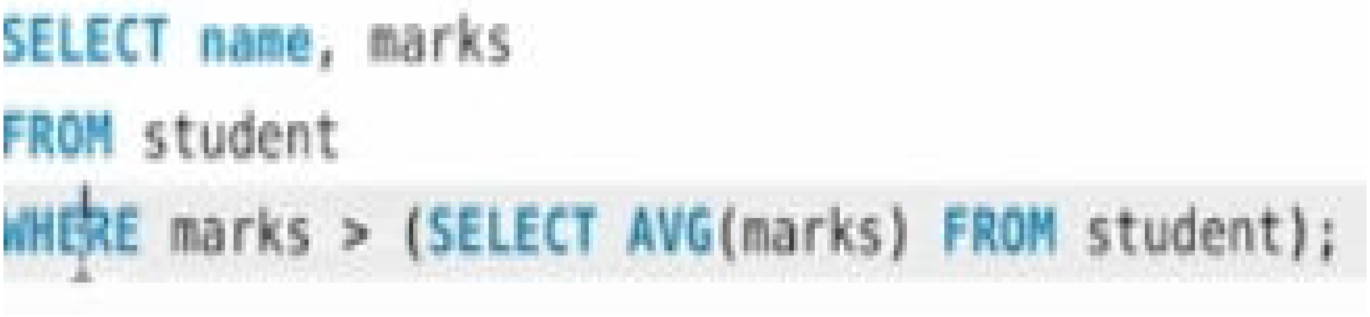
SQL Sub Queries

Example

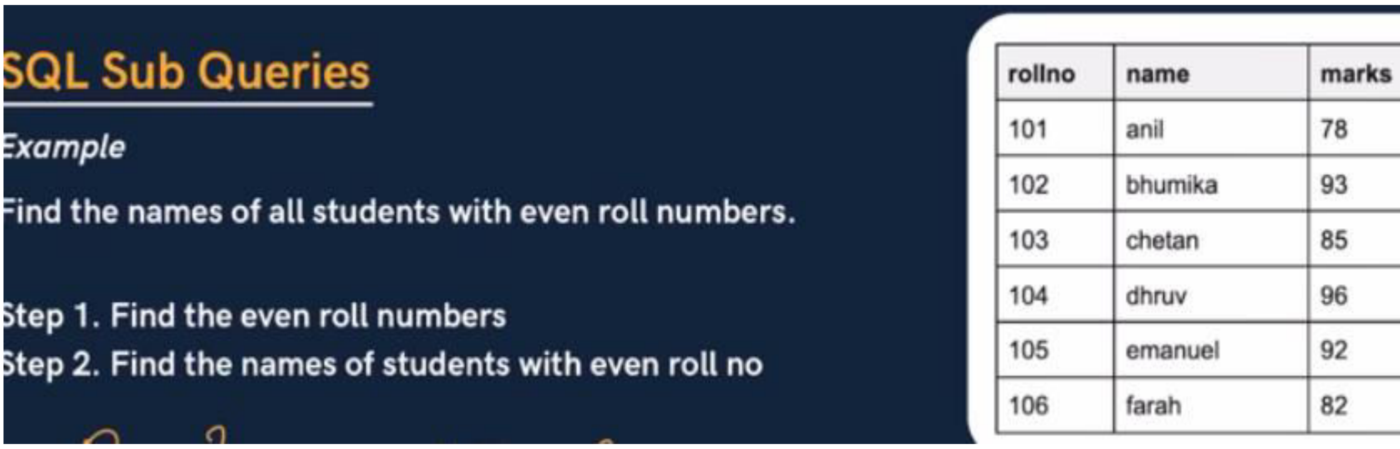
Get names of all students who scored more than class average.

