# ---MySQL--

- **Create Database  Command and USE keyword:**

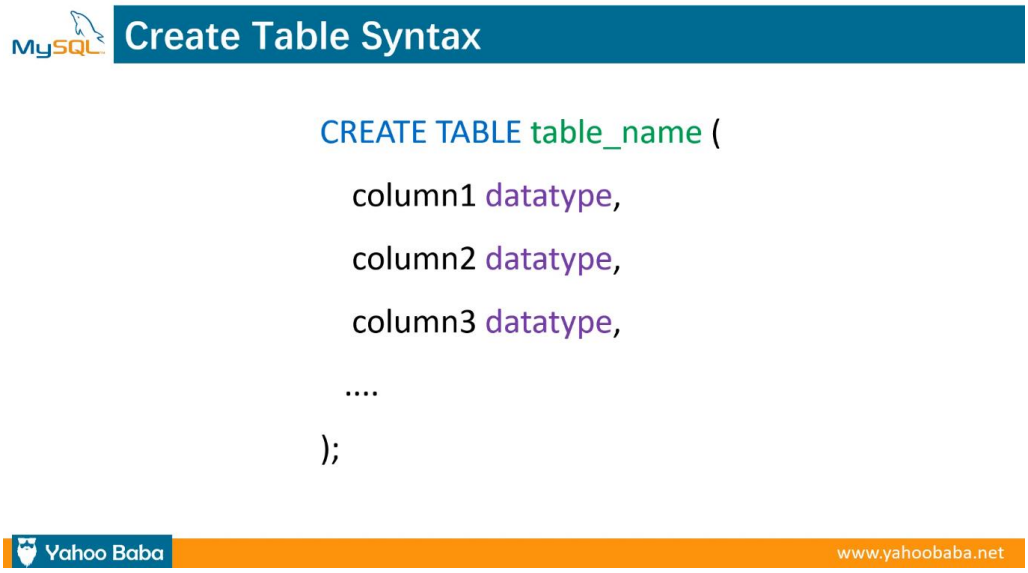
```
CREATE DATABASE DATABASE_NAME;

USE DATABASE_NAME;    // USE keyword are used for to select the database.
```

- **How to show tables from database :**

```
SHOW TABLES;
```

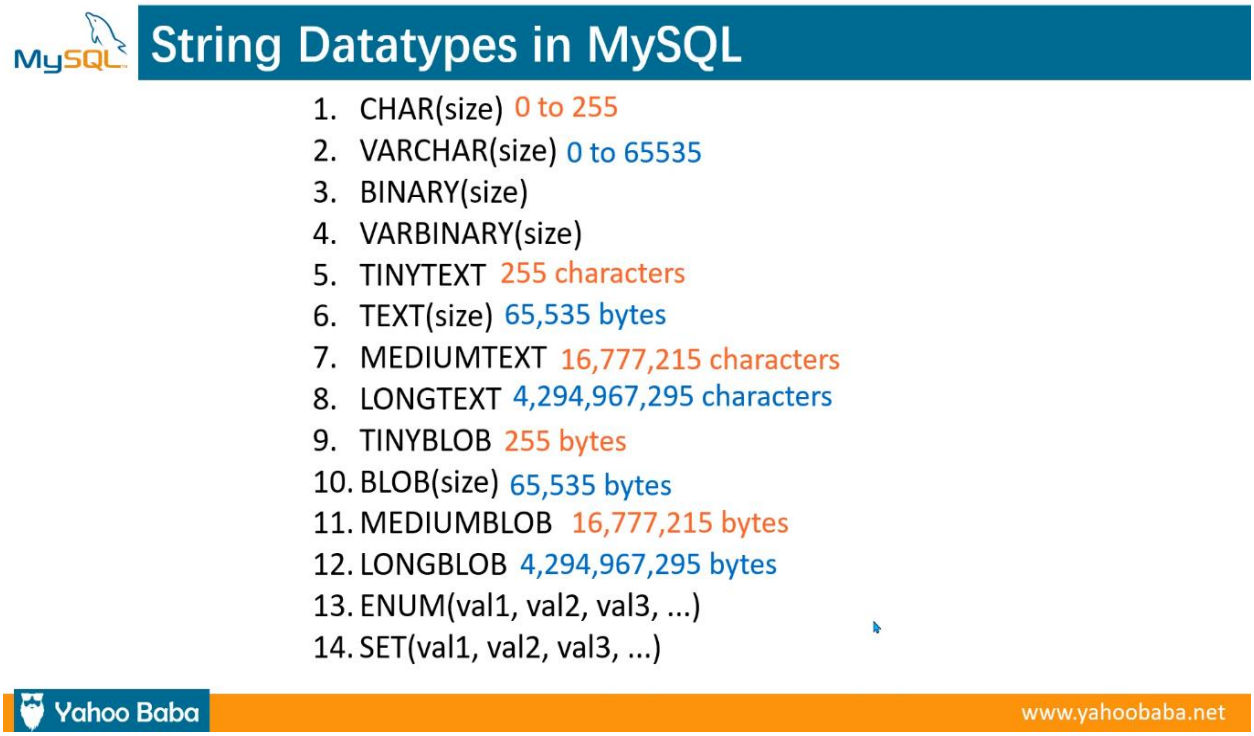
- **Create table query command :**

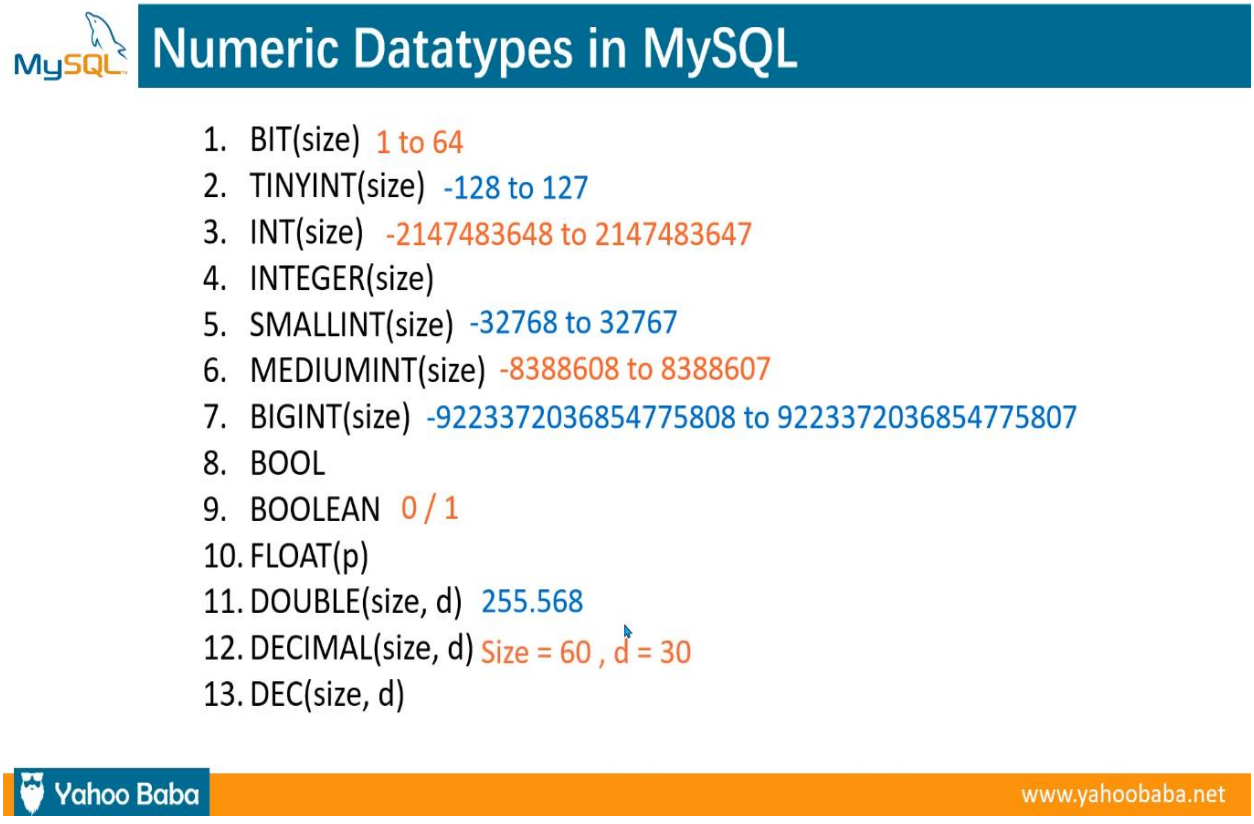



**Example :**

```
create table personal (
        id int,
        name varchar(50),
        birth_date date,
        phone varchar(12),
        gender varchar(1)
);
```

- **Datatype in SQL :**



## Date & Time Datatypes in MySQL

1. DATE '1000-01-01' to '9999-12-31'
2. DATETIME(fsp) YYYY-MM-DD hh:mm:ss
3. TIMESTAMP(fsp)
4. TIME(fsp) hh:mm:ss
5. YEAR four-digit format : 1901



- **Insert Value in Table Query Syntax:**

## How to Insert data in Tables with SQL ?

| Name | Age | Gender |
|---|---|---|
| Ram Kumar | 21 | Male |
| Salman Khan | 22 | Male |
| Meera Khan | 21 | Female |
| Sarita Kumari | 21 | Female |
| Anil Kapoor | 22 | Male |

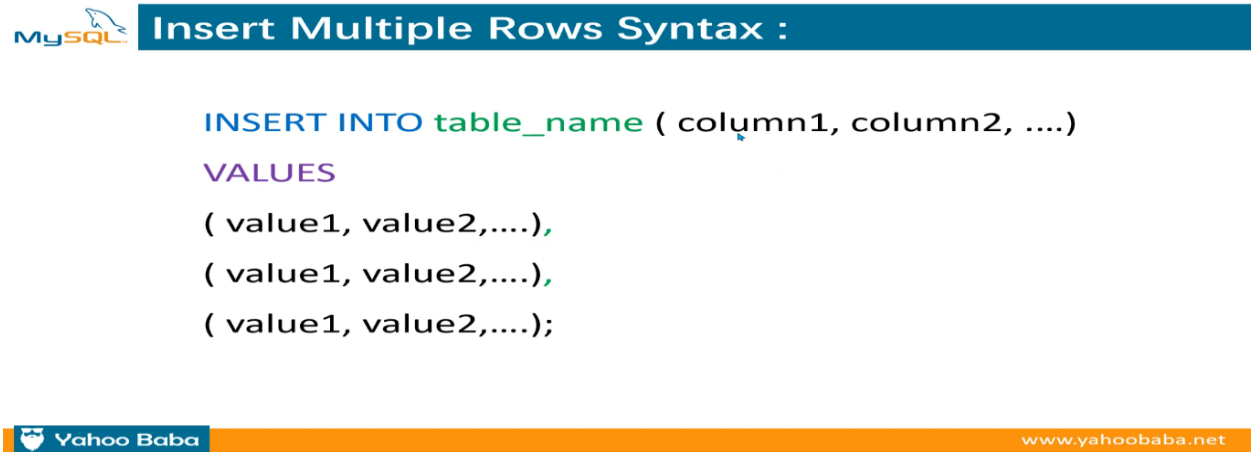
```
INSERT INTO table_name ( column1, column2, ....)
VALUES ( value1, value2,....);
```



