SQL FOR COMPUTER SCIENCE

DATABASE MANAGEMENT

FOR

CLASS – XII

KAPIL SEHGAL

KAPIL SEHGAL (P.G.T. (CS) K.V. NO2 Delhi Cantt,

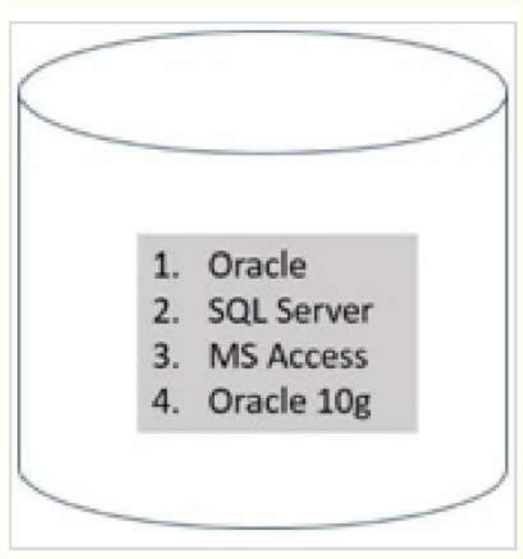
WHAT IS DATABASE

A collected information which is in an organized form for easier access, management, and various updating is known as a database.

Ex.

Containers having a huge amount of data are known as databases, for example, a public library stores books. Databases are computer structures that save, organize, protect, and deliver data.

Any system that manages databases is called a **database management system**, or DBMS. The typical diagram representation for a database is a cylinder.



1. Oracle
2. SQL Server
3. MS Access
4. Oracle 10g

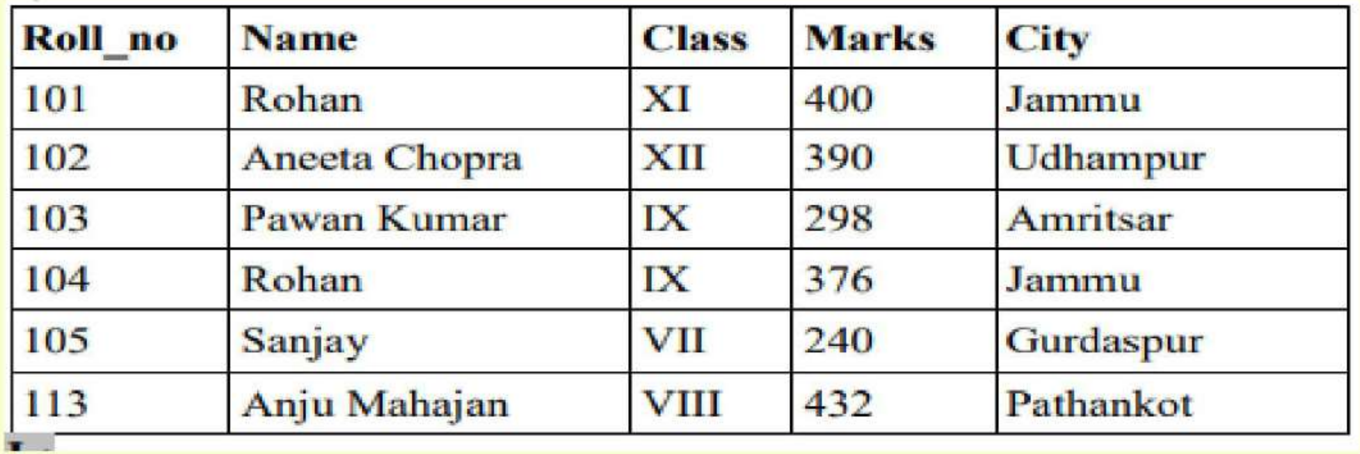
DATABASE, TABLES, ROWS AND COLUMNS

Database

Database is a collection, or a set of tables. Each table has a formalized repeating list of data about one specific information. For example a table for customers, students, orders, products and so on. Visually, it’s often shown like a spread sheet.

Tables

The table is the most basic building of a database. It’s the place where you will put your data, define their data type, and also their relationship with the other tables. It consists of rows, and columns.



Roll_no	Name	Class	Marks	City
101	Rohan	XI	400	Jammu
102	Aneeta Chopra	XII	390	Udhampur
103	Pawan Kumar	IX	298	Amritsar
104	Rohan	IX	376	Jammu
105	Sanjay	VII	240	Gurdaspur
113	Anju Mahajan	VIII	432	Pathankot

ROW & COLUMNS

Within each table, every single row represents one single student, customer, order, or employee. But each of these rows is not free form. You must apply structure to this data.

So, you must say what every row is made of, and you do this by defining the columns in that table. And each column describes one piece of data. It gives it a name like name, id, email, date of birth, and a type, perhaps, a text, or a date, or a number.

Now, every row must follow that same structure, following that same format. It's not allowed to deviate from the way that the columns are set up. And by defining these columns, we're imposing rules on the data, and the DBMS won't let us break them.

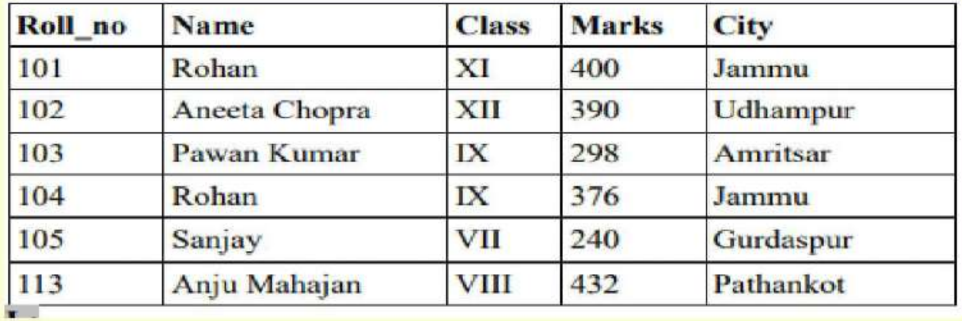
In a nutshell

Columns define what's the data that should be in the table, while the rows hold the actual values that you are going to *retrieve*, *insert*, *update*, and *delete*.

RELATIONAL DATA MODEL

Concept of Domain

A **domain** is the original sets of atomic values used to model **data**. By atomic value, we mean that each value in the **domain** is indivisible as far as the **relational** model is concerned. For example: The **domain** of Marital Status has a set of possibilities: Married, Single, Divorced.



Roll_no	Name	Class	Marks	City
101	Rohan	XI	400	Jammu
102	Aneeta Chopra	XII	390	Udhampur
103	Pawan Kumar	IX	298	Amritsar
104	Rohan	IX	376	Jammu
105	Sanjay	VII	240	Gurdaspur
113	Anju Mahajan	VIII	432	Pathankot

Relation

Relation (collection of rows and columns generally refers to an active entity (not be duplicate) on which we can perform various operations

Schema

The **database schema** of a **database** is its structure described in a formal language supported by the **database** management system (DBMS). The **term** "**schema**" refers to the organization of data as a blueprint of how the **database** is constructed (divided into **database** tables in the case of relational **databases**).

Table

Table (collection of rows and columns generally refers to an passive entity (may be duplicate) on which we can perform various operations

KAPIL SEHGAL

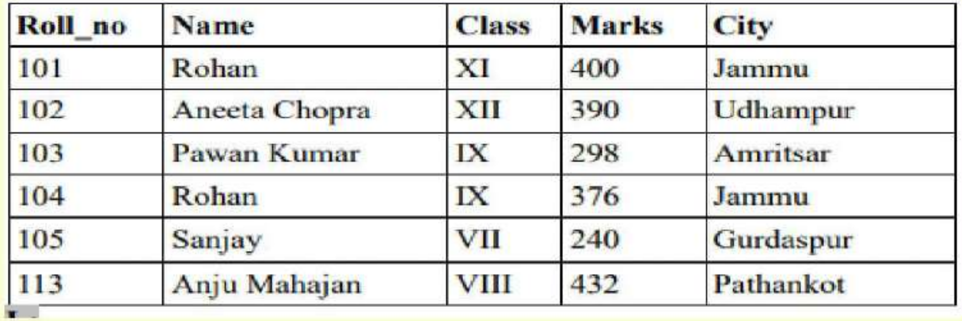
RELATIONAL DATA MODEL

Tuple

A row in a relation is called a tuple.

Attributes

A column in a relation is called an attribute. It is also termed as field or data item



Roll_no	Name	Class	Marks	City
101	Rohan	XI	400	Jammu
102	Aneeta Chopra	XII	390	Udhampur
103	Pawan Kumar	IX	298	Amritsar
104	Rohan	IX	376	Jammu
105	Sanjay	VII	240	Gurdaspur
113	Anju Mahajan	VIII	432	Pathankot

Candidate Key

Set of all attributes which can serve as a primary key in a relation. i.e. A table / relation has more than one columns which all have features to become a primary key. Then all such columns are known as candidate key.

Primary Key

Primary key is a key that can uniquely identifies the records/tuples in a relation. There are following feature of Primary key

(1) It cannot be duplicate in entire column

(2) It cannot be NULL (absent of value)

(3) A table / relation cannot have more than one primary key.

Alternate Key

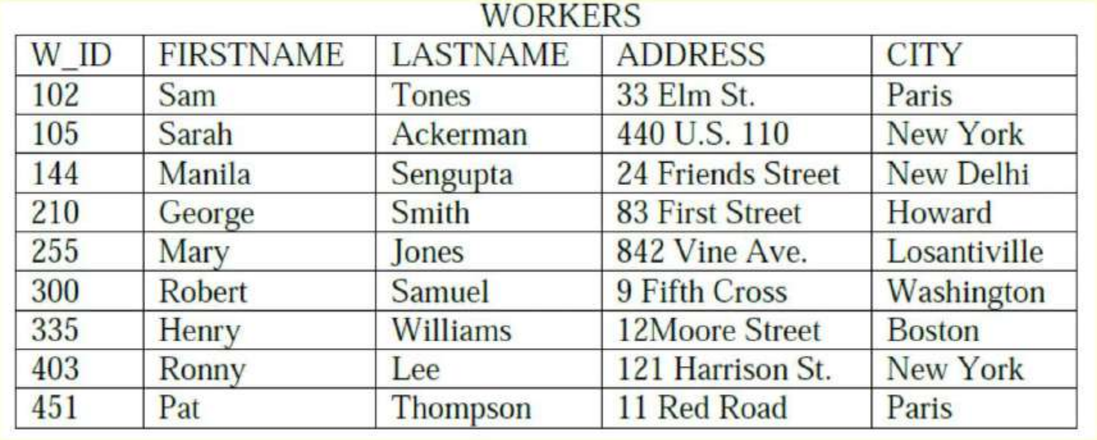
All the candidate keys other than the primary keys of a relation are alternate keys for a relation

Key

Key is the name of column.

KAPIL SEHGAL

Foreign Key



WORKERS

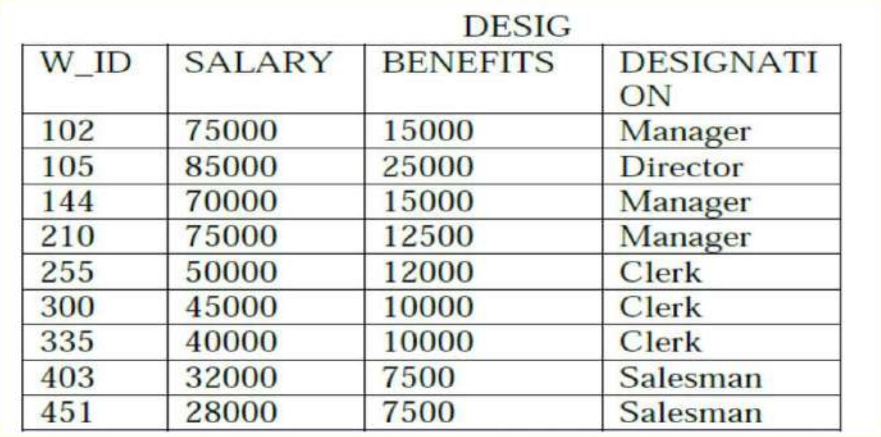
W_ID	FIRSTNAME	LASTNAME	ADDRESS	CITY
102	Sam	Tones	33 Elm St.	Paris
105	Sarah	Ackerman	440 U.S. 110	New York
144	Manila	Sengupta	24 Friends Street	New Delhi
210	George	Smith	83 First Street	Howard
255	Mary	Jones	842 Vine Ave.	Losantiville
300	Robert	Samuel	9 Fifth Cross	Washington
335	Henry	Williams	12Moore Street	Boston
403	Ronny	Lee	121 Harrison St.	New York
451	Pat	Thompson	11 Red Road	Paris

Definition: Foreign keys is the column of a table that points to the **primary key** of another table. They act as a cross-reference between tables.

Definition: Foreign keys is the column of a table that takes the value from primary key from other table from the same database.

KAPIL SEHGAL

RELATIONAL DATA MODEL



DESIG

W_ID	SALARY	BENEFITS	DESIGNATION
102	75000	15000	Manager
105	85000	25000	Director
144	70000	15000	Manager
210	75000	12500	Manager
255	50000	12000	Clerk
300	45000	10000	Clerk
335	40000	10000	Clerk
403	32000	7500	Salesman
451	28000	7500	Salesman

The **table** containing the **foreign key** is called the child **table**, and the **table** containing the candidate **key** is called the referenced or parent **table**.

Here W_ID is the Primary key in Workers Table and foreign key in Desig Table

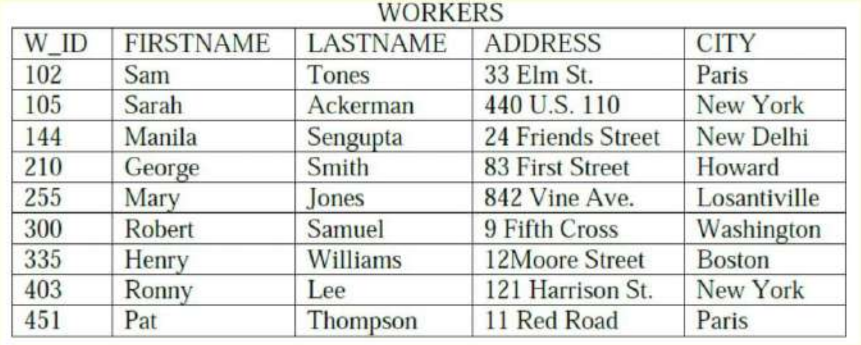
RELATIONAL DATA MODEL

Degree

Definition: Degree of the table / Relation means the number of Column.