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);
```



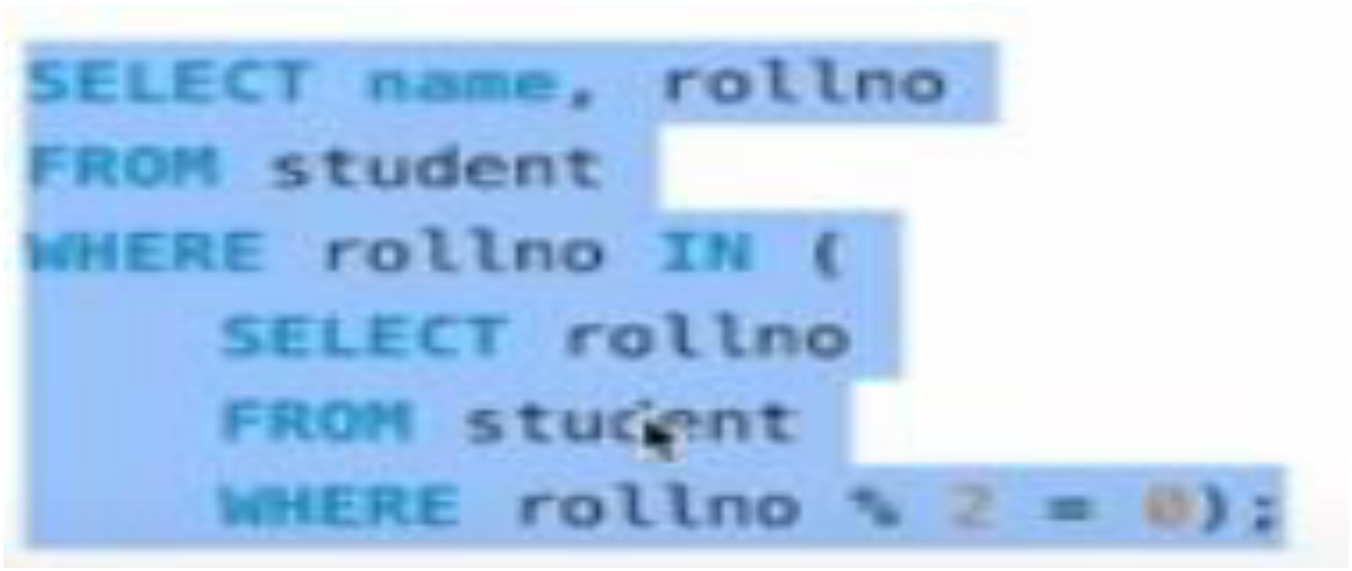
SQL Sub Queries

Example

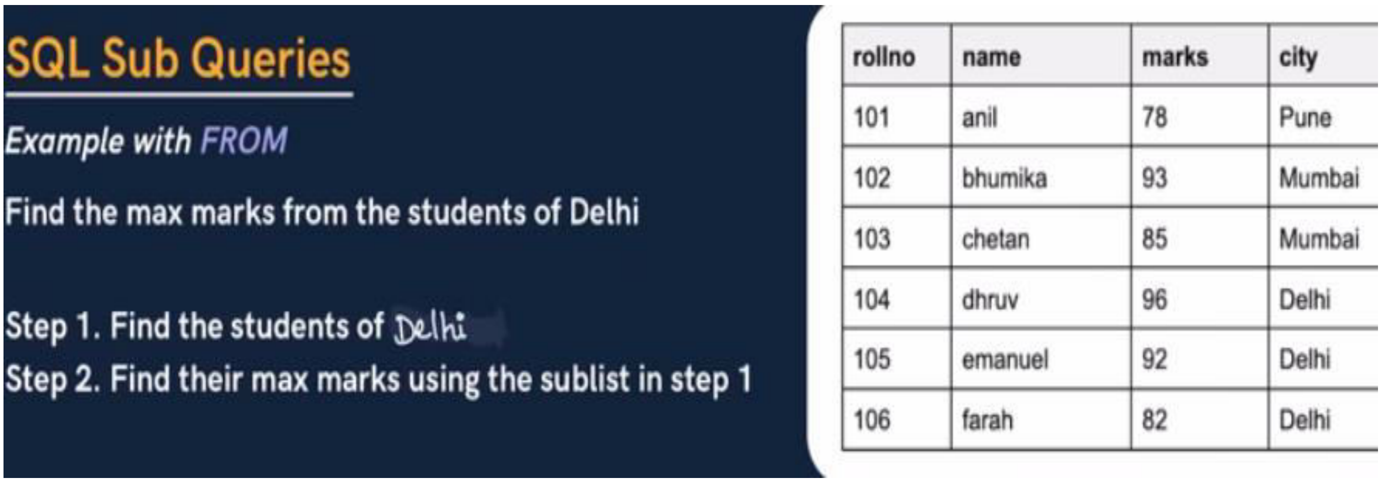
Find the names of all students with even roll numbers.

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);
```