**Example :**

```
insert into personal (id, name, birth_date, phone, gender)
values(1,"karan","1997-08-13","123467890","M");
```

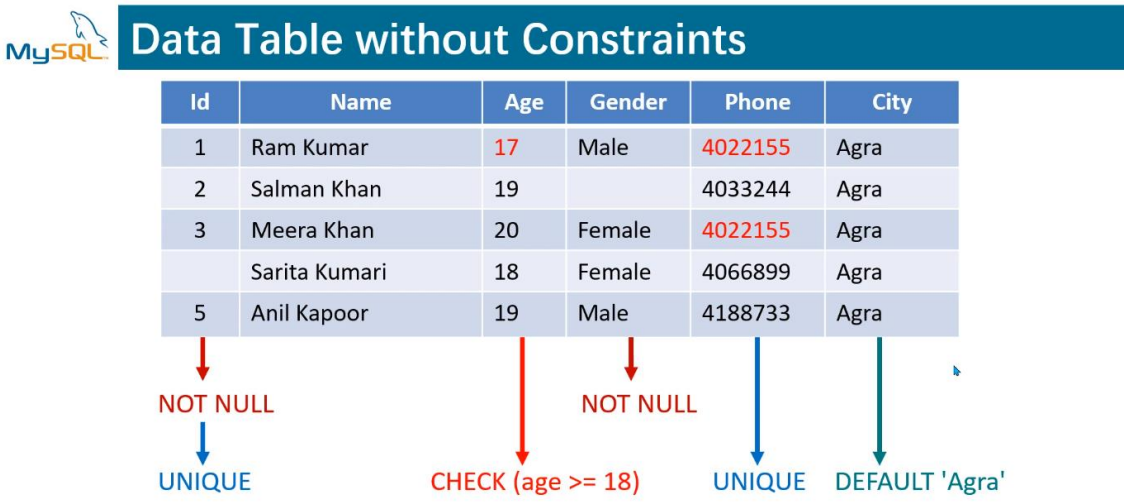
- **Insert  Multiple Values in Table Query Syntax:**




## Example :

```
insert into personal (id, name, birth_date, phone, gender)
values
(2,"Mohan","1997-12-13","12347890","M"),
(3,"Seeta","1997-09-03","12345607809","F");
```

- **List of Constraint in MySQL:**
  1. NOT NULL
  2. UNIQUE
  3. DEFAULT
  4. CHECK
  5. PRIMARY KEY
  6. FOREIGEN KEY



## Create Table Syntax

```
CREATE TABLE table_name (
    id INT NOT NULL UNIQUE,
    name VARCHAR(50) NOT NULL,
    age INT NOT NULL CHECK (age >= 18),
    gender VARCHAR(10) NOT NULL,
    phone VARCHAR(10) NOT NULL UNIQUE,
    city VARCHAR(10) NOT NULL DEFAULT 'Agra',
);
```

Yahoo Baba www.yahoobaba.net

- **Select Values From Table Syntax :**

## SELECT Syntax

```
SELECT column1, column2, column3, ....
FROM table_name;

SELECT *
FROM table_name;
```

Yahoo Baba www.yahoobaba.net

## Example:

Select name, gender, id from login;

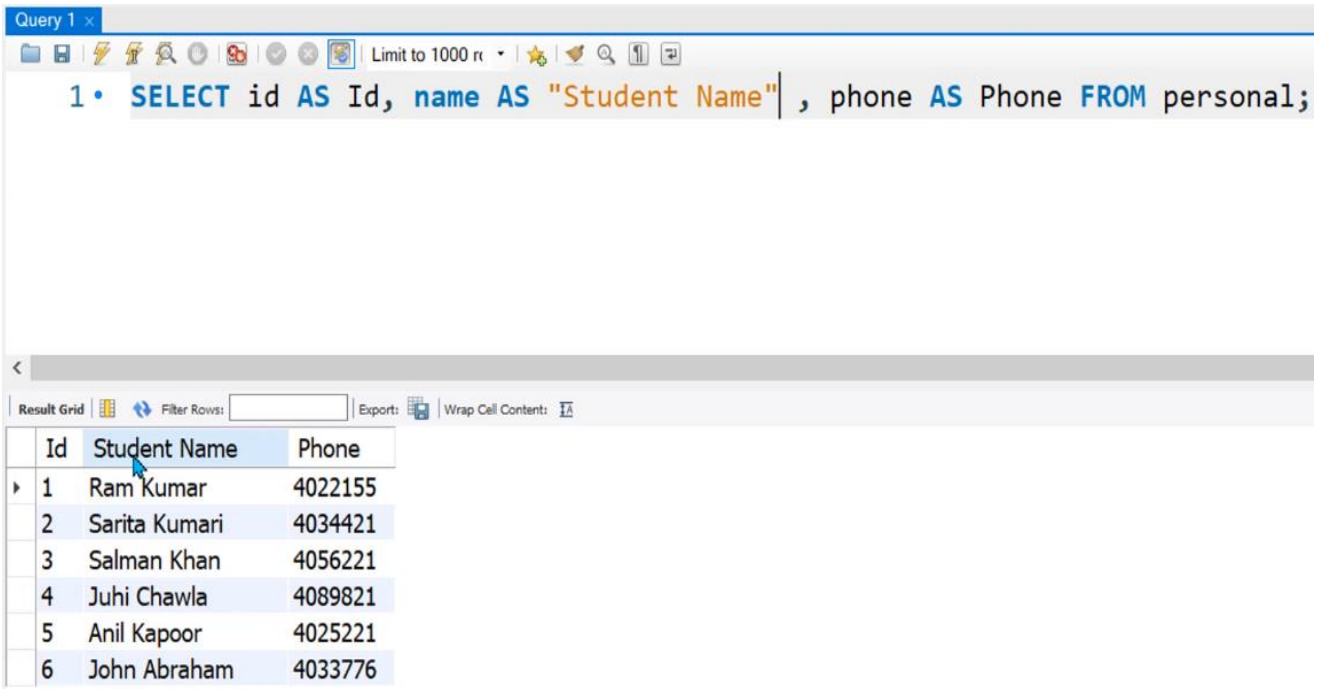
**Output:**

| | name | gender | id |
|---|---|---|---|
| ▶ | karan | M | 1 |
| | Abhishek | M | 2 |
| * | NULL | NULL | NULL |

Select name as NAME, gender as Gender, id as Id from login;

**Output:**

| | NAME | Gender | Id |
|---|---|---|---|
| ▶ | karan | M | 1 |
| | Abhishek | M | 2 |




## SELECT with WHERE Clause Syntax

```
SELECT column1, column2, column3, ....
FROM table_name
WHERE condition;
```



**Example:**

```
Select * from personal where gender="M";
```

**Output:**

| | id | name | birth_date | phone | gender |
|---|---|---|---|---|---|
| ▶ | 1 | karan | 1997-08-13 | 123467890 | M |
| | 2 | Mohan | 1997-12-13 | 12347890 | M |

# WHERE Comparison Operators

| Operator | Description |
|---|---|
| = | Equal |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal |
| <= | Less than or equal |
| <> Or != | Not equal. |
| BETWEEN | Between a certain range |
| LIKE | Search for a pattern |
| IN | To specify multiple possible values for a column |



# SELECT Data with AND & OR Operators

| Name | Age | Gender |
|---|---|---|
| Ram Kumar | 19 | Male |
| Salman Khan | 22 | Male |
| Meera Khan | 21 | Female |
| Sarita Kumari | 18 | Female |
| Anil Kapoor | 22 | Male |

WHERE Age >= 18 AND Age <= 21

| Name | Age | Gender |
|---|---|---|
| Ram Kumar | 19 | Male |
| Meera Khan | 21 | Female |
| Sarita Kumari | 18 | Female |

WHERE Age = 18 OR Age = 21

| Name | Age | Gender |
|---|---|---|
| Meera Khan | 21 | Female |
| Sarita Kumari | 18 | Female |



# Use of NOT operator

Query 1

Limit to 1000 r

```
1 • SELECT * FROM personal
2   WHERE NOT city = "Bhopal" OR city = "Agra";
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| id | name | age | gender | phone | city |
|---|---|---|---|---|---|
| 1 | Ram Kumar | 19 | M | 4022155 | Agra |
| 2 | Sarita Kumari | 21 | F | 4034421 | Delhi |
| 3 | Salman Khan | 20 | M | 4056221 | Agra |
| 5 | Anil Kapoor | 22 | M | 4025221 | Agra |
| 6 | John Abraham | 21 | M | 4033776 | Delhi |

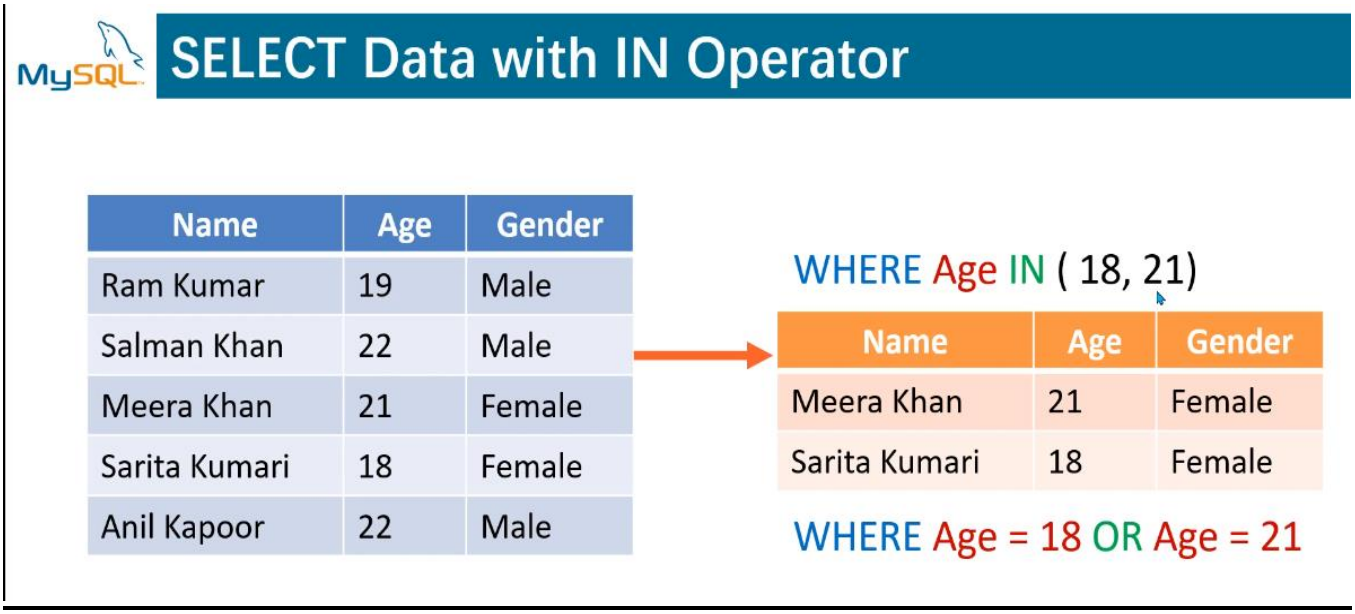
```
1 • SELECT * FROM personal
2   WHERE NOT (city = "Bhopal" OR city = "Agra");
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| id | name | age | gender | phone | city |
|---|---|---|---|---|---|
| 2 | Sarita Kumari | 21 | F | 4034421 | Delhi |
| 6 | John Abraham | 21 | M | 4033776 | Delhi |