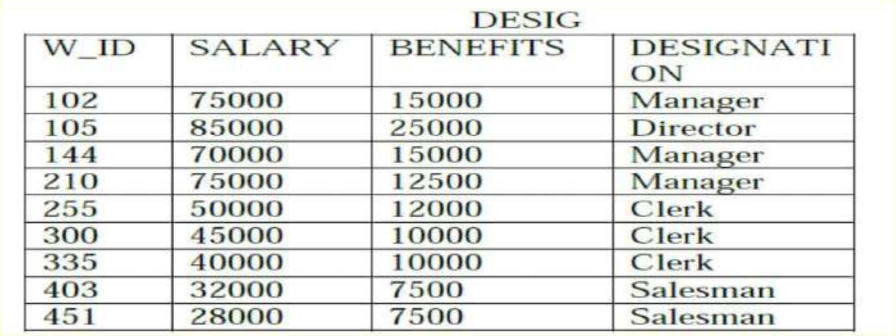
Cardinality

Definition: Cardinality of the table / Relation means the number of Rows.



WORKERS

W_ID	FIRSTNAME	LASTNAME	ADDRESS	CITY
102	Sam	Tones	33 Elm St.	Paris
105	Sarah	Ackerman	440 U.S. 110	New York
144	Manila	Sengupta	24 Friends Street	New Delhi
210	George	Smith	83 First Street	Howard
255	Mary	Jones	842 Vine Ave.	Losantiville
300	Robert	Samuel	9 Fifth Cross	Washington
335	Henry	Williams	12Moore Street	Boston
403	Ronny	Lee	121 Harrison St.	New York
451	Pat	Thompson	11 Red Road	Paris



DESIG

W_ID	SALARY	BENEFITS	DESIGNATION
102	75000	15000	Manager
105	85000	25000	Director
144	70000	15000	Manager
210	75000	12500	Manager
255	50000	12000	Clerk
300	45000	10000	Clerk
335	40000	10000	Clerk
403	32000	7500	Salesman
451	28000	7500	Salesman

Ex. Degree : 5
Cardinality : 9

Ex. Degree : 4
Cardinality : 9

STRUCTURED QUERY LANGUAGE

What is Structured Query Language

SQL is a non procedural language that is used to create, manipulate and process the database relations

Characteristics of SQL

1. It is very easy to learn and use.
2. Large volume of databases can be handled quite easily.
3. It is non procedural language. It means that we do not need to specify the procedures to accomplish a task but just to give a command to perform the activity.
4. SQL can be linked to most of other high level languages that makes it first choice for the database programme

Need of SQL

It is widely used in the Business Intelligence tool.
Data Manipulation and data testing are done through SQL.
Data Science tools depend highly on SQL. Big data tools such as Spark, Impala are dependant on SQL.
It is one of the demanding industrial skills.

STRUCTURED QUERY LANGUAGE

Processing Capabilities of SQL

The following are the processing capabilities of SQL

1. Data Definition Language (DDL)

Data Definition Language (DDL DDL contains commands that are used to create the tables, databases, indexes, views, sequences etc. e.g.: Create table, create view, create index. Other than create command ALTER and DROP is also DDL Commands

2. Data Manipulation Language (DML)

DML contains command that can be used to manipulate the data base objects and to query the databases for information retrieval. e.g. Select, Insert, Delete, Update etc.

3. Data Control Language:

This language is used for controlling the access to the data. Various commands like GRANT, REVOKE etc. are available in DCL

4. Transaction Control Language (TCL)

TCL include commands to control the transactions in a data base system. The commonly used commands in TCL are COMMIT, ROLLBACK etc. Data types of SQL

Data Types

Just like any other programming language, the facility of defining data of various types is available in SQL also. Following are the most common data types of SQL.

1 Int : Used to store a numeric value in a field/column. It is for integer value.

2. Float : Used to store a numeric value in a field/column. It is for real or decimal value

3 Char : Used to store a fixed length characters or fixed length string

4 Varchar : Used to store a variable length characters or variable length string

5 Date ; Used to store a date type value

6 Boolean : Used to store a Boolean type value (i.e. TRUE / FALSE)

STRUCTURED QUERY LANGUAGE

Data Definition Language (DDL)

Creating Database

```
Sql> Create database <databaseName> ;
Sql> Create database  XII ;
```

Here a database named XII will create in your current user id (By default is “root”)

How to make active your database

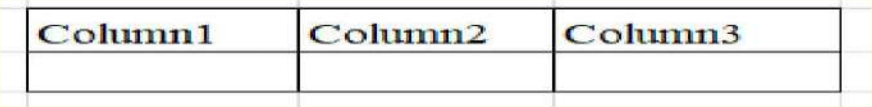
```
Sql> Use <databaseName>;
Sql> Use XII ;
```

Here through the USE command we can activate the database, XII is the database name.

Creating Table

Syntax

CREATE TABLE table_name (column1 datatype, column2 datatype, column3 datatype, ...);

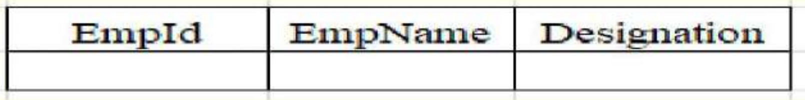


Column1	Column2	Column3

Here original sequence will be Column1, Column2 and Column3 that will never change.

Example

CREATE TABLE Emp (EmpId int, EmpName varchar(30), Designation Varchar(20));



EmpId	EmpName	Designation

Database Constraints

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

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

NOT NULL - Ensures that a column cannot have a NULL value (Ex. **Name Varchar(20) Not Null**)

UNIQUE - Ensures that all values in a column are different (Ex. **Rollno int Unique**)

PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table (Ex. **Admno int Primary Key)**

FOREIGN KEY - Uniquely identifies a row/record in another table

CHECK - Ensures that all values in a column satisfies a specific condition (Ex. **Marks int Check(Marks>0)**

DEFAULT - Sets a default value for a column when no value is specified (Ex. **City Varchar(20) default ‘New Delhi’)**

Database Constraints

Similarity and Difference between UNIQUE and Primary Key.

Similarity

Both Primary Key and Unique cannot be duplicate in entire column

Difference

(1) Primary Key cannot be NULL while Unique can be Null
(2) Primary Key can not more than one in a table but Unique constraints can be more than one in a table.

Ex.

Primary Key : Admission_No, Emp_Id, or any other types of id.
Unique : Phone_No, Vehicle_No.

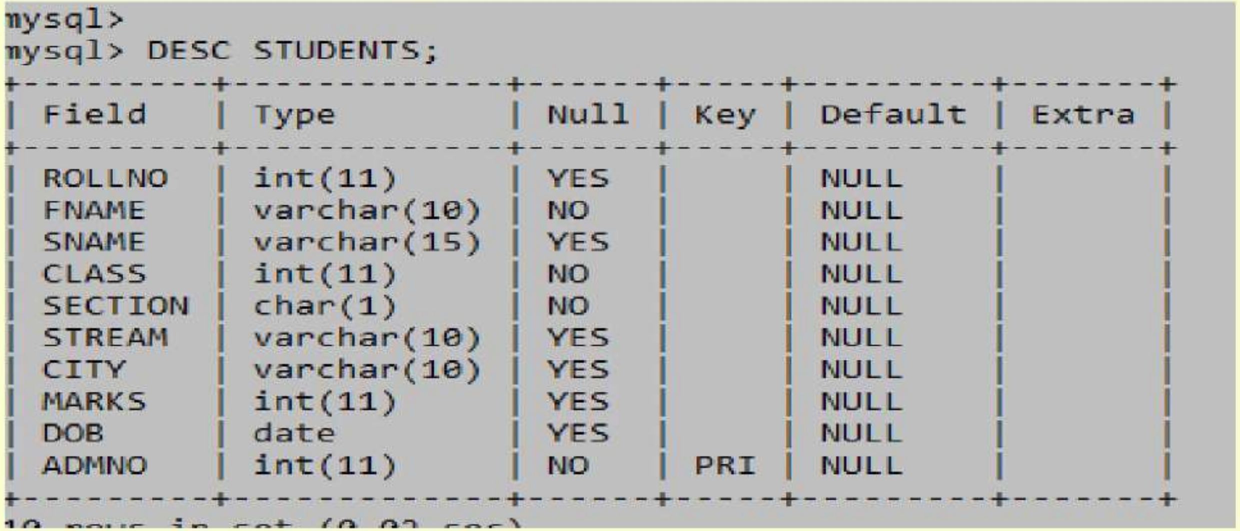
Creating Table D.D.L. STRUCTURED QUERY LANGUAGE

Rollno	Fname	Sname	Class	Section	Stream	City	Marks	DOB	Admno

CREATE TABLE STUDENTS(ROLLNO INT,FNAME VARCHAR(10) NOT NULL, SNAME VARCHAR(15), CLASS INT NOT NULL, SECTION CHAR(1) NOT NULL, STREAM VARCHAR(10),CITY VARCHAR(10), MARKS INT CHECK(MARKS>0),DOB DATE, ADMNO INT PRIMARY KEY);

View the structure of the table

Sql > Desc Students;



```
mysql>
mysql> DESC STUDENTS;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| ROLLNO   | int(11)     | YES  |     | NULL    |       |
| FNAME    | varchar(10) | NO   |     | NULL    |       |
| SNAME    | varchar(15) | YES  |     | NULL    |       |
| CLASS    | int(11)     | NO   |     | NULL    |       |
| SECTION  | char(1)     | NO   |     | NULL    |       |
| STREAM   | varchar(10) | YES  |     | NULL    |       |
| CITY     | varchar(10) | YES  |     | NULL    |       |
| MARKS    | int(11)     | YES  |     | NULL    |       |
| DOB      | date        | YES  |     | NULL    |       |
| ADMNO    | int(11)     | NO   | PRI | NULL    |       |
+----------+-------------+------+-----+---------+-------+
10 rows in set (0.02 sec)
```

KAPIL SEHGAL

Inserting Row into Table D.M.L. STRUCTURED QUERY LANGUAGE

Inserting Row into Table with All Columns

INSERT INTO TABLE-Name Value (value of column1, value of column 2, value of column3…);

Ex.

Insert Into students values (101,’Ram’,’Sharma’,12,’A’,’Science’,’New Delhi’,494,’2001-04-15’,20795);

Insert Into students values (102,’Govind’,’Shukla’,12,’B’,’Science’,’New Delhi’,475,’2002-08-03’,25437);

Insert Into students values (102,’Shyam’,’Tiwari’,12,’C’,’Commerce’,’Bhopal’,485,’2002-08-25’,25737);

Rollno	Fname	Sname	Class	Section	Stream	City	Marks	DOB	Admno
101	Ram	Sharma	12	A	Science	New Delhi	494	2001-04-15	20795
102	Govind	Shukla	12	B	Science	New Delhi	475	2002-08-03	25437
103	Shyam	Tiwari	12	C	Commerce	Bhopal	485	2002-08-25	25737

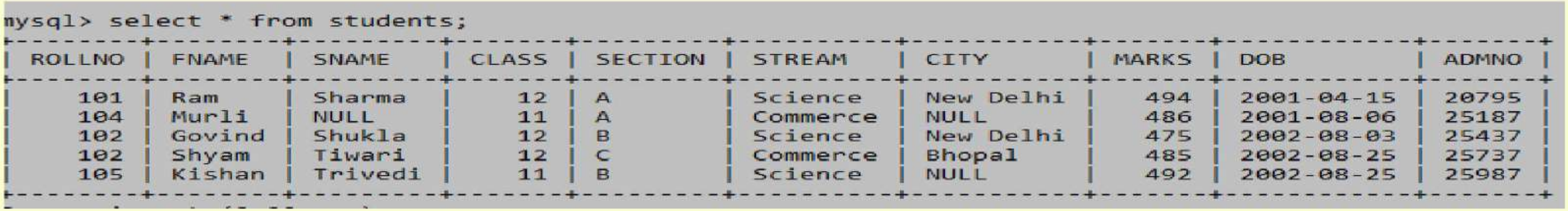
Inserting Row into Table with Selected Columns

INSERT INTO TABLE-Name ColumnsName (column1, column2,…) values (value1,value2,…..));

Ex.

Insert Into students (Rollno,Fname,Class,Section,Stream,Marks,Dob,Admno) Values (104,'Murli',11,'A','Commerce',486,'2001-08-06',25187);

Insert Into students (Rollno,Fname,Sname,Class,Section,Stream,Marks,Dob,Admno) Values (105,'Kishan', 'Trivedi',11,'B','Science',492,'2002-08-25',25987);



```
mysql> select * from students;
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME   | CLASS | SECTION | STREAM   | CITY      | MARKS | DOB        | ADMNO |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
|    101 | Ram    | Sharma  |    12 | A       | Science  | New Delhi |   494 | 2001-04-15 | 20795 |
|    104 | Murli  | NULL    |    11 | A       | Commerce | NULL      |   486 | 2001-08-06 | 25187 |
|    102 | Govind | Shukla  |    12 | B       | Science  | New Delhi |   475 | 2002-08-03 | 25437 |
|    102 | Shyam  | Tiwari  |    12 | C       | Commerce | Bhopal    |   485 | 2002-08-25 | 25737 |
|    105 | Kishan | Trivedi |    11 | B       | Science  | NULL      |   492 | 2002-08-25 | 25987 |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
```

Fetching data from table **SELECT** **STRUCTURED QUERY LANGUAGE**

Select Command for all attribute with from clause

Syntax.

Select * from TableName ;

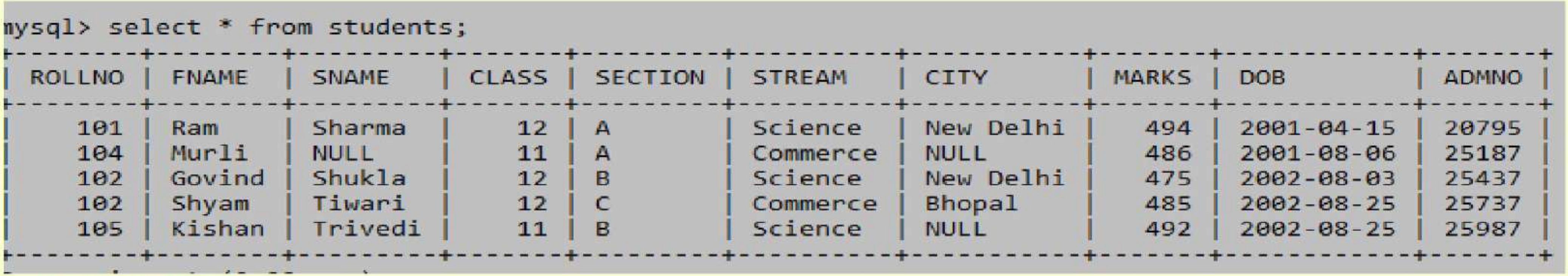
Here " ;" Semicolon represent the termination of SQL command. "From" is a clause and "Select" is a command.