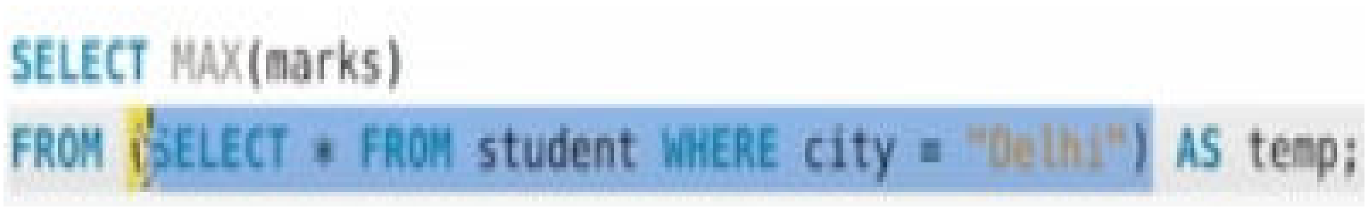
SQL Sub Queries

Example with FROM

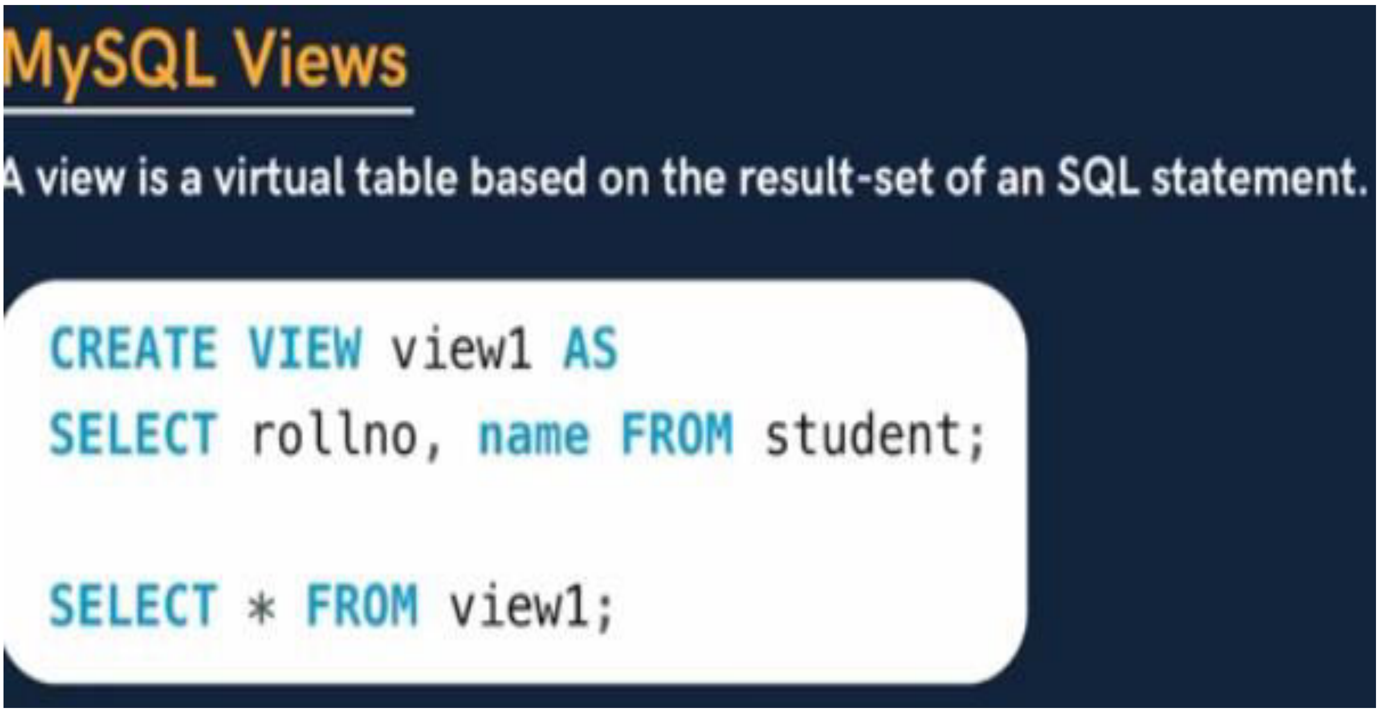
Find the max marks from the students of Delhi

Step 1. Find the students of Delhi
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



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



MySQL Views

A view is a virtual table based on the result-set of an SQL statement.

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

SELECT * FROM view1;
```

AS means declaring a new table with the selected columns.