## Lecture-9

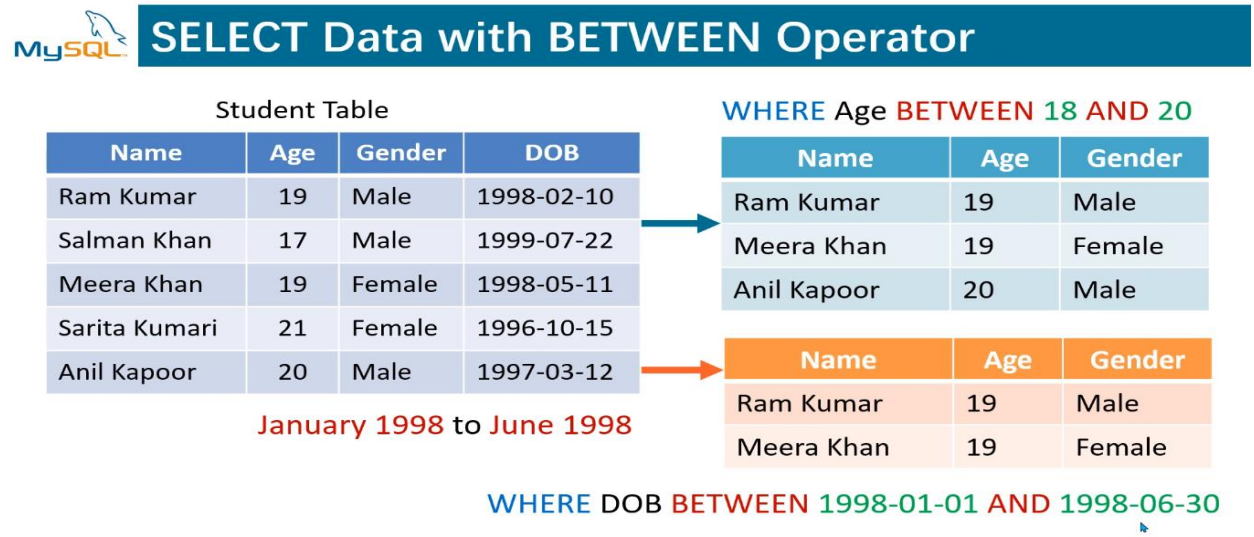
**IN operator:** we are used IN operator in the place of OR operator.




**Syntax of IN Operator:**

```
SELECT column1, column2, column3, ....
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

## Lecture 10:






```
SELECT column1, column2, column3, ....
FROM table_name
WHERE column_name BETWEEN value1 AND value2;

SELECT column1, column2, column3, ....
FROM table_name
WHERE column_name NOT BETWEEN value1 AND value2;
```

---

## Lecture 11:

**The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.**

There are two wildcards often used in conjunction with the LIKE operator:

- % - The percent sign represents zero, one, or multiple characters
- _ - The underscore represents a single character

### LIKE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

# LIKE Operator with Wildcard Patterns

| Pattern | Description |
|---|---|
| LIKE 'a%' | Start with "a" |
| LIKE '%a' | End with "a" |
| LIKE '%am%' | Have "am" in any position |
| LIKE 'a%m' | Start with "a" and Ends with "m" |
| LIKE '_a%' | "a" in the second position |
| LIKE '__a%' | "a" in the third position |
| LIKE '_oy' | "o" in the second and "y" in the third position |

Example

```
SELECT * FROM Customers
WHERE CustomerName LIKE '%a';
```

## Lecture 12:

## Regular expression in MySQL

# SELECT Data with Regular Expression

Student Table

| Name | Age | Gender |
|---|---|---|
| Ram Kumar | 19 | Male |
| Salman Khan | 18 | Male |
| Meera Khan | 19 | Female |
| Sarita Kumari | 21 | Female |
| Anil Kapoor | 20 | Male |

WHERE Name REGEXP "khan$ | poo

| Name | Age | Gender |
|---|---|---|
| Salman Khan | 18 | Male |
| Meera Khan | 19 | Female |
| Anil Kapoor | 20 | Male |

# Regular Expression Patterns with Description

| Sign | Pattern | Description |
|---|---|---|
| ^ | '^ra' | Beginning of string |
| $ | 'an$' | End of string |
| [...] | '[rms]' | Any character listed between the square brackets |
| ^[...] | '^[rms]' | Begins with Any character listed between the square brackets |
| [a-z] | '[a-h]e' | Match with in the range |
| p1\|p2\|p3 | 'tom\|dick\|harry' | matches any of the patterns p1, p2, or p3 |

**Example:**

```
select * from login where name REGEXP 'abhishek';
```

**Output:**

Result Grid | Filter Rows:

| id | name | age | phone | gender |
|---|---|---|---|---|
| 2 | Abhishek | 18 | 12367890 | M |
| NULL | NULL | NULL | NULL | NULL |

---

## Lecture 13:

# SELECT Data with ORDER BY

Student Table

| Name | Age | Gender |
|---|---|---|
| Ram Kumar | 19 | Male |
| Salman Khan | 18 | Male |
| Meera Khan | 19 | Female |
| Sarita Kumari | 21 | Female |
| Anil Kapoor | 20 | Male |

Ascending Order

| Name | Age | Gender |
|---|---|---|
| Anil Kapoor | 20 | Male |
| Meera Khan | 19 | Female |
| Ram Kumar | 19 | Male |
| Salman Khan | 18 | Male |
| Sarita Kumari | 21 | Female |

ORDER BY Name ASC

ORDER BY Name DESC

---

```
SELECT column1, column2, column3, ....
FROM table_name
ORDER BY column1, column2, .... ASC | DESC;
```

Note:- Ascending order is default order in mysql.

**Example:**

```
select * from login order by name desc;
select * from login order by name asc;
```

## SELECT with DISTINCT Syntax

```
SELECT DISTINCT column1, column2, ....
FROM table_name;
```

**Example:**

```
select distinct gender from personal ;
```

| gender |
|---|
| M |
| F |

Distinct are use for show only unique value, not a duplicate value.

# Lecture-14

## SELECT with IS NULL Syntax

```
SELECT column1, column2, column3, ....
FROM table_name
WHERE column IS NULL;

SELECT column1, column2, column3, ....
FROM table_name
WHERE column IS NOT NULL;
```

Yahoo Baba www.yahoobaba.net

**Note:-**

Find record from table wher data are contain null value in the Column.

# Lecture-15

## Limit and Offset

## SELECT with LIMIT Syntax

```
SELECT column1, column2, column3, ....
FROM table_name
WHERE condition
LIMIT number;
```

**Note:-** Limit are use for show limited record given by user. Where condition are optional.

**Example:-**

```
select * from personal limit 2
```

above example only show 2 records from database.

## Syntax of OFFSET:-

```
SELECT column1, column2, column3, ..
FROM table_name
WHERE condition
LIMIT offset , number;
```

## Example:-

```
select * from personal limit 2,1;
```

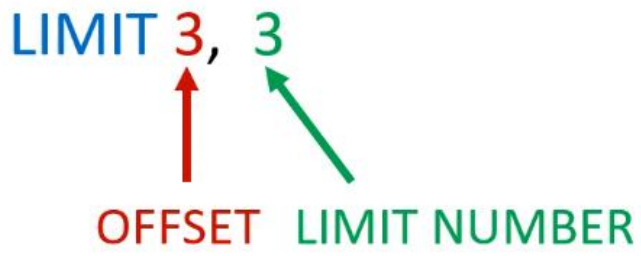
## Output:-

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| | id | name | birth_date | phone | gender |
|---|---|---|---|---|---|
| ▶ | 3 | Seeta | 1997-09-03 | 12345607809 | F |

---

Offset is use for show random records from the database.



# Lecture-16

# Aggregate Function

## SELECT Data with Aggregate Functions

Employee Table

| Name | Age | Gender | Salary |
|---|---|---|---|
| Ram Kumar | 19 | Male | 4500 |
| Salman Khan | 18 | Male | 5200 |
| Meera Khan | 20 | Female | 6000 |
| Sarita Kumari | 21 | Female | 8500 |
| Anil Kapoor | 20 | Male | 6300 |
| Shahid Kapoor | 19 | Male | 4800 |
| Virat Kohli | 21 | Male | 5700 |

COUNT(column_name)

MAX(column_name)

MIN(column_name)

SUM(column_name)

AVG(column_name)



## SELECT with Aggregate Functions Syntax

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;

SELECT SUM(column_name)
FROM table_name
WHERE condition;
```



## Example

```
SELECT COUNT(gender) FROM student
WHERE gender="M";
```

**Output:-**

| | COUNT(gender) |
|---|---|
| ▶ | 5 |

# Lecture-17

## Syntax of Update

```
UPDATE table_name
SET column1_name = value1, column2_name = value2,...
WHERE condition;
```

Without where clause we are not update specific record. It is update whole column.

Employee Table

| Id | Name | Age | Salary |
|---|---|---|---|
| 1 | Ram Kumar | 19 | 4500 |
| 2 | Salman Khan | 18 | 5200 |
| 3 | Sarita Kumari | 21 | 8500 |
| 4 | Anil Kapoor | 20 | 6300 |

6000

```
UPDATE employee
SET Salary = 6000
WHERE Id = 2;
```

# Lecture-18

## Commit Rollback

Commit and rollback command are only work on insert, update and delete.

Commit command use for save all states and query and rollback are use for go previous state. That means if we are update wrong record and trying to correct that record then we use rollback command.

**Example:-**

```
commit;    ( save all old state )
UPDATE personal SET name='karan'
WHERE id=2;
 rollback;        (rollback to previous state and correct that record)
```

# Lecture-19

## Syntax of Delete

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

```
DELETE FROM table_name;
```

# Lecture-20

## Primary and Foreign Key

1. Primary key always has unique data. (unique constraint contain null value but primary key cannot contain any null value)
2. A primary key cannot have null value.
3. In a table, can contain only one primary key. (only one column is primary key in a table).

### Create Table with PRIMARY KEY Syntax

```
CREATE TABLE table_name (
    id INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    age INT NOT NULL,
    city VARCHAR(10) NOT NULL ,
    PRIMARY KEY (id)
);
```

If table is already created then we can set primary key by using alter command.

## Alter Table with PRIMARY KEY Syntax

```
ALTER TABLE table_name
ADD PRIMARY KEY (id);
```

**Foreign key:-**

- A FOREIGN KEY is a key used to link two tables together.
- A FOREIGN key in one table used to point PRIMARY key in another table.