Here "*" represent "All Columns" of table

Ex.

Mysql>Select * from Students;

Here our table students have 9 Columns so all columns is displaying here.



```
mysql> select * from students;
+--------+--------+---------+-------+---------+----------+-------------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME   | CLASS | SECTION | STREAM   | CITY        | MARKS | DOB        | ADMNO |
+--------+--------+---------+-------+---------+----------+-------------+-------+------------+-------+
|    101 | Ram    | Sharma  |    12 | A       | Science  | New Delhi   |   494 | 2001-04-15 | 20795 |
|    104 | Murli  | NULL    |    11 | A       | Commerce | NULL        |   486 | 2001-08-06 | 25187 |
|    102 | Govind | Shukla  |    12 | B       | Science  | New Delhi   |   475 | 2002-08-03 | 25437 |
|    102 | Shyam  | Tiwari  |    12 | C       | Commerce | Bhopal      |   485 | 2002-08-25 | 25737 |
|    105 | Kishan | Trivedi |    11 | B       | Science  | NULL        |   492 | 2002-08-25 | 25987 |
+--------+--------+---------+-------+---------+----------+-------------+-------+------------+-------+
```

KAPIL SEHGAL

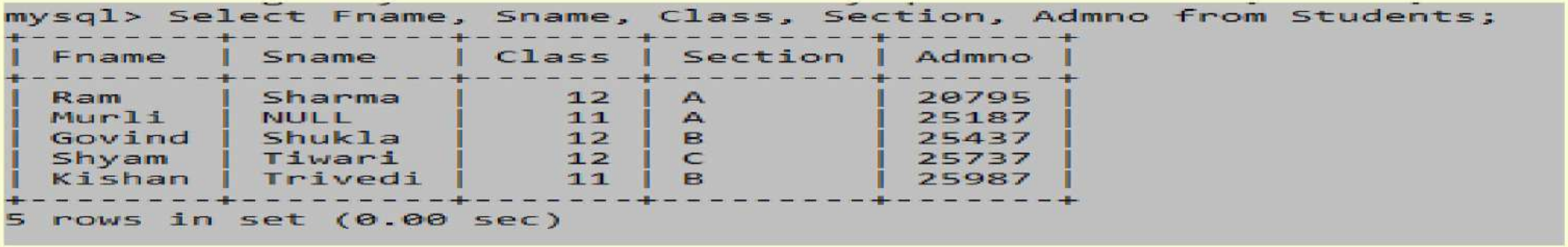
Fetching data from table SELECT STRUCTURED QUERY LANGUAGE

Select Command for selected attributes(Columns)

Syntax.

Select <list of column> Here List of Attributes (Columns) must be separated with comma from TableName ;

Ex. **Mysql>Select Fname, Sname, Class, Section, Admno from Students;**



```
mysql> Select Fname, Sname, Class, Section, Admno from Students;
+----------+-----------+---------+-----------+---------+
| Fname    | Sname     | Class   | Section   | Admno   |
+----------+-----------+---------+-----------+---------+
| Ram      | Sharma    |      12 | A         | 20795   |
| Murli    | NULL      |      11 | A         | 25187   |
| Govind   | Shukla    |      12 | B         | 25437   |
| Shyam    | Tiwari    |      12 | C         | 25737   |
| Kishan   | Trivedi   |      11 | B         | 25987   |
+----------+-----------+---------+-----------+---------+
5 rows in set (0.00 sec)
```

Fetching data from table **WHERE.** **STRUCTURED QUERY LANGUAGE**

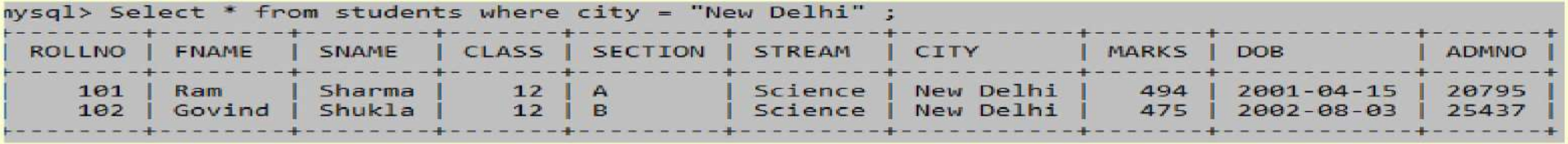
Select Command for selected tuple (Rows) using "WHERE" Clause

Syntax.

**Select <list of column> / *
from TableName
Where <Condition> ;**

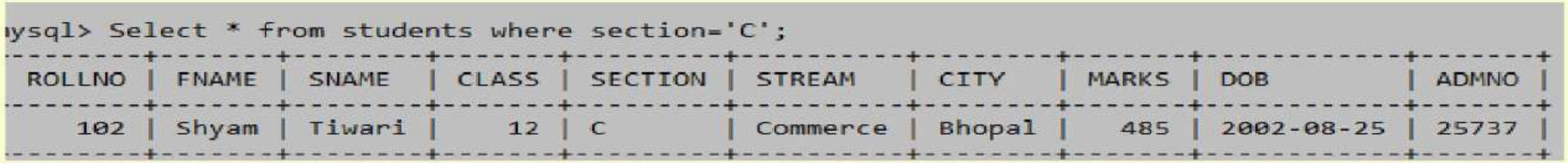
Here Only those rows will display who for which given condition will satisfied. (Condition returns TRUE)

Ex. **Mysql>Select * from students where city = "New Delhi" ;**



```
nysql> Select * from students where city = "New Delhi" ;
+--------+----------+--------+-------+---------+----------+------------+-------+------------+-------+
| ROLLNO | FNAME    | SNAME  | CLASS | SECTION | STREAM   | CITY       | MARKS | DOB        | ADMNO |
+--------+----------+--------+-------+---------+----------+------------+-------+------------+-------+
|    101 | Ram      | Sharma |    12 | A       | Science  | New Delhi  |   494 | 2001-04-15 | 20795 |
|    102 | Govind   | Shukla |    12 | B       | Science  | New Delhi  |   475 | 2002-08-03 | 25437 |
+--------+----------+--------+-------+---------+----------+------------+-------+------------+-------+
```

Ex. **Mysql> Select * from students where section = 'C' ;**



```
ysql> Select * from students where section='C';
+--------+-------+--------+-------+---------+----------+--------+-------+------------+-------+
| ROLLNO | FNAME | SNAME  | CLASS | SECTION | STREAM   | CITY   | MARKS | DOB        | ADMNO |
+--------+-------+--------+-------+---------+----------+--------+-------+------------+-------+
|    102 | Shyam | Tiwari |    12 | C       | Commerce | Bhopal |   485 | 2002-08-25 | 25737 |
+--------+-------+--------+-------+---------+----------+--------+-------+------------+-------+
```

Fetching data from table Logical Operators. **STRUCTURED QUERY LANGUAGE**

Select Command : Selected Rows with Logical Operators (AND,OR,NOT)

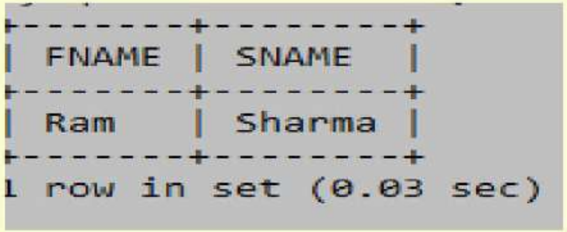
AND : All Conditions must be True to display a row

OR : At least one condition must be True to display a particular row

NOT : NOT operator just reverse the stage from True to False and Vice Versa

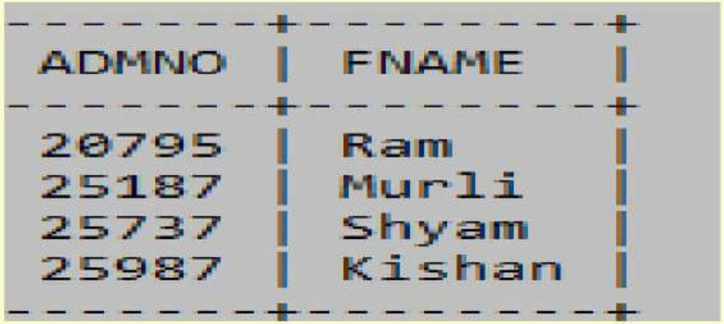
Ex.

SELECT FNAME,SNAME FROM STUDENTS WHERE CITY='New Delhi' AND SECTION='A';



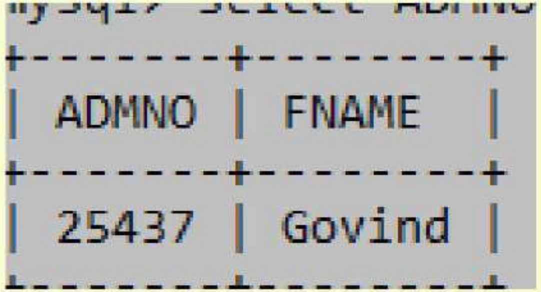
```
+--------+--------+
| FNAME  | SNAME  |
+--------+--------+
| Ram    | Sharma |
+--------+--------+
1 row in set (0.03 sec)
```

SELECT ADMNO,FNAME FROM STUDENTS WHERE STREAM='SCIENCE' AND SECTION='A' OR MARKS > 480;



```
--------+----------+
 ADMNO  | FNAME    |
--------+----------+
 20795  | Ram      |
 25187  | Murli    |
 25737  | Shyam    |
 25987  | Kishan   |
--------+----------+
```

SELECT ADMNO,FNAME FROM STUDENTS WHERE not(STREAM='SCIENCE' AND SECTION='A' OR MARKS > 480);



```
mysql> select ADMNO
+-------+--------+
| ADMNO | FNAME  |
+-------+--------+
| 25437 | Govind |
+-------+--------+
```

KAPIL SEHGAL

Fetching data from table **Range Searching** **STRUCTURED QUERY LANGUAGE**

Range Searching using “BETWEEN” and “AND” Operators

BETWEEN : Value BETWEEN …………… AND ………….. (Inclusive Both end Value)
AND : Value >= ………………. AND Value <= ……………
Here …......... (Range-1) and ……………… (Range - 2)
Value can be inform of (Numeric Value as well as Date Value and String Value)

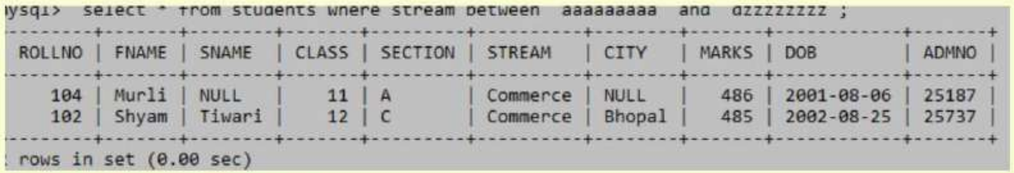
Ex.

Value in form of String

select * from students where stream between 'aaaaaaaaa' and 'dzzzzzzzz';

OR

select * from students where stream >= 'aaaaaaaaa' and stream <= 'dzzzzzzzz';



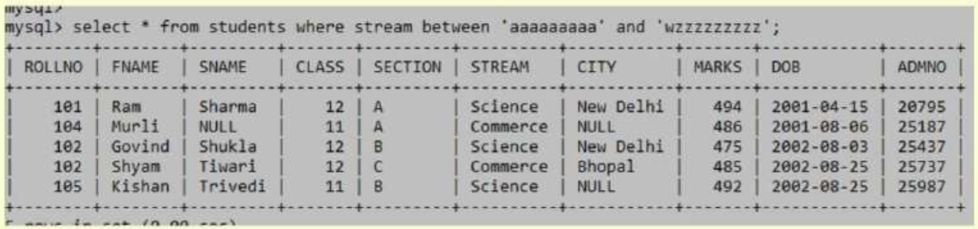
```
ysql>  select * from students where stream between 'aaaaaaaaa' and 'dzzzzzzzz';
--------+-------+--------+-------+---------+----------+--------+-------+------------+-------+
 ROLLNO | FNAME | SNAME  | CLASS | SECTION | STREAM   | CITY   | MARKS | DOB        | ADMNO |
--------+-------+--------+-------+---------+----------+--------+-------+------------+-------+
    104 | Murli | NULL   |    11 | A       | Commerce | NULL   |   486 | 2001-08-06 | 25187 |
    102 | Shyam | Tiwari |    12 | C       | Commerce | Bhopal |   485 | 2002-08-25 | 25737 |
--------+-------+--------+-------+---------+----------+--------+-------+------------+-------+
 rows in set (0.00 sec)
```

KAPIL SEHGAL

select * from students where stream between 'aaaaaaaaa' and 'wzzzzzzzzz';

OR

select * from students where stream >= 'aaaaaaaaa' and stream <= 'wzzzzzzzzz'



```
mysql>
mysql> select * from students where stream between 'aaaaaaaaa' and 'wzzzzzzzzz';
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME   | CLASS | SECTION | STREAM   | CITY      | MARKS | DOB        | ADMNO |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
|    101 | Ram    | Sharma  |    12 | A       | Science  | New Delhi |   494 | 2001-04-15 | 20795 |
|    104 | Murli  | NULL    |    11 | A       | Commerce | NULL      |   486 | 2001-08-06 | 25187 |
|    102 | Govind | Shukla  |    12 | B       | Science  | New Delhi |   475 | 2002-08-03 | 25437 |
|    102 | Shyam  | Tiwari  |    12 | C       | Commerce | Bhopal    |   485 | 2002-08-25 | 25737 |
|    105 | Kishan | Trivedi |    11 | B       | Science  | NULL      |   492 | 2002-08-25 | 25987 |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
5 rows in set (0.00 sec)
```

Fetching data from table **Range Searching** **STRUCTURED QUERY LANGUAGE**

Range Searching using “BETWEEN” and “AND” Operators

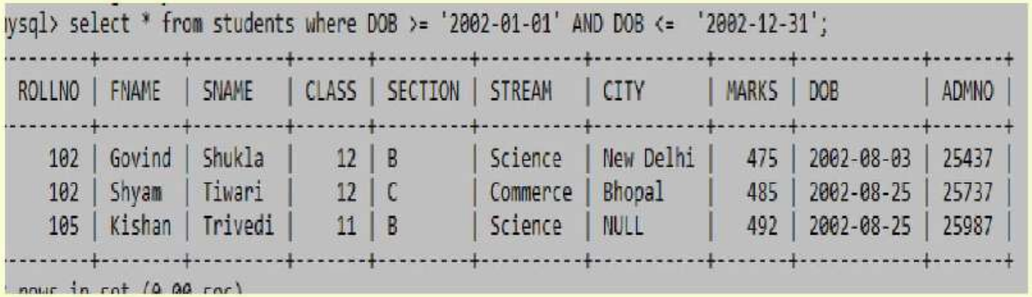
Ex.