## FOREIGN KEY in Table

Student Table

| Id | Name | Age | City |
|---|---|---|---|
| 1 | Ram Kumar | 19 | 1 |
| 2 | Salman Khan | 18 | 2 |
| 3 | Meera Khan | 19 | 1 |
| 4 | Sarita Kumari | 21 | 3 |

City Table

| Cid | City |
|---|---|
| 1 | Agra |
| 2 | Bhopal |
| 3 | Delhi |

Link

PRIMARY KEY FOREIGN KEY PRIMARY KEY

## Create Table with FOREIGN KEY Syntax

```
CREATE TABLE student(
   id INT NOT NULL AUTO_INCREMENT,
   name VARCHAR(50) NOT NULL,
   age INT NOT NULL,
   city VARCHAR(10) NOT NULL ,
   PRIMARY KEY (id),
   FOREIGN KEY (city) REFERENCES City (cid)
);
```

# Alter Table with FOREIGN KEY Syntax

```
ALTER TABLE table_name
ADD FOREIGN KEY (city) REFERENCES City (cid);
```

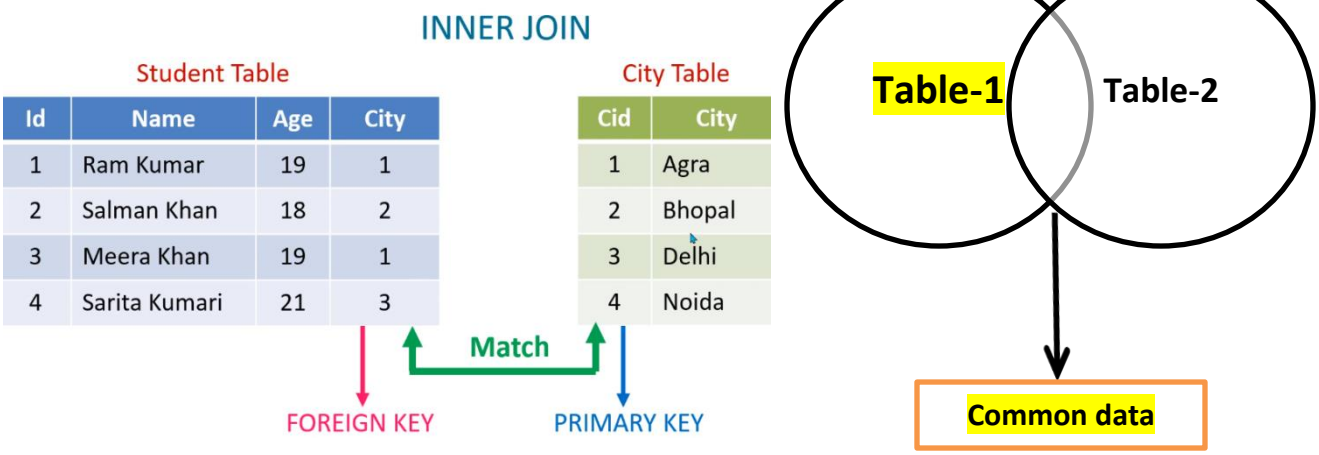
## Lecture-21

## Join in MySQL(Inner Join)

### Types of JOINS in MySQL:-

- Inner join
- Left join
- Right join
- Cross join

**Inner join:-** The inner join select records that have matching in both tables.




**Syntax of inner join:-**

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name;
```

FOREIGN KEY — PRIMARY KEY

# Example of inner join and Primary key & Foreign key

**Step-1** Creates **city** table.

```
CREATE TABLE city(
    cid int not null auto_increment,
    cityname varchar(50) not null,
    primary key(cid)
    );
```

**Step-2** Insert the value in city table.

```
insert into city(cityname)
values('Agra'),
      ('Gorakhpur'),
      ('Delhi'),
      ('Bhopal'),
      ('Noida');
```

**Step-3** Creates student table.

```
CREATE TABLE student (
      id int not null,
      name varchar(50) not null,
      percentage int not null,
      age int not null,
      gender varchar(1) not null,
      city int not null,
      primary key (id),
      foreign key (city) references city (cid)
      );
```

**Step-4** Insert the value in city table.

```
insert into student(id, name, percentage, age, gender, city)
values(1,'Ram Kumar',45,19,"M",1),   (2,'Sarita Kumari',55,22,"F",2),
      (3,'Slaman Khan',62,20,"M",4), (4,'Juhi Chawla',47,18,"F",3),
      (5,'Anil Kapoor',74,22,"M",5),   (6,'Rohan Yadav',35,20,"M",3),
      (7,'Karan Jaiswal',85,24,"M",3), (8,'Seema Dubey',55,22,"F",3);
```

**Step-5** Link both **student and city** tables by using **Inner join**.

```
SELECT * FROM student
        INNER JOIN city
        ON student.city= city.cid;
```

in above query we use allies such as.

```
SELECT * FROM student s
        INNER JOIN city c
       ON s.city=c.cid;
```

here **'s'** & **'c'** latter are allies of student and city tables.

## Outout:-

| id | name | percentage | age | gender | city | cid | cityname |
|---|---|---|---|---|---|---|---|
| 1 | Ram Kumar | 45 | 19 | M | 1 | 1 | Agra |
| 2 | Sarita Kumari | 55 | 22 | F | 2 | 2 | Gorakhpur |
| 4 | Juhi Chawla | 47 | 18 | F | 3 | 3 | Delhi |
| 6 | Rohan Yadav | 35 | 20 | M | 3 | 3 | Delhi |
| 7 | Karan Jaiswal | 85 | 24 | M | 3 | 3 | Delhi |
| 8 | Seema Dubey | 55 | 22 | F | 3 | 3 | Delhi |
| 3 | Slaman Khan | 62 | 20 | M | 4 | 4 | Bhopal |
| 5 | Anil Kapoor | 74 | 22 | M | 5 | 5 | Noida |

---

In output, city and cid columns are show but if we are showing specific column Then we use other query like..

```
SELECT s.id, s.name, s.percentage, s.age, s.gender, c.cityname
        FROM student s
        INNER JOIN city c
        ON s.city=c.cid;
```

## Outout:-

| id | name | percentage | age | gender | cityname |
|---|---|---|---|---|---|
| 1 | Ram Kumar | 45 | 19 | M | Agra |
| 2 | Sarita Kumari | 55 | 22 | F | Gorakhpur |
| 4 | Juhi Chawla | 47 | 18 | F | Delhi |
| 6 | Rohan Yadav | 35 | 20 | M | Delhi |
| 7 | Karan Jaiswal | 85 | 24 | M | Delhi |
| 8 | Seema Dubey | 55 | 22 | F | Delhi |
| 3 | Slaman Khan | 62 | 20 | M | Bhopal |
| 5 | Anil Kapoor | 74 | 22 | M | Noida |

---

**OR**

```
SELECT s.id, s.name, s.percentage, s.age, s.gender, c.cityname as "City Name"
        FROM student s
        INNER JOIN city c
        ON s.city=c.cid;
```

**Outout:-**

| id | name | percentage | age | gender | City Name |
|---|---|---|---|---|---|
| 1 | Ram Kumar | 45 | 19 | M | Agra |
| 2 | Sarita Kumari | 55 | 22 | F | Gorakhpur |
| 4 | Juhi Chawla | 47 | 18 | F | Delhi |
| 6 | Rohan Yadav | 35 | 20 | M | Delhi |
| 7 | Karan Jaiswal | 85 | 24 | M | Delhi |
| 8 | Seema Dubey | 55 | 22 | F | Delhi |
| 3 | Slaman Khan | 62 | 20 | M | Bhopal |
| 5 | Anil Kapoor | 74 | 22 | M | Noida |

- **Inner join with where clause for showing specific data:-**

```
SELECT s.id, s.name, s.percentage, s.age, s.gender, c.cityname as "City Name"
        FROM student s
        INNER JOIN city c
        ON s.city=c.cid
        WHERE c.cityname=”delhi”;
```

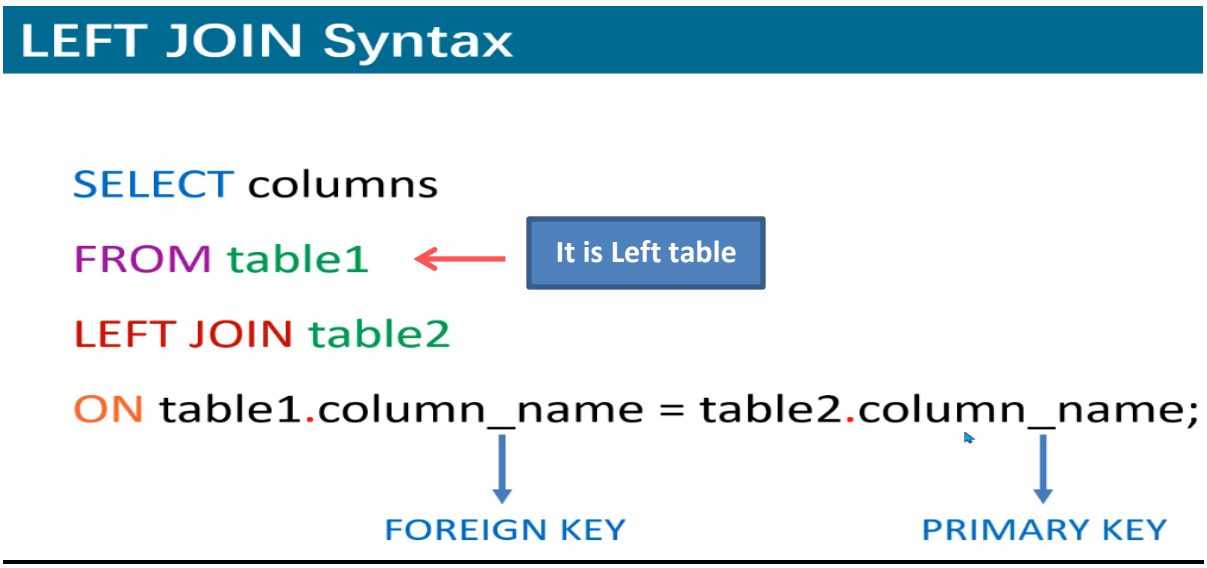
**Outout:-**

| id | name | percentage | age | gender | City Name |
|---|---|---|---|---|---|
| 4 | Juhi Chawla | 47 | 18 | F | Delhi |
| 6 | Rohan Yadav | 35 | 20 | M | Delhi |
| 7 | Karan Jaiswal | 85 | 24 | M | Delhi |
| 8 | Seema Dubey | 55 | 22 | F | Delhi |

# Lecture-22

## Join in MySQL(Left Join and Right Join)

**Left Join:-** left join are use for show all data of left table as well as common data in both tables but inner join not show missing data. Table1 is left table.(before left join table is called left table)




**Query:-**