Value in form of Date

select * from students where DOB BETWEEN '2002-01-01' AND '2002-12-31';

OR

select * from students where DOB >= '2002-01-01' AND DOB <= '2002-12-31';

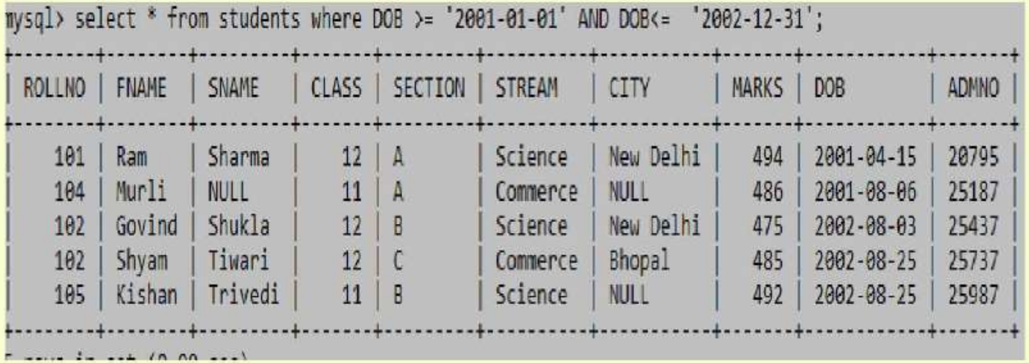


```
ysql> select * from students where DOB >= '2002-01-01' AND DOB <=  '2002-12-31';
--------+--------+---------+-------+---------+----------+------------+-------+------------+-------+
 ROLLNO | FNAME  | SNAME   | CLASS | SECTION | STREAM   | CITY       | MARKS | DOB        | ADMNO |
--------+--------+---------+-------+---------+----------+------------+-------+------------+-------+
    102 | Govind | Shukla  |    12 | B       | Science  | New Delhi  |   475 | 2002-08-03 | 25437 |
    102 | Shyam  | Tiwari  |    12 | C       | Commerce | Bhopal     |   485 | 2002-08-25 | 25737 |
    105 | Kishan | Trivedi |    11 | B       | Science  | NULL       |   492 | 2002-08-25 | 25987 |
--------+--------+---------+-------+---------+----------+------------+-------+------------+-------+
 rows in set (0.00 sec)
```

select * from students where DOB BETWEEN '2001-01-01' AND '2002-12-31';

OR

select * from students where DOB >= '2001-01-01' AND DOB<= '2002-12-31';



```
mysql> select * from students where DOB >= '2001-01-01' AND DOB<=  '2002-12-31';
+--------+--------+---------+-------+---------+----------+------------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME   | CLASS | SECTION | STREAM   | CITY       | MARKS | DOB        | ADMNO |
+--------+--------+---------+-------+---------+----------+------------+-------+------------+-------+
|    101 | Ram    | Sharma  |    12 | A       | Science  | New Delhi  |   494 | 2001-04-15 | 20795 |
|    104 | Murli  | NULL    |    11 | A       | Commerce | NULL       |   486 | 2001-08-06 | 25187 |
|    102 | Govind | Shukla  |    12 | B       | Science  | New Delhi  |   475 | 2002-08-03 | 25437 |
|    102 | Shyam  | Tiwari  |    12 | C       | Commerce | Bhopal     |   485 | 2002-08-25 | 25737 |
|    105 | Kishan | Trivedi |    11 | B       | Science  | NULL       |   492 | 2002-08-25 | 25987 |
+--------+--------+---------+-------+---------+----------+------------+-------+------------+-------+
5 rows in set (0.00 sec)
```

Fetching data from table Range Searching

STRUCTURED QUERY LANGUAGE

Range Searching using “BETWEEN” and “AND” Operators

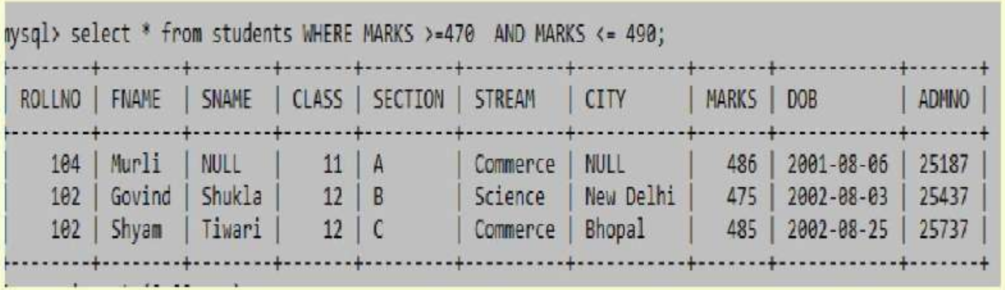
Ex.

Value in form of Numeric Value

select * from students WHERE MARKS BETWEEN 470 AND 490;

OR

select * from students WHERE MARKS >=470 AND MARKS <= 490;

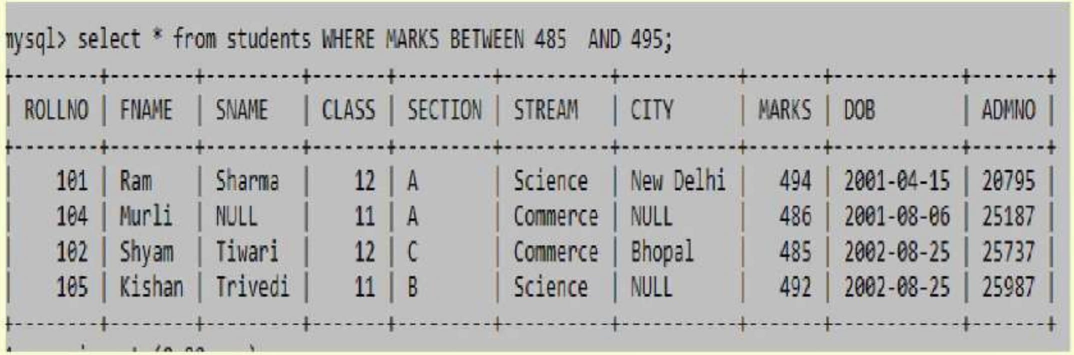


```
mysql> select * from students WHERE MARKS >=470  AND MARKS <= 490;
+--------+--------+--------+-------+---------+----------+-----------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME  | CLASS | SECTION | STREAM   | CITY      | MARKS | DOB        | ADMNO |
+--------+--------+--------+-------+---------+----------+-----------+-------+------------+-------+
|    104 | Murli  | NULL   |    11 | A       | Commerce | NULL      |   486 | 2001-08-06 | 25187 |
|    102 | Govind | Shukla |    12 | B       | Science  | New Delhi |   475 | 2002-08-03 | 25437 |
|    102 | Shyam  | Tiwari |    12 | C       | Commerce | Bhopal    |   485 | 2002-08-25 | 25737 |
+--------+--------+--------+-------+---------+----------+-----------+-------+------------+-------+
```

select * from students WHERE MARKS BETWEEN 485 AND 495;

OR

select * from students WHERE MARKS >=485 AND MARKS <= 495;



```
mysql> select * from students WHERE MARKS BETWEEN 485  AND 495;
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME   | CLASS | SECTION | STREAM   | CITY      | MARKS | DOB        | ADMNO |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
|    101 | Ram    | Sharma  |    12 | A       | Science  | New Delhi |   494 | 2001-04-15 | 20795 |
|    104 | Murli  | NULL    |    11 | A       | Commerce | NULL      |   486 | 2001-08-06 | 25187 |
|    102 | Shyam  | Tiwari  |    12 | C       | Commerce | Bhopal    |   485 | 2002-08-25 | 25737 |
|    105 | Kishan | Trivedi |    11 | B       | Science  | NULL      |   492 | 2002-08-25 | 25987 |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
```

KAPIL SEHGAL

Fetching data from table In / Not In STRUCTURED QUERY LANGUAGE

IN keyword only affects the rows whose values matches the list of values provided in the IN keyword. IN helps reduces number of OR clauses you may have to use

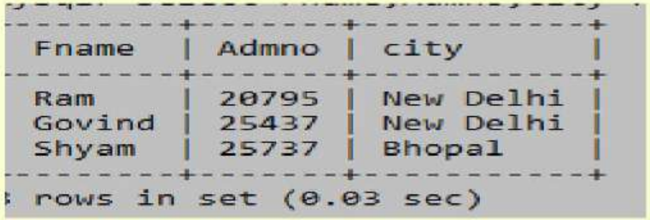
Ex.

Give the Name and Admission number and city of those students who city either New Delhi , Mumbai or Bhopal

select Fname,Admno,city from students where city in ('New Delhi','Bhopal','Mumbai');

OR

select Fname,Admno,city from students where city='New Delhi' OR city='Mumbai' or city='Bhopal';



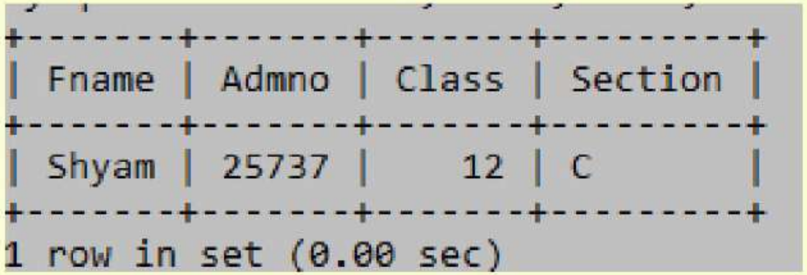
```
--------+-------+-----------+
 Fname  | Admno | city      |
--------+-------+-----------+
 Ram    | 20795 | New Delhi |
 Govind | 25437 | New Delhi |
 Shyam  | 25737 | Bhopal    |
--------+-------+-----------+
3 rows in set (0.03 sec)
```

Give the Name and Admission number, Class and Section of those students who studying either Neither Section A Nor Section B.

select Fname,Admno,Class,Section from Students where Section not in ('A','B');

OR

select Fname,Admno,Class,Section from Students where Section<>'A' AND Section<>'B';



```
+-------+-------+-------+---------+
| Fname | Admno | Class | Section |
+-------+-------+-------+---------+
| Shyam | 25737 |    12 | C       |
+-------+-------+-------+---------+
1 row in set (0.00 sec)
```

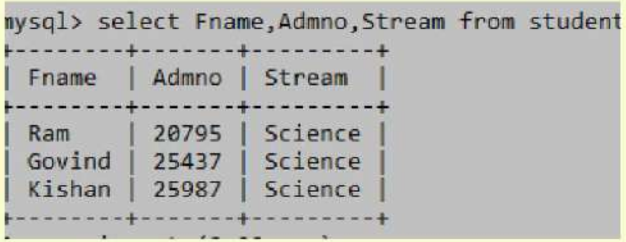
Pattern Matching using “%” and “_” with ‘LIKE’ clause

It **matches** any **pattern** based on some conditions provided using the wildcard characters. Some of the commonly used wildcard characters in MySQL are as follows: '%' represents zero or more characters. '_' represents exactly 1 character

Ex.

Give Fname, Admno and Stream whose name First Name starts with ‘S’

```
select Fname,Admno,Stream
from students
where stream like 'S%';
```

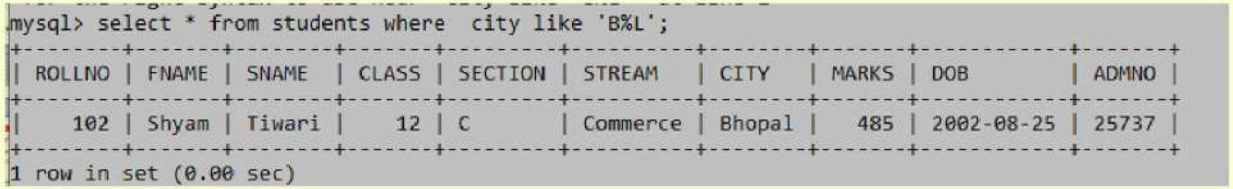


```
mysql> select Fname,Admno,Stream from student
+--------+-------+---------+
| Fname  | Admno | Stream  |
+--------+-------+---------+
| Ram    | 20795 | Science |
| Govind | 25437 | Science |
| Kishan | 25987 | Science |
+--------+-------+---------+
```

KAPIL SEHGAL

Give the details of students whose city name start with B and ends with L

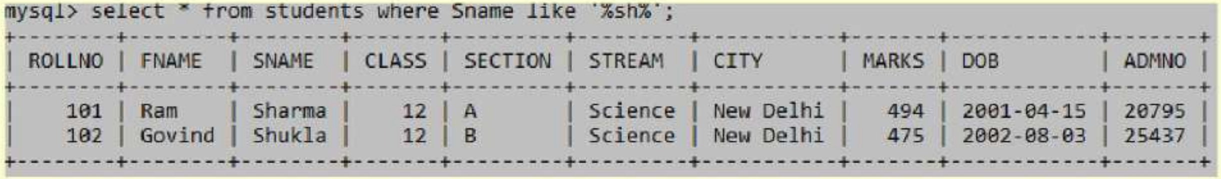
```
select * from students where  city like 'B%L';
```



```
mysql> select * from students where  city like 'B%L';
+--------+-------+--------+-------+---------+----------+--------+-------+------------+-------+
| ROLLNO | FNAME | SNAME  | CLASS | SECTION | STREAM   | CITY   | MARKS | DOB        | ADMNO |
+--------+-------+--------+-------+---------+----------+--------+-------+------------+-------+
|    102 | Shyam | Tiwari |    12 | C       | Commerce | Bhopal |   485 | 2002-08-25 | 25737 |
+--------+-------+--------+-------+---------+----------+--------+-------+------------+-------+
1 row in set (0.00 sec)
```

Give the details of students whose sname contains “sh” any where in any location (either beginning ,end or middle)

```
select * from students where Sname like '%sh%';
```

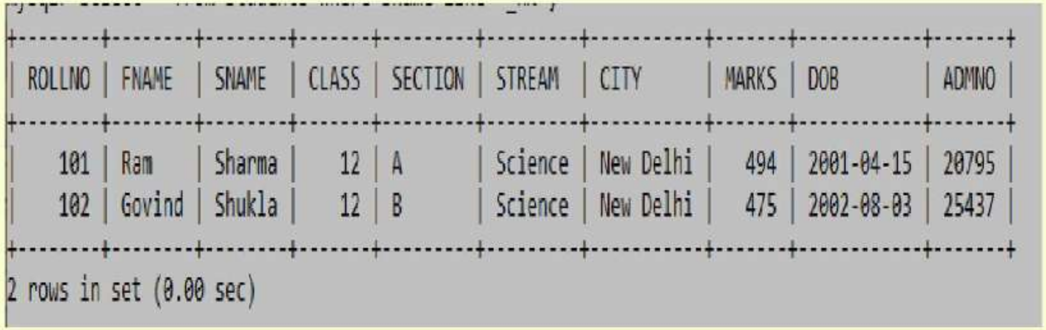


```
mysql> select * from students where Sname like '%sh%';
+--------+--------+--------+-------+---------+---------+-----------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME  | CLASS | SECTION | STREAM  | CITY      | MARKS | DOB        | ADMNO |
+--------+--------+--------+-------+---------+---------+-----------+-------+------------+-------+
|    101 | Ram    | Sharma |    12 | A       | Science | New Delhi |   494 | 2001-04-15 | 20795 |
|    102 | Govind | Shukla |    12 | B       | Science | New Delhi |   475 | 2002-08-03 | 25437 |
+--------+--------+--------+-------+---------+---------+-----------+-------+------------+-------+
```

Pattern Matching using "%" and "_" with 'LIKE' clause

Give The Details of those students whose second letter of surname is 'h'

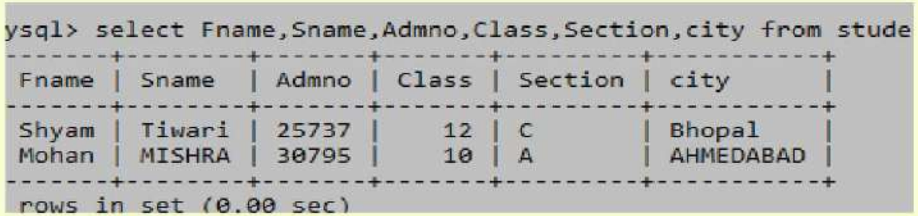
select * from students where sname like '_h%';



```
+--------+--------+--------+-------+---------+---------+-----------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME  | CLASS | SECTION | STREAM  | CITY      | MARKS | DOB        | ADMNO |
+--------+--------+--------+-------+---------+---------+-----------+-------+------------+-------+
|    101 | Ram    | Sharma |    12 | A       | Science | New Delhi |   494 | 2001-04-15 | 20795 |
|    102 | Govind | Shukla |    12 | B       | Science | New Delhi |   475 | 2002-08-03 | 25437 |
+--------+--------+--------+-------+---------+---------+-----------+-------+------------+-------+
2 rows in set (0.00 sec)
```

Give The First Name, Surname, Class, Section, Admission No and City of those students whose city contains second letter 'h' and second last letter is 'a'

select Fname,Sname,Admno,Class,Section,city from students where city like '_h%a_';



```
ysql> select Fname,Sname,Admno,Class,Section,city from stude
-------+--------+-------+-------+---------+-----------+
 Fname | Sname  | Admno | Class | Section | city      |
-------+--------+-------+-------+---------+-----------+
 Shyam | Tiwari | 25737 |    12 | C       | Bhopal    |
 Mohan | MISHRA | 30795 |    10 | A       | AHMEDABAD |
-------+--------+-------+-------+---------+-----------+
 rows in set (0.00 sec)
```

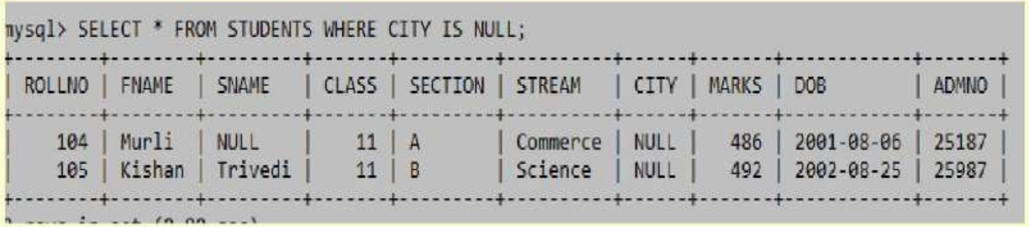
Searching NULL Values

IS NULL / IS NOT NULL

NULL is a special value that signifies 'no value‘. Comparing a column to NULL using **IS NULL** or **IS NOT NULL**.

Give The Details of those students Whose city contains NULL

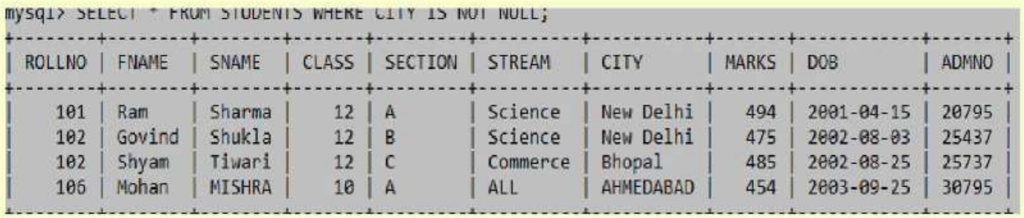
SELECT * FROM STUDENTS WHERE CITY IS NULL;



```
mysql> SELECT * FROM STUDENTS WHERE CITY IS NULL;
+--------+--------+---------+-------+---------+----------+------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME   | CLASS | SECTION | STREAM   | CITY | MARKS | DOB        | ADMNO |
+--------+--------+---------+-------+---------+----------+------+-------+------------+-------+
|    104 | Murli  | NULL    |    11 | A       | Commerce | NULL |   486 | 2001-08-06 | 25187 |
|    105 | Kishan | Trivedi |    11 | B       | Science  | NULL |   492 | 2002-08-25 | 25987 |
+--------+--------+---------+-------+---------+----------+------+-------+------------+-------+
2 rows in set (0.00 sec)
```

Give The Details of those students Whose city contains NOT NULL

SELECT * FROM STUDENTS WHERE CITY IS NOT NULL;



```
mysql> SELECT * FROM STUDENTS WHERE CITY IS NOT NULL;
+--------+--------+--------+-------+---------+----------+-----------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME  | CLASS | SECTION | STREAM   | CITY      | MARKS | DOB        | ADMNO |
+--------+--------+--------+-------+---------+----------+-----------+-------+------------+-------+
|    101 | Ram    | Sharma |    12 | A       | Science  | New Delhi |   494 | 2001-04-15 | 20795 |
|    102 | Govind | Shukla |    12 | B       | Science  | New Delhi |   475 | 2002-08-03 | 25437 |
|    102 | Shyam  | Tiwari |    12 | C       | Commerce | Bhopal    |   485 | 2002-08-25 | 25737 |
|    106 | Mohan  | MISHRA |    10 | A       | ALL      | AHMEDABAD |   454 | 2003-09-25 | 30795 |
+--------+--------+--------+-------+---------+----------+-----------+-------+------------+-------+
```

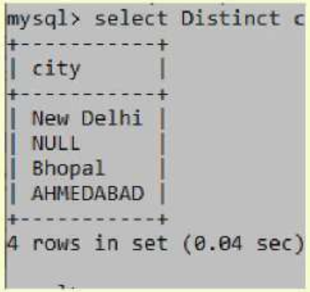
Using "DISTINCT"

The SQL SELECT DISTINCT Statement

The SELECT **DISTINCT** statement is used to return only **distinct** (different) values. Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (**distinct**) values.

List the Different city names from students table

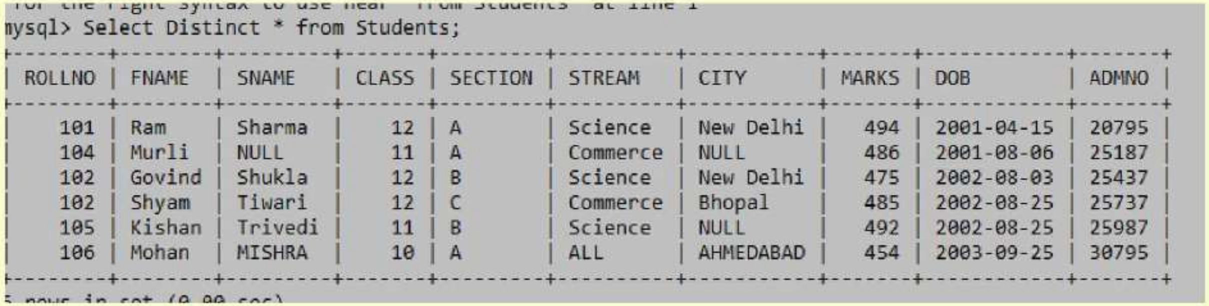
Select Distinct City from Students;



```
mysql> select Distinct c
+-----------+
| city      |
+-----------+
| New Delhi |
| NULL      |
| Bhopal    |
| AHMEDABAD |
+-----------+
4 rows in set (0.04 sec)
```

Give the details of all students but duplicate record will show only once.

SELECT Distinct * FROM STUDENTS;



```
for the right syntax to use near ' from Students' at line 1
mysql> Select Distinct * from Students;
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME   | CLASS | SECTION | STREAM   | CITY      | MARKS | DOB        | ADMNO |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
|    101 | Ram    | Sharma  |    12 | A       | Science  | New Delhi |   494 | 2001-04-15 | 20795 |
|    104 | Murli  | NULL    |    11 | A       | Commerce | NULL      |   486 | 2001-08-06 | 25187 |
|    102 | Govind | Shukla  |    12 | B       | Science  | New Delhi |   475 | 2002-08-03 | 25437 |
|    102 | Shyam  | Tiwari  |    12 | C       | Commerce | Bhopal    |   485 | 2002-08-25 | 25737 |
|    105 | Kishan | Trivedi |    11 | B       | Science  | NULL      |   492 | 2002-08-25 | 25987 |
|    106 | Mohan  | MISHRA  |    10 | A       | ALL      | AHMEDABAD |   454 | 2003-09-25 | 30795 |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
5 rows in set (0.00 sec)
```

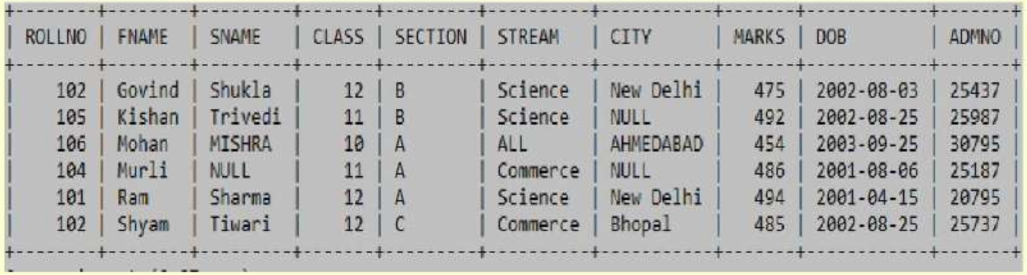
Order By

The **ORDER BY Clause** can be used along with the SELECT statement to **sort** the data of specific fields in an **ordered** way. It is used to **sort** the result-set in ascending or descending **order**.

List the details of the students which is arranged by the Fname in ascending Order

Select * from students order by Fname;

Note: By default sorting of data will be in ascending order

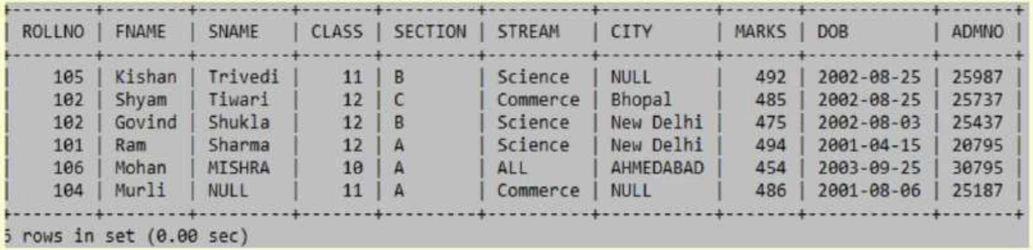


```
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME   | CLASS | SECTION | STREAM   | CITY      | MARKS | DOB        | ADMNO |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
|    102 | Govind | Shukla  |    12 | B       | Science  | New Delhi |   475 | 2002-08-03 | 25437 |
|    105 | Kishan | Trivedi |    11 | B       | Science  | NULL      |   492 | 2002-08-25 | 25987 |
|    106 | Mohan  | MISHRA  |    10 | A       | ALL      | AHMEDABAD |   454 | 2003-09-25 | 30795 |
|    104 | Murli  | NULL    |    11 | A       | Commerce | NULL      |   486 | 2001-08-06 | 25187 |
|    101 | Ram    | Sharma  |    12 | A       | Science  | New Delhi |   494 | 2001-04-15 | 20795 |
|    102 | Shyam  | Tiwari  |    12 | C       | Commerce | Bhopal    |   485 | 2002-08-25 | 25737 |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
```

List the details of the students which is arranged by the Sname in Descending order

Note: for arranging data in descending order use 'DESC' Keyword (clause)\

SELECT * FROM STUDENTS ORDER BY SNAME desc;



```
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME   | CLASS | SECTION | STREAM   | CITY      | MARKS | DOB        | ADMNO |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
|    105 | Kishan | Trivedi |    11 | B       | Science  | NULL      |   492 | 2002-08-25 | 25987 |
|    102 | Shyam  | Tiwari  |    12 | C       | Commerce | Bhopal    |   485 | 2002-08-25 | 25737 |
|    102 | Govind | Shukla  |    12 | B       | Science  | New Delhi |   475 | 2002-08-03 | 25437 |
|    101 | Ram    | Sharma  |    12 | A       | Science  | New Delhi |   494 | 2001-04-15 | 20795 |
|    106 | Mohan  | MISHRA  |    10 | A       | ALL      | AHMEDABAD |   454 | 2003-09-25 | 30795 |
|    104 | Murli  | NULL    |    11 | A       | Commerce | NULL      |   486 | 2001-08-06 | 25187 |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
5 rows in set (0.00 sec)
```

Order By with multiple field

When **multiple columns** are used in ORDER BY, first the rows will be sorted based on the first column and then by the second column and so on.