```
SELECT *
FROM student s
LEFT JOIN city c
ON s.city=c.cid;
```

**Outout:-**

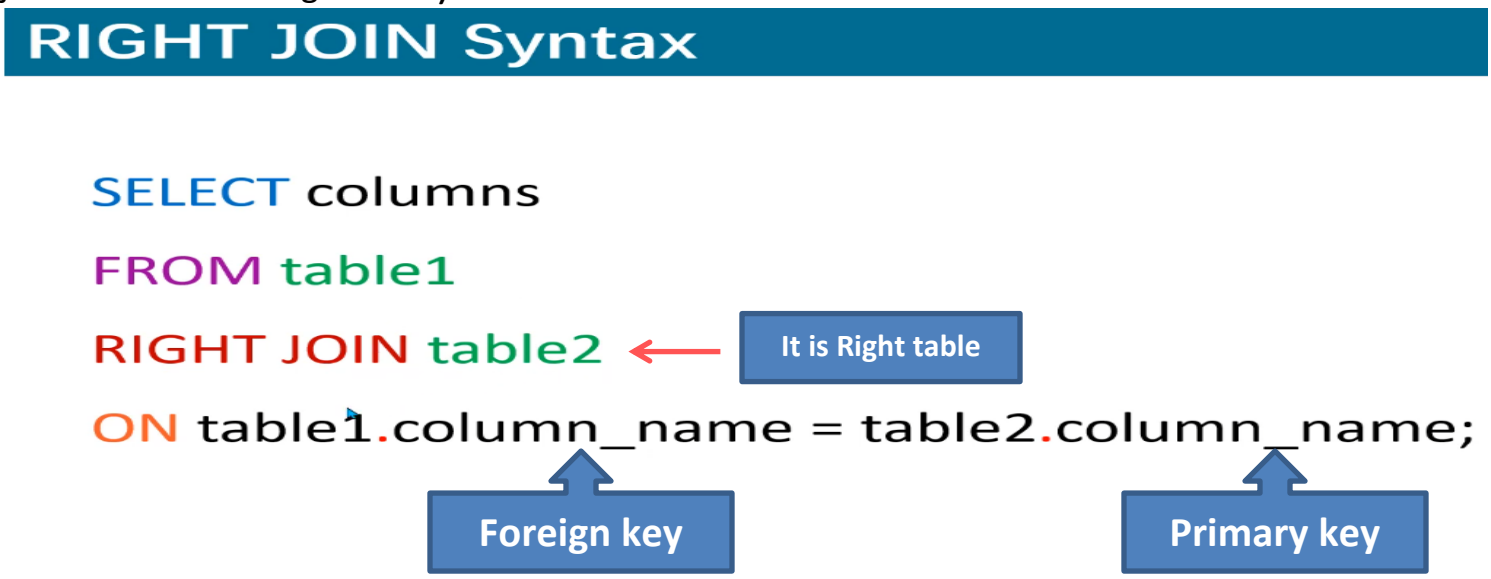
| id | name | percentage | age | gender | city | cid | cityname |
|---|---|---|---|---|---|---|---|
| 1 | Ram Kumar | 45 | 19 | M | 1 | 1 | Agra |
| 2 | Sarita Kumari | 55 | 22 | F | 2 | 2 | Gorakhpur |
| 3 | Slaman Khan | 62 | 20 | M | 4 | 4 | Bhopal |
| 4 | Juhi Chawla | 47 | 18 | F | 2 | 2 | Gorakhpur |
| 5 | Anil Kapoor | 74 | 22 | M | NULL | NULL | NULL |
| 6 | Rohan Yadav | 35 | 20 | M | 3 | 3 | Delhi |
| 7 | Karan Jaiswal | 85 | 24 | M | 4 | 4 | Bhopal |
| 8 | Seema Dubey | 55 | 22 | F | 3 | 3 | Delhi |

Row 5 (id no. 5) and in student column name city contain null value but left join are show remaining all data in row 5( id, name, percentage, age, gender).

Student table

City table

**Right Join:-** Right Join is opposite of left join. Table 2 is Right table(after right join table is called right table)




**Query:-**

```
SELECT *
FROM student s
RIGHT JOIN city c
ON s.city=c.cid;
```

**Outout:-**

| id | name | percentage | age | gender | city | cid | cityname |
|---|---|---|---|---|---|---|---|
| 1 | Ram Kumar | 45 | 19 | M | 1 | 1 | Agra |
| 2 | Sarita Kumari | 55 | 22 | F | 2 | 2 | Gorakhpur |
| 3 | Slaman Khan | 62 | 20 | M | 4 | 4 | Bhopal |
| 4 | Juhi Chawla | 47 | 18 | F | 2 | 2 | Gorakhpur |
| 6 | Rohan Yadav | 35 | 20 | M | 3 | 3 | Delhi |
| 7 | Karan Jaiswal | 85 | 24 | M | 4 | 4 | Bhopal |
| 8 | Seema Dubey | 55 | 22 | F | 3 | 3 | Delhi |
| NULL | NULL | NULL | NULL | NULL | NULL | 5 | Noida |

Student table

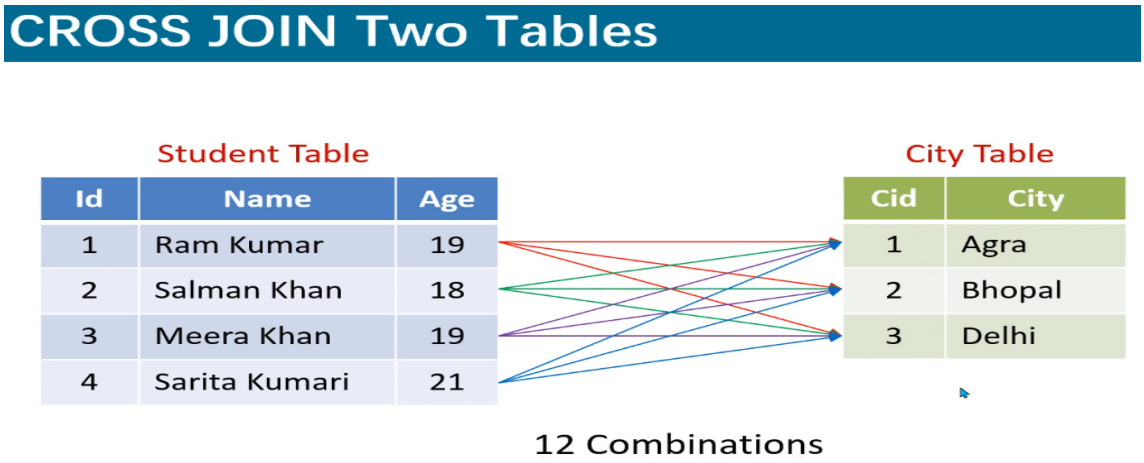
City table

Row 5 (id no. 5) and in student column name city contain null value but RIGHT join are show remaining all data in city table (cid, City name) and student contain null value in id 5.

# Lecture-23

## Join in MySQL(Cross Join)

**Cross Join:-** in cross join, no need of primary key and foreign key.
It is join every row of first table to every row of second table.




## Syntax of cross join:-

```
SELECT columns
FROM table1
CROSS JOIN table2;
```

## Example:-

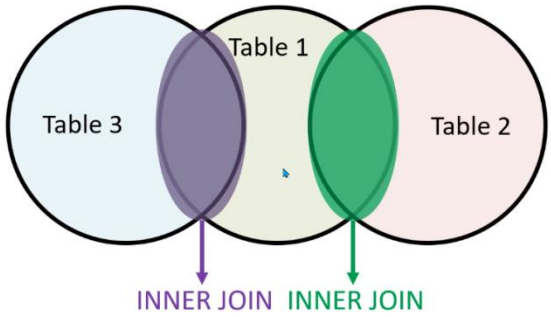
```
SELECT *
FROM student
CROSS JOIN city;
```

## Outout:-

| Id | Name | Age | Cid | City |
|---|---|---|---|---|
| 1 | Ram Kumar | 19 | 1 | Agra |
| 2 | Ram Kumar | 19 | 2 | Bhopal |
| 3 | Ram Kumar | 19 | 3 | Delhi |
| 4 | Salman Khan | 18 | 1 | Agra |
| 5 | Salman Khan | 18 | 2 | Bhopal |
| 6 | Salman Khan | 18 | 3 | Delhi |
| 7 | Meera Khan | 19 | 1 | Agra |
| 8 | Meera Khan | 19 | 2 | Bhopal |
| 9 | Meera Khan | 19 | 3 | Delhi |
| 10 | Sarita Kumari | 21 | 1 | Agra |
| 11 | Sarita Kumari | 21 | 2 | Bhopal |
| 12 | Sarita Kumari | 21 | 3 | Delhi |

# Lecture-24

## Join Multiple table in MySQL




## Syntax of Join Multiple:-

```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column_name = table2.column_name
INNER JOIN table3
ON table1.column_name = table3.column_name;
```

***Here three tables….***

1. Student Table

| id | name | percentage | age | gender | city | cource |
|---|---|---|---|---|---|---|
| 1 | Ram Kumar | 45 | 19 | M | 1 | 1 |
| 2 | Sarita Kumari | 55 | 22 | F | 2 | 3 |
| 3 | Slaman Khan | 62 | 20 | M | 4 | 2 |
| 4 | Juhi Chawla | 47 | 18 | F | 2 | 1 |
| 5 | Anil Kapoor | 74 | 22 | M | 5 | 3 |
| 6 | Rohan Yadav | 35 | 20 | M | 3 | 2 |
| 7 | Karan Jaiswal | 85 | 24 | M | 4 | 2 |
| 8 | Seema Dubey | 55 | 22 | F | 3 | 3 |

2. City Table

| cid | cityname |
|---|---|
| 1 | Agra |
| 2 | Gorakhpur |
| 3 | Delhi |
| 4 | Bhopal |
| 5 | Noida |

3. Courses Table

| crid | cource |
|---|---|
| 1 | Btech |
| 2 | BCA |
| 3 | MBA |

Now we are join all tables by **inner join, left join** and **right join** command according to requirement.

**Example: -** Query are-

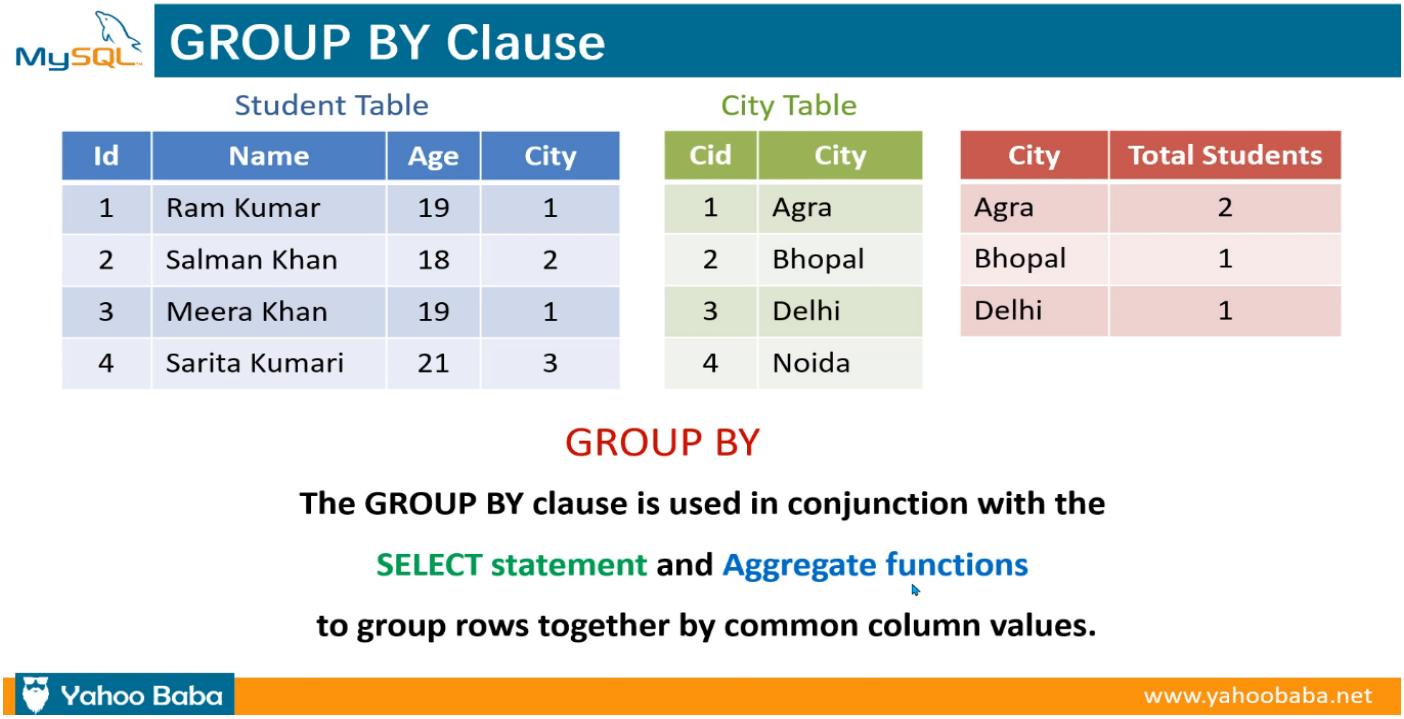
```
SELECT * FROM student s
    INNER JOIN city c
    ON s.city=c.cid
    INNER JOIN cources co
    ON s.cource=co.crid;
```

**Output:-**

| id | name | percentage | age | gender | city | cource | cid | cityname | crid | cource |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Ram Kumar | 45 | 19 | M | 1 | 1 | 1 | Agra | 1 | Btech |
| 2 | Sarita Kumari | 55 | 22 | F | 2 | 3 | 2 | Gorakhpur | 3 | MBA |
| 3 | Slaman Khan | 62 | 20 | M | 4 | 2 | 4 | Bhopal | 2 | BCA |
| 4 | Juhi Chawla | 47 | 18 | F | 2 | 1 | 2 | Gorakhpur | 1 | Btech |
| 5 | Anil Kapoor | 74 | 22 | M | 5 | 3 | 5 | Noida | 3 | MBA |
| 6 | Rohan Yadav | 35 | 20 | M | 3 | 2 | 3 | Delhi | 2 | BCA |
| 7 | Karan Jaiswal | 85 | 24 | M | 4 | 2 | 4 | Bhopal | 2 | BCA |
| 8 | Seema Dubey | 55 | 22 | F | 3 | 3 | 3 | Delhi | 3 | MBA |

# Lecture-25

## Group by and Having clause in MySQL




If I use group by then aggregate function (lecture) are compulsory.

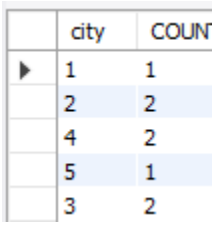
## SELECT with GROUP BY Syntax