Note: By default sorting of data will be in ascending order, You can also make the combination of First sorted with ascending order of any column then second may be descending order or vice versa

Ex. Give the details of students sorted Class then Section the third key will be first name.

```
Select * from students order by class , section , Fname
```

Ex. Give the details of students sorted Class then Section descending order the third key will be first name.

```
select * from students order by class, section desc , fname ;
```

Ex. Give the details of students sorted Class descending order then Section the third key will be first name in descending order.

```
select * from students order by class desc, section , fname desc;
```

Delete Command

STRUCTURED QUERY LANGUAGE

DELETE command to **delete** data from a **MySQL** table. If the WHERE clause is not specified, then all the records will be **deleted** from the given **MySQL** table. You can specify any condition using the WHERE clause

To delete all records from table.

Delete from TableName;

Ex. Delete All Records from Student Table

Delete from Students;

To delete selected records from table.

Delete from TableName where Condition ;

KAPIL SEHGAL

Delete all those students whose city is not mentioned

Delete from students where city is NULL ;

Delete all those students whose Name is "RAJ"

Delete from students where fname = "RAJ" ;

Delete all those students whose marks between 400 to 450

Delete from students where marks between 400 and 500 ;

Delete all those students whose name starts with 'A'

Delete from students where fname like 'A%';

Update Command

The MySQL UPDATE query is used to update existing records in a table in a MySQL database.

(1) It can be used to update one or more field at the same time.

(2) It can be used to specify any condition using the WHERE clause.

Syntax :

```
Update TableName
Set Column1=value1 [, Column2= Value2, ..]
[Where <Condition>;]
```

Update city as "New Delhi"all those students whose city is not mentioned

```
Update students Set City = 'New Delhi'
where city is NULL ;
```

Update Fanme as "Raj Kumar' all that students whose admno = 1250

```
Update students set fname = 'Raj Kumar'
where admno=1250
```

Update marks as 500 all those students whose marks between 400 to 450

```
Update Students set marks = 500 where marks
between 400 and 500 ;
```

AGGREGATE FUNCTIONS

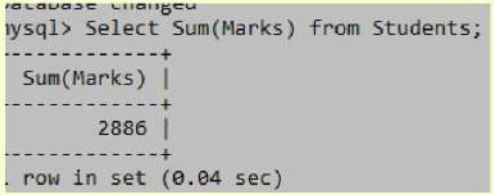
SUM()

Returns the **SUM** of all the values, or only the **DISTINCT** values, in the expression. SUM can be used with numeric columns only. Null values are ignored.

SUM ([DISTINCT] expression / Column)
Here Duplicate(in case of Distinct) values counts only once.

Give the sum of marks of all students

Select Sum(Marks) from Students;

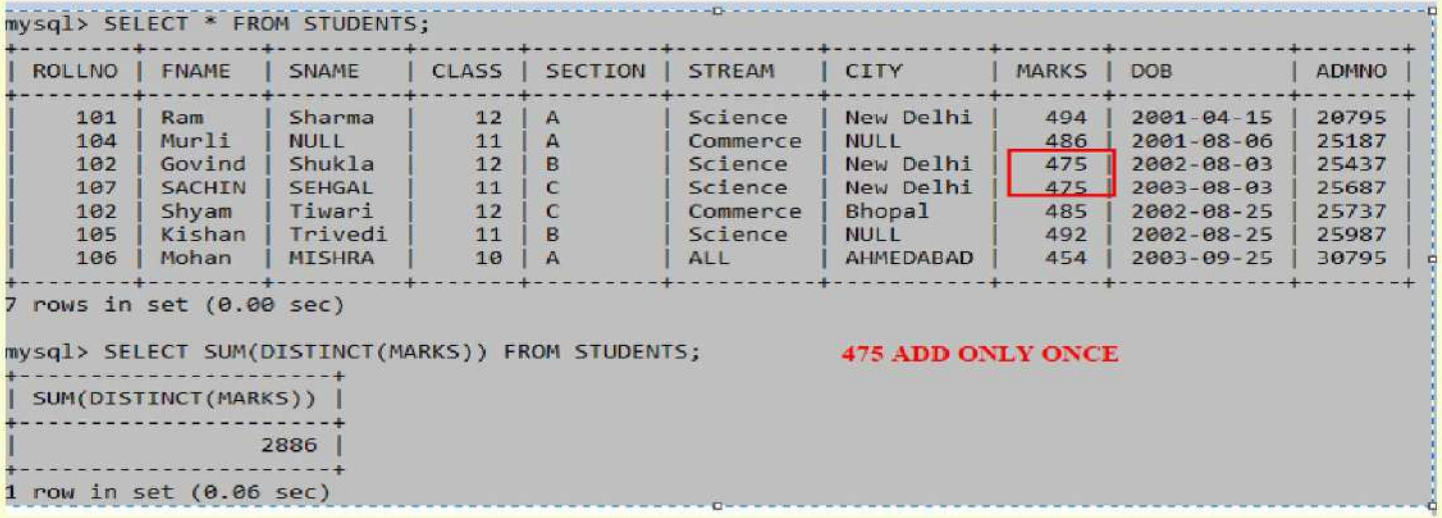


```
atabase changed
ysql> Select Sum(Marks) from Students;
-------------+
 Sum(Marks) |
-------------+
       2886 |
-------------+
. row in set (0.04 sec)
```

KAPIL SEHGAL

Give the sum of Distinct marks of all students (Duplicate marks count once)

Select Sum(Distinct(Marks)) from Students;



```
mysql> SELECT * FROM STUDENTS;
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME   | CLASS | SECTION | STREAM   | CITY      | MARKS | DOB        | ADMNO |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
|    101 | Ram    | Sharma  |    12 | A       | Science  | New Delhi |   494 | 2001-04-15 | 20795 |
|    104 | Murli  | NULL    |    11 | A       | Commerce | NULL      |   486 | 2001-08-06 | 25187 |
|    102 | Govind | Shukla  |    12 | B       | Science  | New Delhi |   475 | 2002-08-03 | 25437 |
|    107 | SACHIN | SEHGAL  |    11 | C       | Science  | New Delhi |   475 | 2003-08-03 | 25687 |
|    102 | Shyam  | Tiwari  |    12 | C       | Commerce | Bhopal    |   485 | 2002-08-25 | 25737 |
|    105 | Kishan | Trivedi |    11 | B       | Science  | NULL      |   492 | 2002-08-25 | 25987 |
|    106 | Mohan  | MISHRA  |    10 | A       | ALL      | AHMEDABAD |   454 | 2003-09-25 | 30795 |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
7 rows in set (0.00 sec)

mysql> SELECT SUM(DISTINCT(MARKS)) FROM STUDENTS;
+----------------------+
| SUM(DISTINCT(MARKS)) |
+----------------------+
|                 2886 |
+----------------------+
1 row in set (0.06 sec)
```

475 ADD ONLY ONCE

AGGREGATE FUNCTIONS

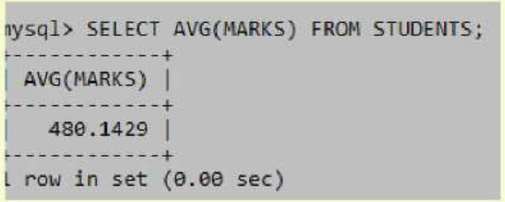
AVG()

Returns the **AVG** of all the values, or only the **DISTINCT** values, in the expression. AVG can be used with numeric columns only. Null values are ignored.

AVG ([DISTINCT] expression / Column)
Here Duplicate(in case of Distinct) values counts only once.

Give the Average of marks of all students

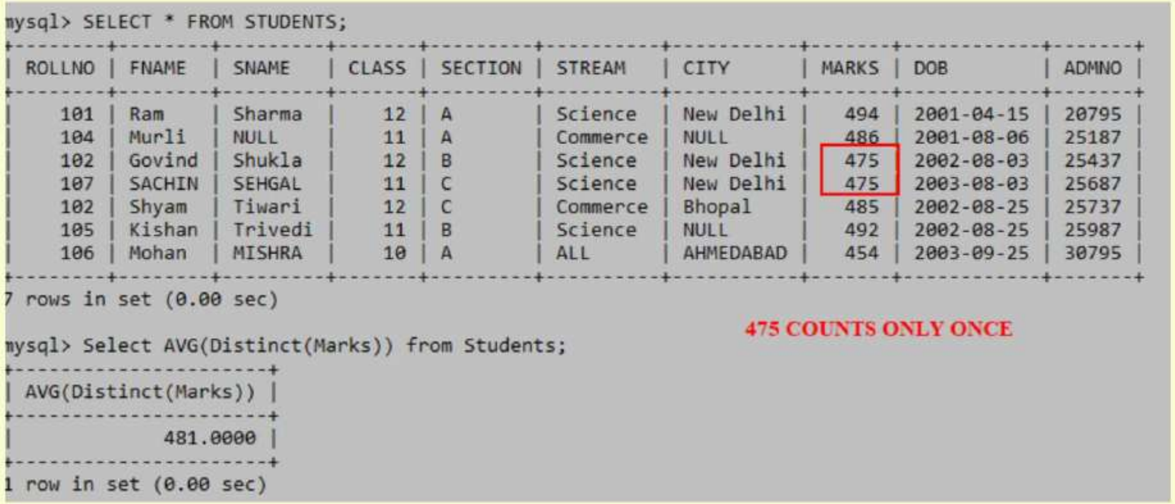
Select AVG(Marks) from Students;



```
mysql> SELECT AVG(MARKS) FROM STUDENTS;
+-------------+
| AVG(MARKS) |
+-------------+
|   480.1429 |
+-------------+
1 row in set (0.00 sec)
```

Give the Average of Distinct marks of all students (Duplicate marks count once)

Select AVG(Distinct(Marks)) from Students;



```
mysql> SELECT * FROM STUDENTS;
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME   | CLASS | SECTION | STREAM   | CITY      | MARKS | DOB        | ADMNO |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
|    101 | Ram    | Sharma  |    12 | A       | Science  | New Delhi |   494 | 2001-04-15 | 20795 |
|    104 | Murli  | NULL    |    11 | A       | Commerce | NULL      |   486 | 2001-08-06 | 25187 |
|    102 | Govind | Shukla  |    12 | B       | Science  | New Delhi |   475 | 2002-08-03 | 25437 |
|    107 | SACHIN | SEHGAL  |    11 | C       | Science  | New Delhi |   475 | 2003-08-03 | 25687 |
|    102 | Shyam  | Tiwari  |    12 | C       | Commerce | Bhopal    |   485 | 2002-08-25 | 25737 |
|    105 | Kishan | Trivedi |    11 | B       | Science  | NULL      |   492 | 2002-08-25 | 25987 |
|    106 | Mohan  | MISHRA  |    10 | A       | ALL      | AHMEDABAD |   454 | 2003-09-25 | 30795 |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
7 rows in set (0.00 sec)

mysql> Select AVG(Distinct(Marks)) from Students;
+----------------------+
| AVG(Distinct(Marks)) |
+----------------------+
|             481.0000 |
+----------------------+
1 row in set (0.00 sec)
```

475 COUNTS ONLY ONCE

KAPIL SEHGAL

MAX()

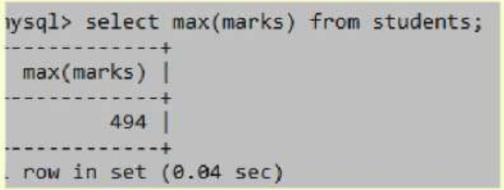
AGGREGATE FUNCTIONS

Returns the **Maximum Value** from entire Column Max() can be used with numeric columns only.

MAX (Column)

Give the Average of marks of all students

Select Max(Marks) from Students;

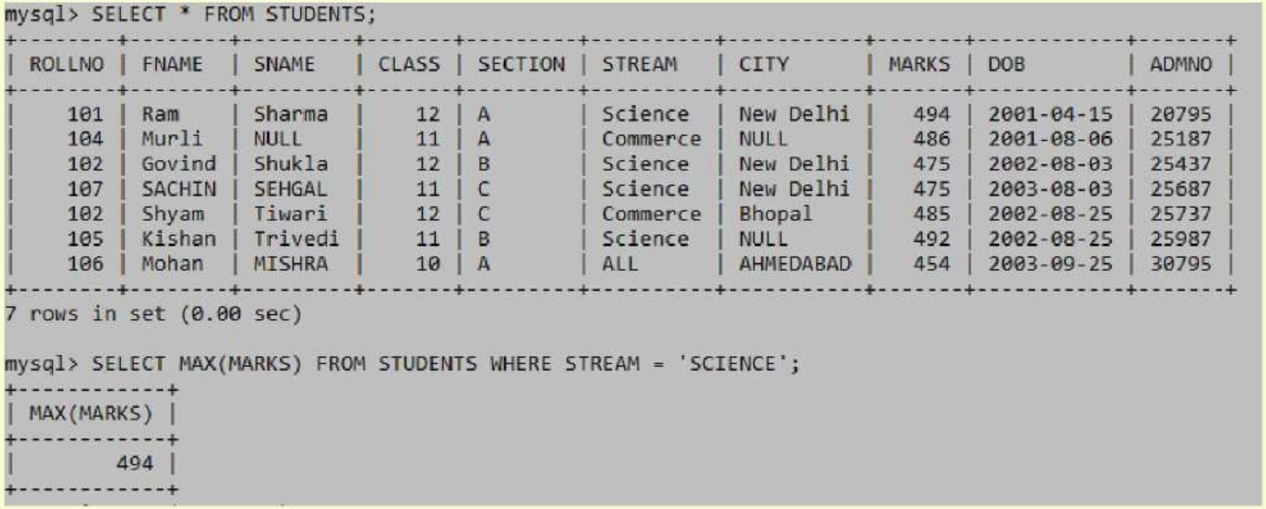


```
ysql> select max(marks) from students;
-------------+
 max(marks) |
-------------+
        494 |
-------------+
 row in set (0.04 sec)
```

KAPIL SEHGAL

Give the Maximum Marks from students table who are studying in Science stream.