```
SELECT columns
FROM table_name
WHERE condition
GROUP BY column_name(s);
```

**Example:-**

```
SELECT city, COUNT(city) FROM student
GROUP BY city;
```

**Output:-**



| | city | COUN |
|---|---|---|
| ▶ | 1 | 1 |
| | 2 | 2 |
| | 4 | 2 |
| | 5 | 1 |
| | 3 | 2 |

**Group by with join:-**

## SELECT with GROUP BY Syntax

```
SELECT columns
FROM table1 INNER JOIN table2
ON table1.column_name = table2.column_name
WHERE condition
GROUP BY column_name(s);
```

**Example:-**

```
SELECT c.cityname, COUNT(s.city) FROM student s
INNER JOIN city c
ON s.city=c.cid
GROUP BY city;
```

## Output:-

| | cityname | COUNT(s.city) |
|---|---|---|
| ▶ | Agra | 1 |
| | Gorakhpur | 2 |
| | Bhopal | 2 |
| | Noida | 1 |
| | Delhi | 2 |

## Example:-

```
SELECT c.cityname, COUNT(s.city) AS TOTAL FROM student s
INNER JOIN city c
ON s.city=c.cid
GROUP BY city;
```

## Output:-

| | cityname | TOTAL |
|---|---|---|
| ▶ | Agra | 1 |
| | Gorakhpur | 2 |
| | Bhopal | 2 |
| | Noida | 1 |
| | Delhi | 2 |

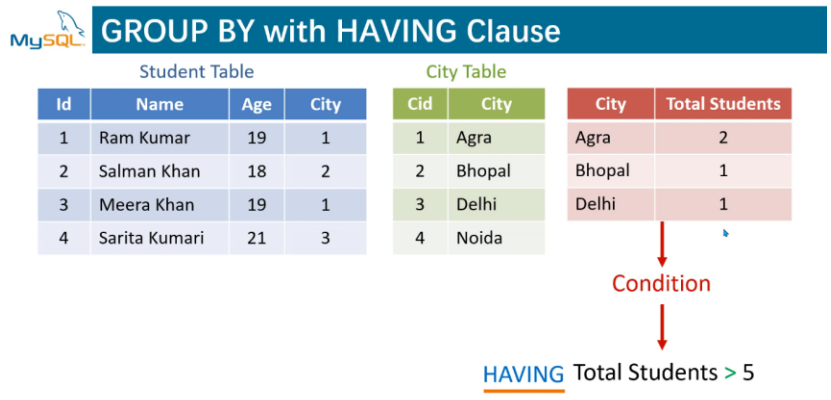
## Example:-

```
SELECT c.cityname, COUNT (s.city) AS TOTAL FROM student s
INNER JOIN city c
ON s.city=c.cid
WHERE s.age>=20
GROUP BY city
ORDER BY COUNT(s.city) ASC;
```

## Output:-

| | cityname | TOTAL |
|---|---|---|
| ▶ | Gorakhpur | 1 |
| | Noida | 1 |
| | Bhopal | 2 |
| | Delhi | 2 |

## HAVING CLAUSE:-



HAVING Clause never used without group by. Having clause is condition apply on result of group by.

```
SELECT columns

FROM table_name

GROUP BY column_name(s)

HAVING condition;
```

**Example:-**

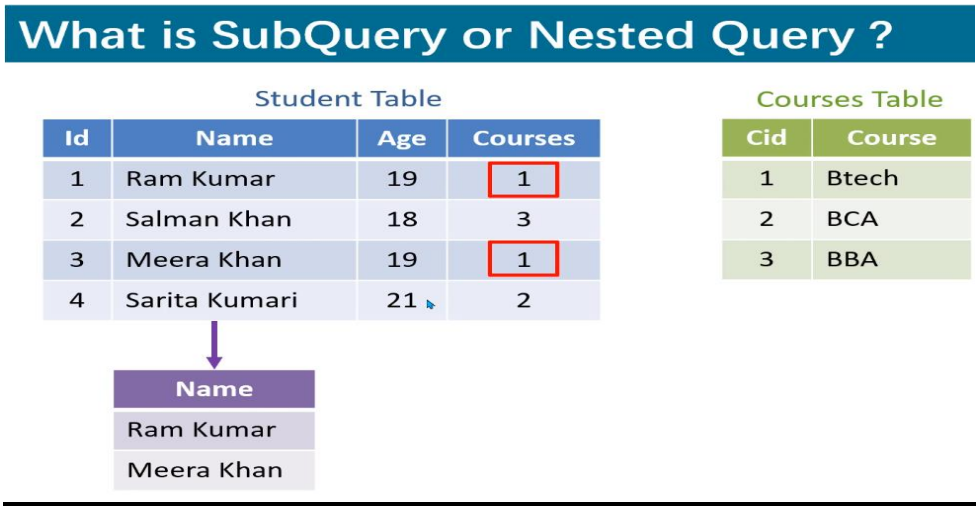
```
SELECT c.cityname, COUNT (s.city) AS TOTAL FROM student s
        INNER JOIN city c
        ON s.city=c.cid
        WHERE s.age>=20
        GROUP BY city
        HAVING COUNT(s.city)>1
        ORDER BY COUNT(s.city) ASC;
```

**Output:-**

| | cityname | TOTAL |
|---|---|---|
| ▶ | Bhopal | 2 |
| | Delhi | 2 |

## Lecture-26

### Sub-query with Exists & Not Exists in MySQL



# SELECT with SubQuery Syntax

```
SELECT columns
FROM table1
WHERE
column = (SELECT columns FROM table2 WHERE condition);
```

It is Parent query.

It is Nested(child) query.

**Example:-**

```
select name from student
where cource =(select crid from cources where cource ="Btech");
```

**Output:-**

| | name |
|---|---|
| ▶ | Ram Kumar |
| | Juhi Chawla |
| | Anil Kapoor |

**Above query is search name which student join "Btech" course.**

**If we want to search two or more value in one column.**

**Example:-**

```
select name from student
where cource in(select crid from cources where cource in("Btech","BCA"));
```

**Output:-**

| | name |
|---|---|
| ▶ | Ram Kumar |
| | Sarita Kumari |
| | Juhi Chawla |
| | Anil Kapoor |
| | Karan Jaiswal |
| | Seema Dubey |

**Exists:-**

## SELECT with EXISTS Syntax

```
SELECT columns
FROM table1
WHERE
EXISTS (SELECT columns FROM table2 WHERE condition);
```

If any Single Record Exists
Then
Parent command show results

## Not Exists:-


```
SELECT columns
FROM table1
WHERE
NOT EXISTS (SELECT columns FROM table2 WHERE condition);
```

If not any Single Record Exists
Then
Parent command show results

---

## Example:-

```
select id, name from student
where exists(select crid from cources where cource in("bba"));
```

in above query is not give any output because in cources table no any record **bba** in cource column but if I use **not exists** then it is show all record from name column.

### Example:-

```
select id, name from student
where not exists(select crid from cources where cource in("bba"));
```

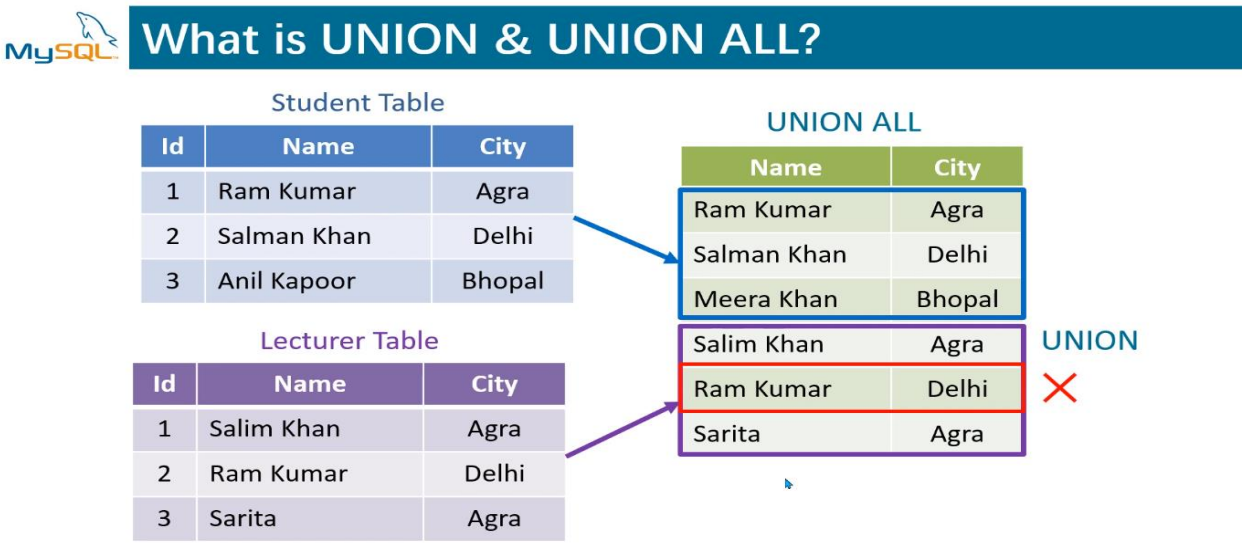
## Output:-

| id | name |
|---|---|
| 1 | Ram Kumar |
| 2 | Sarita Kumari |
| 3 | Slaman Khan |
| 4 | Juhi Chawla |
| 5 | Anil Kapoor |
| 6 | Rohan Yadav |
| 7 | Karan Jaiswal |
| 8 | Seema Dubey |

# Lecture-27

## Union All and Union

**Union all** and **Union** are used for combine the two or more similar tables. Similar table's means **Number of column** and **name of column are same in both or all tables**. **Name of table are not same.**

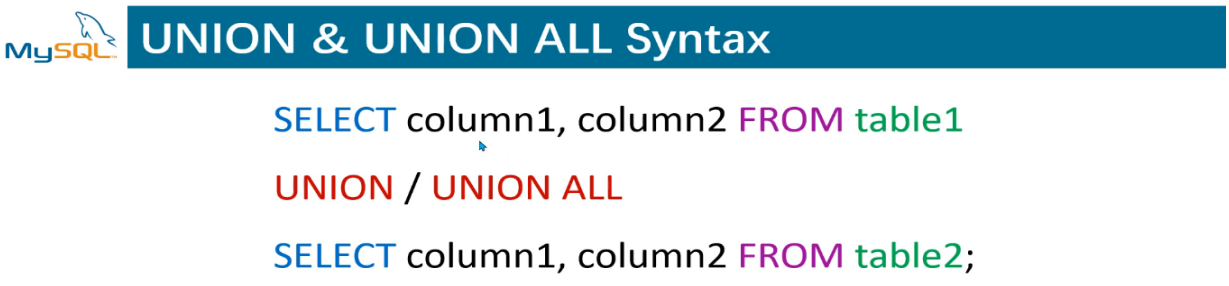



In student and lecture table, **Number of column** and **name of column (id, Name, City) are same in both tables** are same.

**Union All: -** union all show duplicate records after combine of tables.

**Union: -** union are not show duplicate records after combine of tables.

In above image, "**Ram Kumar"** is two times in both tables. **Union all** show "**Ram Kumar"** tow times but **Union** show single record of "**Ram Kumar".**




- Each SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement must also be in the same order

# Lecture-27

## If and Case clause

**If clause:-** If make conditional based statement or column then we use **if clause.**

Student Table

| Id | Name | Percentage |
|---|---|---|
| 1 | Ram Kumar | 57 |
| 2 | Salman Khan | 28 |
| 3 | Meera Khan | 81 |
| 4 | Sarita Kumari | 45 |

City Table

| Id | Name | Percentage | Result |
|---|---|---|---|
| 1 | Ram Kumar | 57 | PASS |
| 2 | Salman Khan | 28 | FAIL |
| 3 | Meera Khan | 81 | PASS |
| 4 | Sarita Kumari | 45 | PASS |

IF Clause

Percentage >= 33% ⟶ Pass

Percentage < 33% ⟶ Fail

---

## Syntax of if clause:-

```
SELECT column1, column2,
IF (Condition, TRUE Result, FALSE Result ) AS alias_name
FROM table_name;
```

**Example:-**

```
SELECT id, name, percentage,
IF(percentage>=50,"Pass","fail")AS Result
FROM student;
```

**Output:-**

| id | name | percentage | Result |
|---|---|---|---|
| 1 | Ram Kumar | 45 | fail |
| 2 | Sarita Kumari | 55 | Pass |
| 3 | Slaman Khan | 62 | Pass |
| 4 | Juhi Chawla | 47 | fail |
| 5 | Anil Kapoor | 74 | Pass |
| 6 | Rohan Yadav | 35 | fail |
| 7 | Karan Jaiswal | 85 | Pass |
| 8 | Seema Dubey | 55 | Pass |

**Case clause:-** if I want use multiple condition then we use case clause.

## What is CASE Clause ?

Student Table

| Id | Name | Percentage |
|---|---|---|
| 1 | Ram Kumar | 57 |
| 2 | Salman Khan | 28 |
| 3 | Meera Khan | 81 |
| 4 | Sarita Kumari | 45 |

City Table

| Id | Name | Percentage | Grade |
|---|---|---|---|
| 1 | Ram Kumar | 57 | IInd Division |
| 2 | Salman Khan | 28 | FAIL |
| 3 | Meera Khan | 81 | Merit |
| 4 | Sarita Kumari | 43 | IIIrd Division |

Percentage >= 80 AND Percentage <= 100 → Merit

Percentage >= 60 AND Percentage < 80 → Ist Division

Percentage >= 45 AND Percentage < 60 → IInd Division

Percentage >= 33 AND Percentage < 45 → IIIrd Division

Percentage < 33 → FAIL

Yahoo Baba www.yahoobaba.net

## SELECT with CASE Clause Syntax

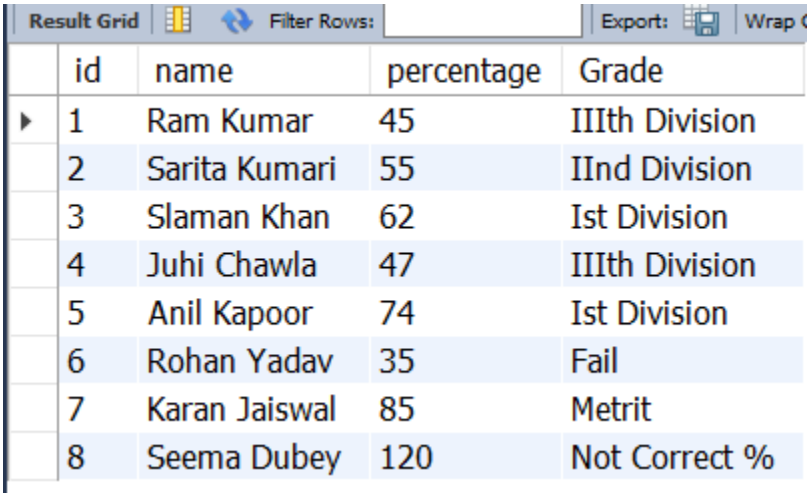
```
SELECT column1, column2,
CASE
    WHEN Condition1 THEN result1
    WHEN Condition2 THEN result2
    WHEN Condition3 THEN result3
    ELSE result alias_name
END AS alias_name
FROM table_name;
```

Yahoo Baba www.yahoobaba.net

## Example:-

```
SELECT id, name, percentage,
    CASE
      WHEN percentage >= 80 AND percentage <= 100 THEN "Metrit"
      WHEN percentage >= 60 AND percentage < 80 THEN "Ist Division"
      WHEN percentage >= 50 AND percentage < 60 THEN "IInd Division"
      WHEN percentage >= 40 AND percentage < 50 THEN "IIIth Division"
      WHEN percentage <=40 THEN "Fail"
      ELSE "Not Correct %"
   END AS Grade
FROM student;
```

## Output:-



Result Grid | Filter Rows: | Export: | Wrap

| id | name | percentage | Grade |
|---|---|---|---|
| 1 | Ram Kumar | 45 | IIIth Division |
| 2 | Sarita Kumari | 55 | IInd Division |
| 3 | Slaman Khan | 62 | Ist Division |
| 4 | Juhi Chawla | 47 | IIIth Division |
| 5 | Anil Kapoor | 74 | Ist Division |
| 6 | Rohan Yadav | 35 | Fail |
| 7 | Karan Jaiswal | 85 | Metrit |
| 8 | Seema Dubey | 120 | Not Correct % |

## How to use **Case Clause** with **Update** command

## Example:-

```
UPDATE student SET
        percentage=(
                CASE id
                        WHEN 4 THEN 52
                        WHEN 8 THEN 76
                END
                )
    WHERE id IN(4, 8);
```

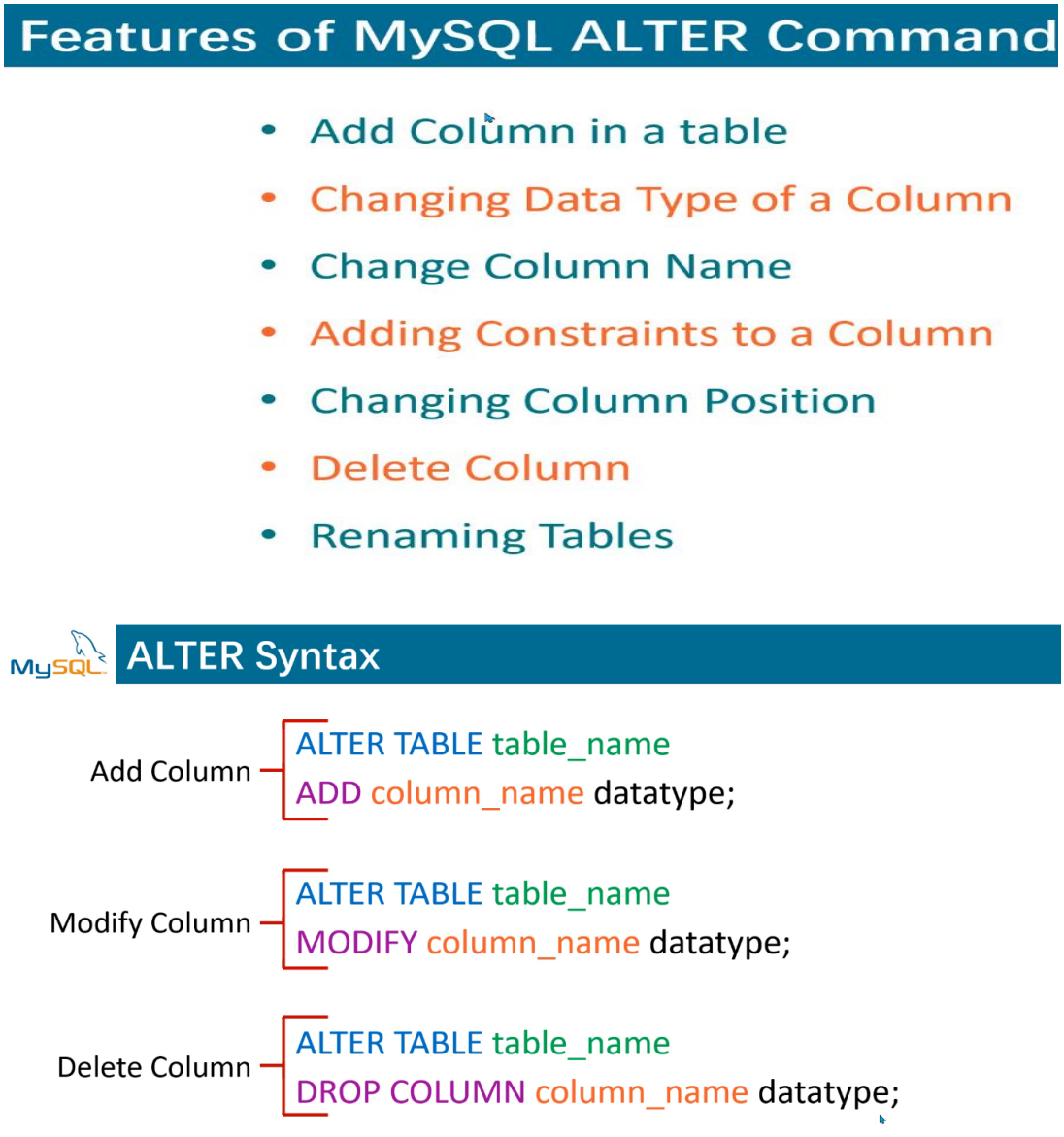
**OR**

```
UPDATE student SET
        percentage=(
                CASE
                        WHEN id= 4 THEN 52
                        WHEN id= 8 THEN 76
                END
                )
    WHERE id IN(4, 8);
```

# Lecture-35

## Alter command

Alter command used to change something like.




Rename Column

```
ALTER TABLE table_name
CHANGE column_name New_name datatype;
```

Rename Table

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

**Example: -** add new column in student table.