SELECT MAX(MARKS) FROM STUDENTS WHERE STREAM = 'SCIENCE';



```
mysql> SELECT * FROM STUDENTS;
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME   | CLASS | SECTION | STREAM   | CITY      | MARKS | DOB        | ADMNO |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
|    101 | Ram    | Sharma  |    12 | A       | Science  | New Delhi |   494 | 2001-04-15 | 20795 |
|    104 | Murli  | NULL    |    11 | A       | Commerce | NULL      |   486 | 2001-08-06 | 25187 |
|    102 | Govind | Shukla  |    12 | B       | Science  | New Delhi |   475 | 2002-08-03 | 25437 |
|    107 | SACHIN | SEHGAL  |    11 | C       | Science  | New Delhi |   475 | 2003-08-03 | 25687 |
|    102 | Shyam  | Tiwari  |    12 | C       | Commerce | Bhopal    |   485 | 2002-08-25 | 25737 |
|    105 | Kishan | Trivedi |    11 | B       | Science  | NULL      |   492 | 2002-08-25 | 25987 |
|    106 | Mohan  | MISHRA  |    10 | A       | ALL      | AHMEDABAD |   454 | 2003-09-25 | 30795 |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
7 rows in set (0.00 sec)

mysql> SELECT MAX(MARKS) FROM STUDENTS WHERE STREAM = 'SCIENCE';
+------------+
| MAX(MARKS) |
+------------+
|        494 |
+------------+
```

MIN()

AGGREGATE FUNCTIONS

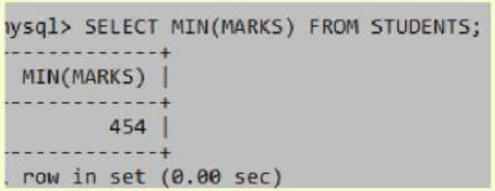
Returns the **Minimum Value** from entire Column Min() can be used with numeric columns only.

MIN (Column)

Give the Minimum Marks from students table who are studying in 12th class.

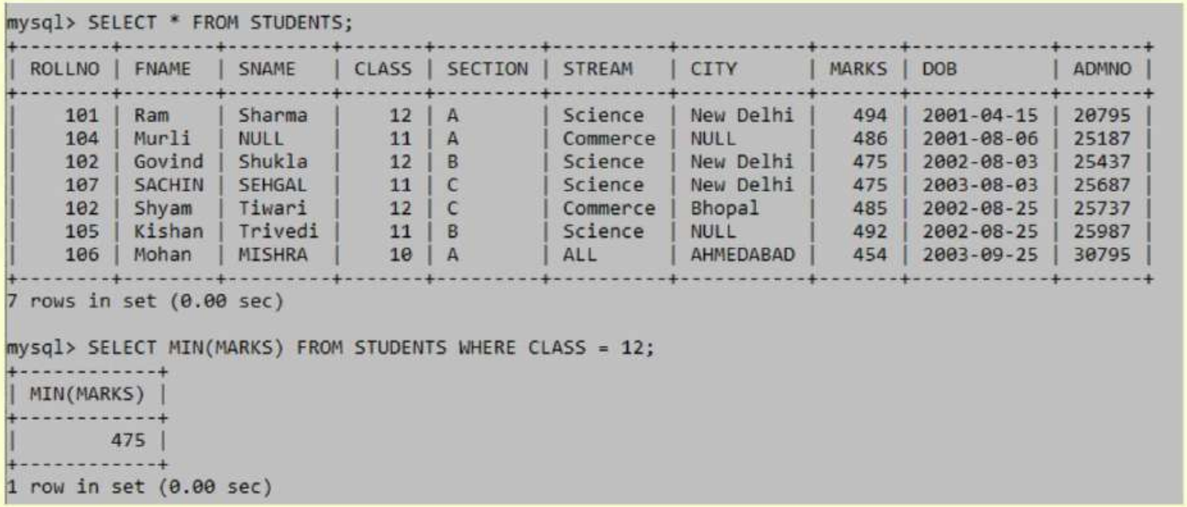
Give the Average of marks of all students

Select Min(Marks) from Students;



```
ysql> SELECT MIN(MARKS) FROM STUDENTS;
-------------+
 MIN(MARKS) |
-------------+
        454 |
-------------+
 row in set (0.00 sec)
```

SELECT MAX(MARKS) FROM STUDENTS WHERE CLASS = 12;



```
mysql> SELECT * FROM STUDENTS;
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME   | CLASS | SECTION | STREAM   | CITY      | MARKS | DOB        | ADMNO |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
|    101 | Ram    | Sharma  |    12 | A       | Science  | New Delhi |   494 | 2001-04-15 | 20795 |
|    104 | Murli  | NULL    |    11 | A       | Commerce | NULL      |   486 | 2001-08-06 | 25187 |
|    102 | Govind | Shukla  |    12 | B       | Science  | New Delhi |   475 | 2002-08-03 | 25437 |
|    107 | SACHIN | SEHGAL  |    11 | C       | Science  | New Delhi |   475 | 2003-08-03 | 25687 |
|    102 | Shyam  | Tiwari  |    12 | C       | Commerce | Bhopal    |   485 | 2002-08-25 | 25737 |
|    105 | Kishan | Trivedi |    11 | B       | Science  | NULL      |   492 | 2002-08-25 | 25987 |
|    106 | Mohan  | MISHRA  |    10 | A       | ALL      | AHMEDABAD |   454 | 2003-09-25 | 30795 |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
7 rows in set (0.00 sec)

mysql> SELECT MIN(MARKS) FROM STUDENTS WHERE CLASS = 12;
+------------+
| MIN(MARKS) |
+------------+
|        475 |
+------------+
1 row in set (0.00 sec)
```

KAPIL SEHGAL

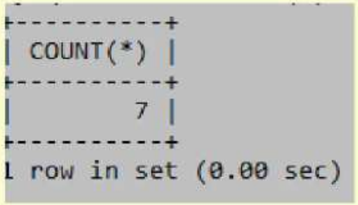
COUNT()

AGGREGATE FUNCTIONS

1. Count (*) returns the **number of rows** : it does not consider any column contain NULL value or not.
2. Count (Column Name) : returns Number of Rows that contains **Valid (Not Null)** values.

Count How many students in your table.

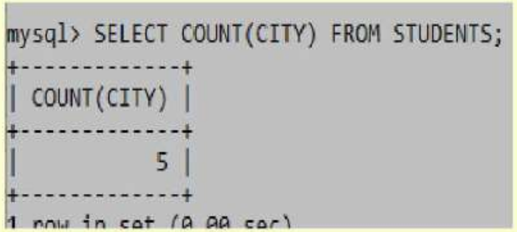
Select Count(*) from Students;



```
+----------+
| COUNT(*) |
+----------+
|        7 |
+----------+
1 row in set (0.00 sec)
```

Count the number of Cities mention in the students table

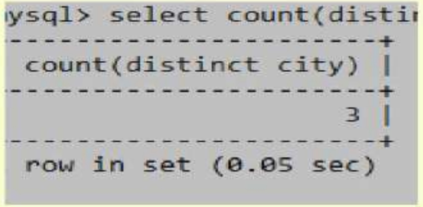
Select Count(city) from Students;



```
mysql> SELECT COUNT(CITY) FROM STUDENTS;
+-------------+
| COUNT(CITY) |
+-------------+
|           5 |
+-------------+
1 row in set (0.00 sec)
```

Count number of distinct cities

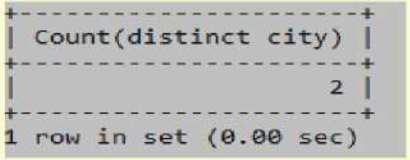
Select Count(distinct city) from Students;



```
ysql> select count(disti
---------------------+
 count(distinct city) |
---------------------+
                    3 |
---------------------+
 row in set (0.05 sec)
```

Count number of distinct cities of those students who studying in class 12th.

Select Count(distinct city) from Students where class=12;



```
+----------------------+
| Count(distinct city) |
+----------------------+
|                    2 |
+----------------------+
1 row in set (0.00 sec)
```

KAPIL SEHGAL

Mathematical Functions

SCALAR FUNCTION OR NON-DATABASE FUNCTIONS

Mathematical functions perform mathematical operations on numeric values.

POWER() / POW()

Returns the argument raised to the specified power. pow() works the same way.

Its syntax is: **POW(m,n)**

Here, function calculates m raise to power n.

Ex.

(i) select pow(2,4); Result: 16
(ii) select pow(2,–2); Result: 0.25
(iii) select pow(–2,3); Result: –8

MOD()

The mod() function returns the remainder of one number divided by another.

Its syntax is: **MOD(dividend, divisor)**

Ex.

(i) select mod(11, 3); Result: 2
(ii) select mod(10.5, 3); Result: 1.5

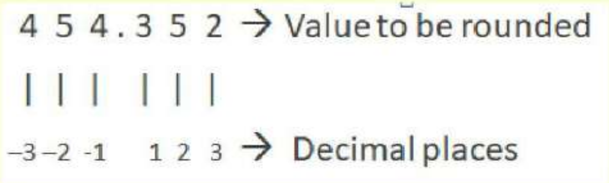
KAPIL SEHGAL

Mathematical Functions

SCALAR FUNCTION OR NON-DATABASE FUNCTIONS

Round(nmuber / Column, [Number of Decimal places])

The number of decimal places rounded to. This value will be a positive or negative integer or zero. If this parameter is omitted, the truncate function will round the number to 0 decimal places.



```
4 5 4 . 3 5 2 → Value to be rounded
| | |   | | |
-3 -2 -1  1 2 3 → Decimal places
```

1. Decimal place position value is rounded off to the next integer if the next number on the right side is greater than 5 (>=5).
2. Default decimal place is 0 if nothing is specified and returns the result in integer form.

Ex. Second parameter is positive or zero

1. Select round(–1.23); Result: –1
2. Select round(–1.58); Result: –2
3. Select round(1.58); Result: 2
4. Select round(3.798, 1); Result: 3.8
5. Select round(1.298, 0); Result: 1
6. Select round(23.298, 2); Result: 23.30

Ex. Second Parameter is negative

1. Select round(27542.29,–1); Result: 27540

2 .Select round(27542.29,–2); Result: 27500

3. Select round(27542.29, –3); Result: 28000
4. Select round(27542.29,–4); Result: 30000
5. Select round(77542.29,–5); Result: 100000
6. Select round(27542.29,–6); Result: 0

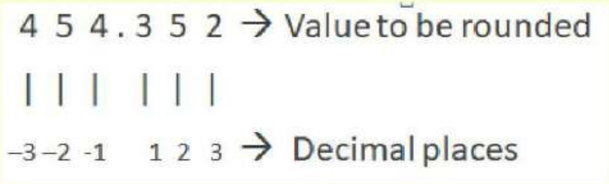
KAPIL SEHGAL

Mathematical Functions

SCALAR FUNCTION OR NON-DATABASE FUNCTIONS

Truncate(nmuber / Column, [Number of Decimal places])

The number of decimal places truncated (deleted) to. This value will be a positive or negative integer or zero.



```
4 5 4 . 3 5 2 → Value to be rounded
| | |   | | |
-3-2 -1  1 2 3 → Decimal places
```

Decimal place position value is TRUNCATED off to the next integer if the next number on the right side is greater than 5 (>=5).

Ex. Second parameter is positive or zero

1. Select truncate(3.798, 1);   Result: 3.7
2. Select truncate(1.298, 0);   Result: 1
3. Select truncate(23.298, 2); Result: 23.29

Ex. Second Parameter is negative

1. Select truncate(27542.29,–1); Result: 27540
2 .Select truncate(27542.29,–2); Result: 27500
3. Select truncate(27542.29, –3); Result: 27000
4. Select truncate(27542.29,–4); Result: 20000
5. Select truncate(77542.29,–5); Result: 0

KAPIL SEHGAL

String Functions

These functions are used to deal with the string type values. The various built-in String library functions are:

ascii(), lower(), upper(), length() instr() left(), right(),
trim(), ltrim(), rtrim() mid(), substring() substr()

ASCII()

Returns the ASCII code value of a character (leftmost character of string).

Syntax: ascii(character);

Ex.

```
select ascii('a') from dual;     returns 97.
select ascii('A') from dual;     returns 65.
select ascii('1') from dual;     returns 49.
select ascii('ABC') from dual; returns 65.
```

LOWER()/ LCASE()

Converts character strings data into lower case.

Syntax: lower(string);

Ex.

```
select lower("INFORMATION TECHNOLOGY");
Returns – information technology
```

String Functions

SCALAR FUNCTION OR NON-DATABASE FUNCTIONS

UPPER()/ UCASE()

Converts character strings data into upper case.

Syntax: upper(string);

Ex.

select lower(“Information");
Returns – INFORMATION

LENGTH()

Returns the length of the character string. It takes spaces between the strings into account for calculating the total length of the string passed as an argument to length().

Syntax: length(string);

Ex.

select length('Information');
Returns – 11

KAPIL SEHGAL

LEFT()

Returns leftmost characters from a string, passed as an argument, with the specified number of characters counting from left. left() function is used to retrieve portions of the string.

Ex. **Syntax: left(string, integer);**

select left('INFORMATION TECHNOLOGY', 6);
Returns – INFORM

RIGHT()

Returns leftmost characters from a string, passed as an argument, with the specified number of characters counting from left. left() function is used to retrieve portions of the string.

Syntax: left(string, integer);

Ex. select right('INFORMATION TECHNOLOGY', 6);
Returns – NOLOGY

String Functions

SCALAR FUNCTION OR NON-DATABASE FUNCTIONS

LTRIM()

Returns a string after removing leading spaces/blanks from the left side of the string passed as an argument.

Syntax: ltrim(string);

Ex.

```
select ltrim(' LIBRARY FUNCTION');
Returns – LIBRARY FUNCTION
```

RTRIM()

Returns a string after removing leading spaces/blanks from the right side of the string passed as an argument.

Syntax: rtrim(string);

Ex.

```
select rtrim('LIBRARY FUNCTION    ');
Returns – LIBRARY FUNCTION
```

KAPIL SEHGAL

TRIM()

Returns a string after removing the spaces from both ends the string passed as the argument to it.

Syntax: trim(string);

Ex.

```
select trim(‘     LIBRARY FUNCTION     ');
Returns – LIBRARY FUNCTION
```

INSTR()

Returns the position of the second string in the first string, if present else return 0.

Syntax: instr(first_string, string_to_search);

Ex.

```
select instr(‘Hello’,’ll’);
Returns – 3
```

```
select instr(‘Hello’,’io’);
Returns – 0
```

String Functions

SCALAR FUNCTION OR NON-DATABASE FUNCTIONS

SUBSTRING()/ SUBSTR()/ MID()

Returns part of an inputted string. A substring() function retrieves a portion of the given string starting at the specified character (start_index) to the number of characters specified (length).