```
ALTER TABLE student
ADD Email VARCHAR(25);
```

**Output:-**

| id | name | percentage | age | gender | city | cource | Email |
|---|---|---|---|---|---|---|---|
| 1 | Ram Kumar | 45 | 19 | M | 1 | 1 | NULL |
| 2 | Sarita Kumari | 55 | 22 | F | 2 | 2 | NULL |
| 3 | Slaman Khan | 62 | 20 | M | 4 | 2 | NULL |
| 4 | Juhi Chawla | 52 | 18 | F | 2 | 3 | NULL |
| 5 | Anil Kapoor | 74 | 22 | M | 5 | 2 | NULL |
| 6 | Rohan Yadav | 35 | 20 | M | 3 | 1 | NULL |
| 7 | Karan Jaiswal | 85 | 24 | M | 4 | 2 | NULL |
| 8 | Seema Dubey | 76 | 22 | F | 3 | 4 | NULL |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

**Example: -** Change position Email column after name column in student table.

```
ALTER TABLE student
MODIFY Email VARCHAR(25) AFTER name;
```

**Output:-**

Result Grid | Filter Rows: | Edit: | Export/Import:

| id | name | Email | percentage | age | gender | city | cource |
|---|---|---|---|---|---|---|---|
| 1 | Ram Kumar | NULL | 45 | 19 | M | 1 | 1 |
| 2 | Sarita Kumari | NULL | 55 | 22 | F | 2 | 2 |
| 3 | Slaman Khan | NULL | 62 | 20 | M | 4 | 2 |
| 4 | Juhi Chawla | NULL | 52 | 18 | F | 2 | 3 |
| 5 | Anil Kapoor | NULL | 74 | 22 | M | 5 | 2 |
| 6 | Rohan Yadav | NULL | 35 | 20 | M | 3 | 1 |
| 7 | Karan Jaiswal | NULL | 85 | 24 | M | 4 | 2 |
| 8 | Seema Dubey | NULL | 76 | 22 | F | 3 | 4 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

**Example: -** Change data type of Email column in student table.

```
ALTER TABLE student
MODIFY Email INT(10) ;
```

**Output:-** this query change data type of email column varchar into int.

**Example: -** provide some constraint (Unique) to Email column in student table.

```
ALTER TABLE student
ADD UNIQUE(Email);

ALTER TABLE student
ADD PRIMARY(Email);
```

**Output:-** this query set Unique constraint to the email column in student table.

**Example: -** Change column name of Email column into Email_id in student table.

```
ALTER TABLE student
CHANGE Email Email_id  VARCHAR(25);
```

**Output:-**

| id | name | Email_id | percentage | age | gender | city | cource |
|---|---|---|---|---|---|---|---|
| 1 | Ram Kumar | NULL | 45 | 19 | M | 1 | 1 |
| 2 | Sarita Kumari | NULL | 55 | 22 | F | 2 | 2 |
| 3 | Slaman Khan | NULL | 62 | 20 | M | 4 | 2 |
| 4 | Juhi Chawla | NULL | 52 | 18 | F | 2 | 3 |
| 5 | Anil Kapoor | NULL | 74 | 22 | M | 5 | 2 |
| 6 | Rohan Yadav | NULL | 35 | 20 | M | 3 | 1 |
| 7 | Karan Jaiswal | NULL | 85 | 24 | M | 4 | 2 |
| 8 | Seema Dubey | NULL | 76 | 22 | F | 3 | 4 |

**Example: -** Delete column in student table.

```
ALTER TABLE student
DROP COLUMN Email_id;
```

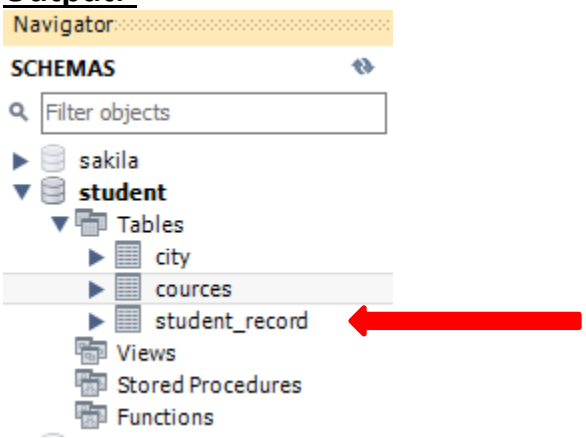
**Output:-**

| id | name | percentage | age | gender | city | cou |
|---|---|---|---|---|---|---|
| 1 | Ram Kumar | 45 | 19 | M | 1 | 1 |
| 2 | Sarita Kumari | 55 | 22 | F | 2 | 2 |
| 3 | Slaman Khan | 62 | 20 | M | 4 | 2 |
| 4 | Juhi Chawla | 52 | 18 | F | 2 | 3 |
| 5 | Anil Kapoor | 74 | 22 | M | 5 | 2 |
| 6 | Rohan Yadav | 35 | 20 | M | 3 | 1 |
| 7 | Karan Jaiswal | 85 | 24 | M | 4 | 2 |
| 8 | Seema Dubey | 76 | 22 | F | 3 | 4 |

**Example: -** If I want rename the table name.(student to student_record)

```
ALTER TABLE student
RENAME student_record;
```

**Output:-**



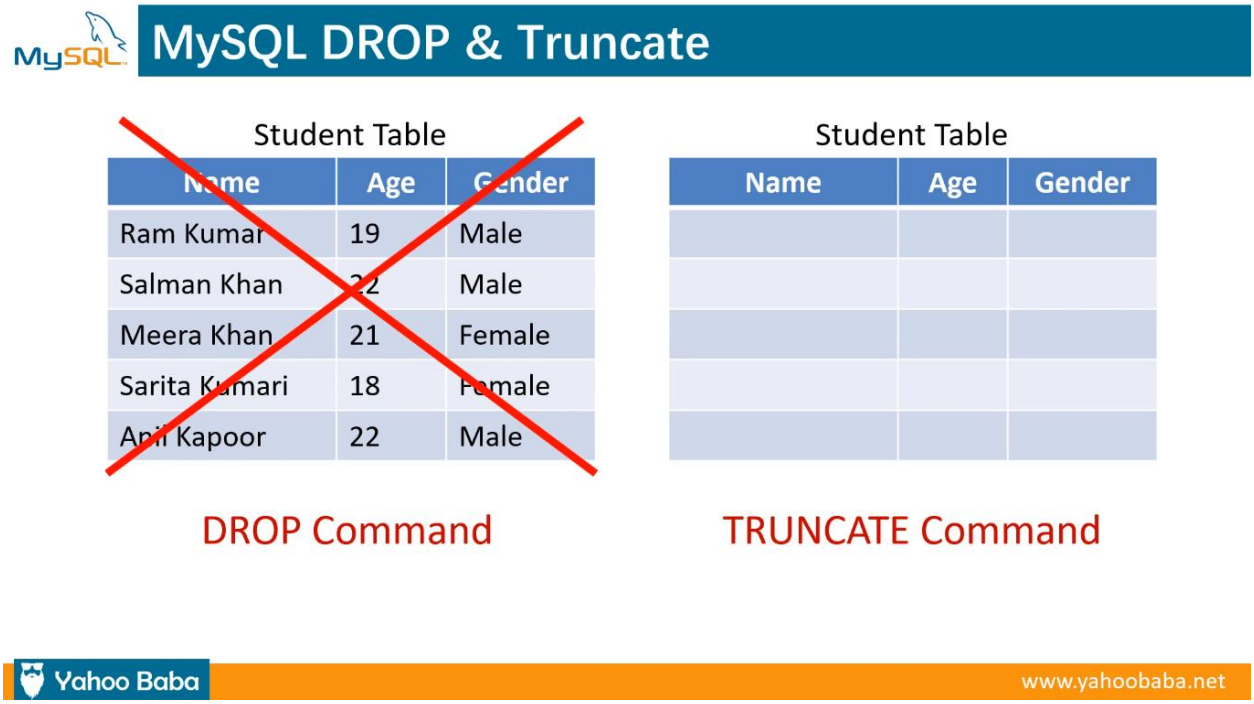
Note:- if AUTO_INCREMENT column are start from previous value. But I want to alter table.

**Example: -**

```
ALTER TABLE student
AUTO_INCREMENT = GIVE STARTING VALUE
```

# Lecture-36

## Drop and Truncate table




**Drop:-** Drop command is delete whole table (table structure, all data or record).

**Syntax of Drop command:** DROP TABLE table_name;

**Truncate:-** Truncate command is not delete whole table. It only delete all data or record of the table and table structure are maintain.

**Syntax of Drop command:** TRUNCATE TABLE table_name;

# Lecture-37

## View in MySQL

It is use for create the view. Means if I am writing more complex and line of query is more and use many times. So we are not writing same query again. It is reduce the time for code the query.

Note:- the name of view is not matched any other table name.

**Syntax of View:-**

```
CREATE VIEW view_name
AS your query;
```

**Example: -**

```
CREATE VIEW student_view
AS
SELECT * FROM student s
INNER JOIN city c
ON s.city=c.cid;
```

**Output:-** this is create student_view view.

If I want to select view then we use select query command like.

```
SELECT * FROM student_view;
```

or

```
SELECT id, name, cityname From student_view;
```

**Output:-**

Result Grid | Filter Rows:

| id | name | cityname |
|---|---|---|
| 1 | Ram Kumar | Agra |
| 2 | Sarita Kumari | Gorakhpur |
| 3 | Slaman Khan | Bhopal |
| 4 | Juhi Chawla | Gorakhpur |
| 5 | Anil Kapoor | Noida |
| 6 | Rohan Yadav | Delhi |
| 7 | Karan Jaiswal | Bhopal |
| 8 | Seema Dubey | Delhi |

## How to rename the view in MySQL

**Syntax:-** **RENAME TABLE old_name_of_view**
**TO new_name_of_view;**

**Example:-** **RENAME TABLE** student_view
**TO** student_data**;**

**Output:-** **i**t change the name of view.

## How to delete the view in MySQL

**Syntax:-** **DROP VIEW name_of_view;**

**Example:-** **DROP VIEW** student_data**;**

**Output:-** **i**t delete the view.

# Lecture-36

# Index in MySQL

Index command are used for fast searching.


```
CREATE INDEX index_name
ON table_name(column1, column2, column3, .... );


DROP INDEX index_name
ON table_name;
```

## Guidelines of Index :

- Automatically creates the indexes for PRIMARY KEY and UNIQUE columns.
- Index columns that you frequently use to retrieve the data.
- Index columns that are used for joins to improve join performance.
- Avoid columns that contain too many NULL values.
- Small tables do not require indexes.



Example:-

```
CREATE INDEX age2 ON student(age);

SHOW INDEX FROM student;

DROP INDEX age2 ON student;
```