Syntax: substring(string, start_index, length);

Ex.

```
select substring('STRING FUNCTION', 1, 6);
Returns : STRING
```

Ex.

```
select substring('STRING FUNCTION', 8, 4);
Returns :  FUNC
```

Date Functions

SCALAR FUNCTION OR NON-DATABASE FUNCTIONS

These functions are used to deal with date.

Curdate()

Returns the current system date.

Ex. select curdate(); Result: '2020-08-06'

Now()

Returns the current date and time.

Ex, select now(); Result: '2020-06-11 13:58:11'

Sysdate()

Returns the time at which the function executes.

Ex. select sysdate();
Result: '2020-06-11 13:59:23'

Date()

Extracts the date part of a date or date-time expression.

Ex. select date('2020-06-11 01:02:03');
Result: '2020-06-11'

KAPIL SEHGAL

Date Functions

SCALAR FUNCTION OR NON-DATABASE FUNCTIONS

Dayname()

Returns the name of the weekday.

Ex, select dayname('2020-06-11'); Result: THURSDAY

Dayofweek()

Returns the weekday index of the argument.

Ex, select dayofweek('2020-06-11'); Result: 5 (Sunday is counted as 1)

Dayofyear()

Returns the day of the year (1-366).

Ex, select dayofyear('2020-06-11'); Result: 202

Date Functions

SCALAR FUNCTION OR NON-DATABASE FUNCTIONS

Month()

Returns the month from the date passed.

Ex, select month('2020-06-11'); Result: 6

Year()

Returns the year from the inputted date.

Ex, select year('2020-06-11'); Result: 2020

Day()

Returns the day from the inputted date.

Ex, select year('2020-06-11'); Result: 11

GROUP BY

The **GROUP BY clause** is a **SQL** command that is used to **group** rows that have the same values. The **GROUP BY clause** is used in the SELECT statement . Optionally it is used in conjunction with aggregate functions to produce summary reports from the database. That's what it does, summarizing data from the database.

Syntax

```
SELECT c1, c2,..., cn, aggregate_function(ci) FROM table
WHERE where_conditions
GROUP BY c1 , c2,...,cn;
```

"SELECT statements..." is the standard SQL SELECT command query.

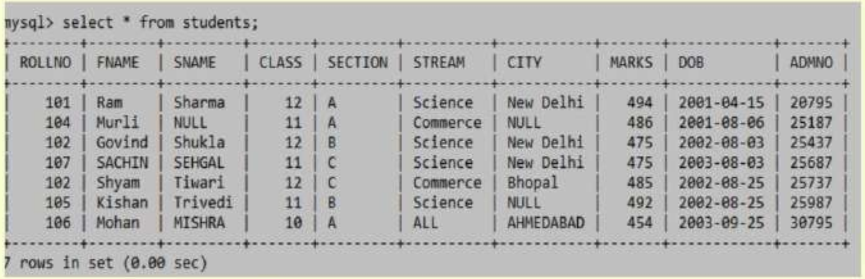
"**GROUP BY** *column_name1*" is the clause that performs the grouping based on column_name1.

"[,column_name2,...]" is optional; represents other column names when the grouping is done on more than one column.

"[HAVING condition]" is optional; it is used to restrict the rows affected by the GROUP BY clause. It is similar to the WHERE clause.

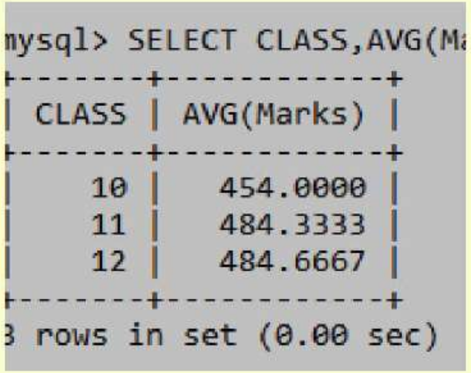
GROUP BY

Find the average marks class wise from following table employee



```
mysql> select * from students;
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
| ROLLNO | FNAME  | SNAME   | CLASS | SECTION | STREAM   | CITY      | MARKS | DOB        | ADMNO |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
|    101 | Ram    | Sharma  |    12 | A       | Science  | New Delhi |   494 | 2001-04-15 | 20795 |
|    104 | Murli  | NULL    |    11 | A       | Commerce | NULL      |   486 | 2001-08-06 | 25187 |
|    102 | Govind | Shukla  |    12 | B       | Science  | New Delhi |   475 | 2002-08-03 | 25437 |
|    107 | SACHIN | SEHGAL  |    11 | C       | Science  | New Delhi |   475 | 2003-08-03 | 25687 |
|    102 | Shyam  | Tiwari  |    12 | C       | Commerce | Bhopal    |   485 | 2002-08-25 | 25737 |
|    105 | Kishan | Trivedi |    11 | B       | Science  | NULL      |   492 | 2002-08-25 | 25987 |
|    106 | Mohan  | MISHRA  |    10 | A       | ALL      | AHMEDABAD |   454 | 2003-09-25 | 30795 |
+--------+--------+---------+-------+---------+----------+-----------+-------+------------+-------+
7 rows in set (0.00 sec)
```

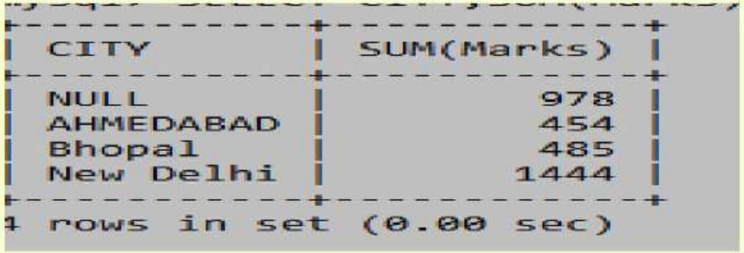
SELECT CLASS, AVG(Marks) FROM STUDENTS GROUP BY CLASS;



```
mysql> SELECT CLASS,AVG(Ma
+-------+------------+
| CLASS | AVG(Marks) |
+-------+------------+
|    10 |   454.0000 |
|    11 |   484.3333 |
|    12 |   484.6667 |
+-------+------------+
3 rows in set (0.00 sec)
```

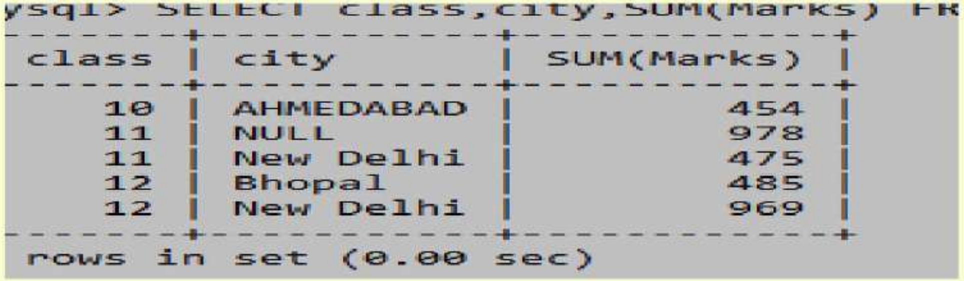
Find the SUM of marks city wise from following table employee

SELECT CITY, SUM(Marks) FROM STUDENTS GROUP BY CITY;



```
+-----------+------------+
| CITY      | SUM(Marks) |
+-----------+------------+
| NULL      |        978 |
| AHMEDABAD |        454 |
| Bhopal    |        485 |
| New Delhi |       1444 |
+-----------+------------+
4 rows in set (0.00 sec)
```

Find the SUM of marks class wise then city wise.



```
ysql> SELECT class,city,SUM(Marks) FR
+-------+-----------+------------+
| class | city      | SUM(Marks) |
+-------+-----------+------------+
|    10 | AHMEDABAD |        454 |
|    11 | NULL      |        978 |
|    11 | New Delhi |        475 |
|    12 | Bhopal    |        485 |
|    12 | New Delhi |        969 |
+-------+-----------+------------+
 rows in set (0.00 sec)
```

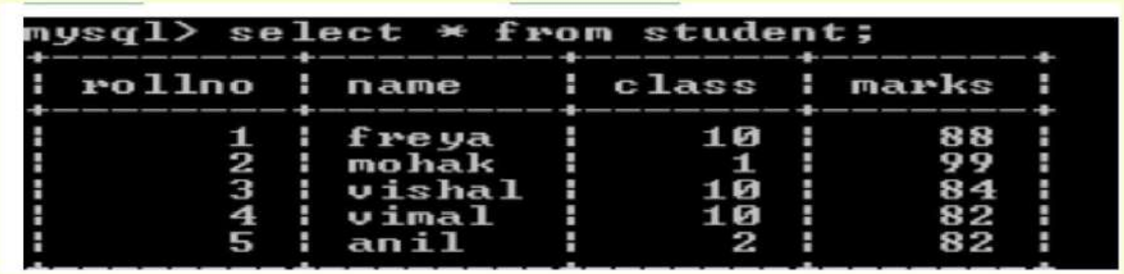
select class,city,sum(marks)
from students group by class,city;

HAVING with Aggregate Function

GROUP BY WITH HAVING

The **HAVING** clause is used in the SELECT statement to specify filter conditions for a group of rows or aggregates. The HAVING clause is often used with the GROUP BY clause to filter groups based on a specified condition. To filter the groups returned by GROUP BY clause, we use a HAVING clause. WHERE is applied before GROUP BY, HAVING is applied after (and can filter on aggregates)

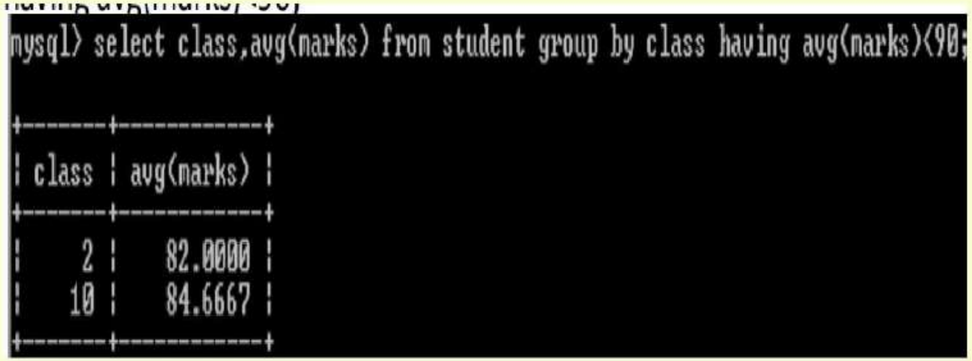
we are having student table with following data.



```
mysql> select * from student;
+--------+---------+-------+-------+
| rollno | name    | class | marks |
+--------+---------+-------+-------+
|      1 | freya   |    10 |    88 |
|      2 | mohak   |     1 |    99 |
|      3 | vishal  |    10 |    84 |
|      4 | vimal   |    10 |    82 |
|      5 | anil    |     2 |    82 |
+--------+---------+-------+-------+
```

KAPIL SEHGAL

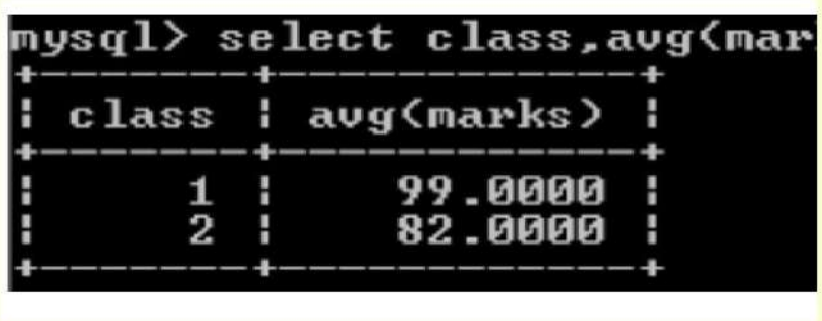
Ex, select class,avg(marks) from student group by class having avg(marks)<90;



```
mysql> select class,avg(marks) from student group by class having avg(marks)<90;

+-------+------------+
| class | avg(marks) |
+-------+------------+
|     2 |    82.0000 |
|    10 |    84.6667 |
+-------+------------+
```

Ex, select class,avg(marks) from student group by class having count(*)<3;



```
mysql> select class,avg(mar
+-------+------------+
| class | avg(marks) |
+-------+------------+
|     1 |    99.0000 |
|     2 |    82.0000 |
+-------+------------+
```

D.D.L. Command

ALTER TABLE

To change a column's definition, use **MODIFY** or **CHANGE** clause along with the ALTER command

Alter Command Can do the following Things

!. Add a Column

2. Delete a Column

3. Rename a Column

4. Increase / Decrease the size of Column

5. Add a Primary key

6. Remove a Primary Key.

7. Change the Type of Column.

KAPIL SEHGAL

Alter Command Can not do when data Present in table

!. Add Primary Key

2. Change the Type of Column

Which Keyword (Clause) used with Alter Command

!. Add

2. Drop

3. Modify

4. Change

D.D.L. Command

ALTER TABLE

Add New Column using "ADD" Keywords

```
ALTER TABLE STUDENTS ADD(MotherName
varchar(25));
```

Removing a Column using "DROP" Keywords

```
ALTER TABLE STUDENTS DROP MotherName ;
```

Change the size of Column city using "MODIFY" Keywords either increase or decrease.

```
ALTER TABLE STUDENTS MODIFY CITY
VARCHAR(25);
```

Some Modification can be done only when table data not present. (i.e. Table is Empty)

Change the Type of Column city using "MODIFY" Keywords either increase or decrease.

```
ALTER TABLE STUDENTS MODIFY CITY int;
```

Add Primary Key

```
ALTER TABLE STUDENTS ADD PRIMARY KEY(Column Name)
```

Remove Primary Key

```
ALTER TABLE STUDENTS DROP PRIMAY KEY.
```

Rename a Column

```
ALTER TABLE STUDENTS CHANGE
OLD_COLUMN NEW_COLUMN DATATYPE;
```

DIFFERENCES

Delete and Drop Command

The following Differences as follows

1. Delete is DML Command While Drop is DDL Command.
2. Delete Command is Deletes the rows from the tables but Drop Command Delete the Entire table with rows. Apart from this Drop command Delete the database and sequence, index etc.
3. Delete Operations can be rolled back(Undone) while Drop Command Operation Can not be rolled back.

Update and Alter Command

The following Differences as follows

1. Update is DML Command While Alter is DDL Command.
2. Update Command modifies in the data (Rows of a tables) while Alter tables changes the design of the table (Make Changes in the structure of the table).

DATABASE - CONSTRAINTS

FOREIGN KEY

A FOREIGN KEY is a key used to link two tables together.

A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.

The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

Syntax

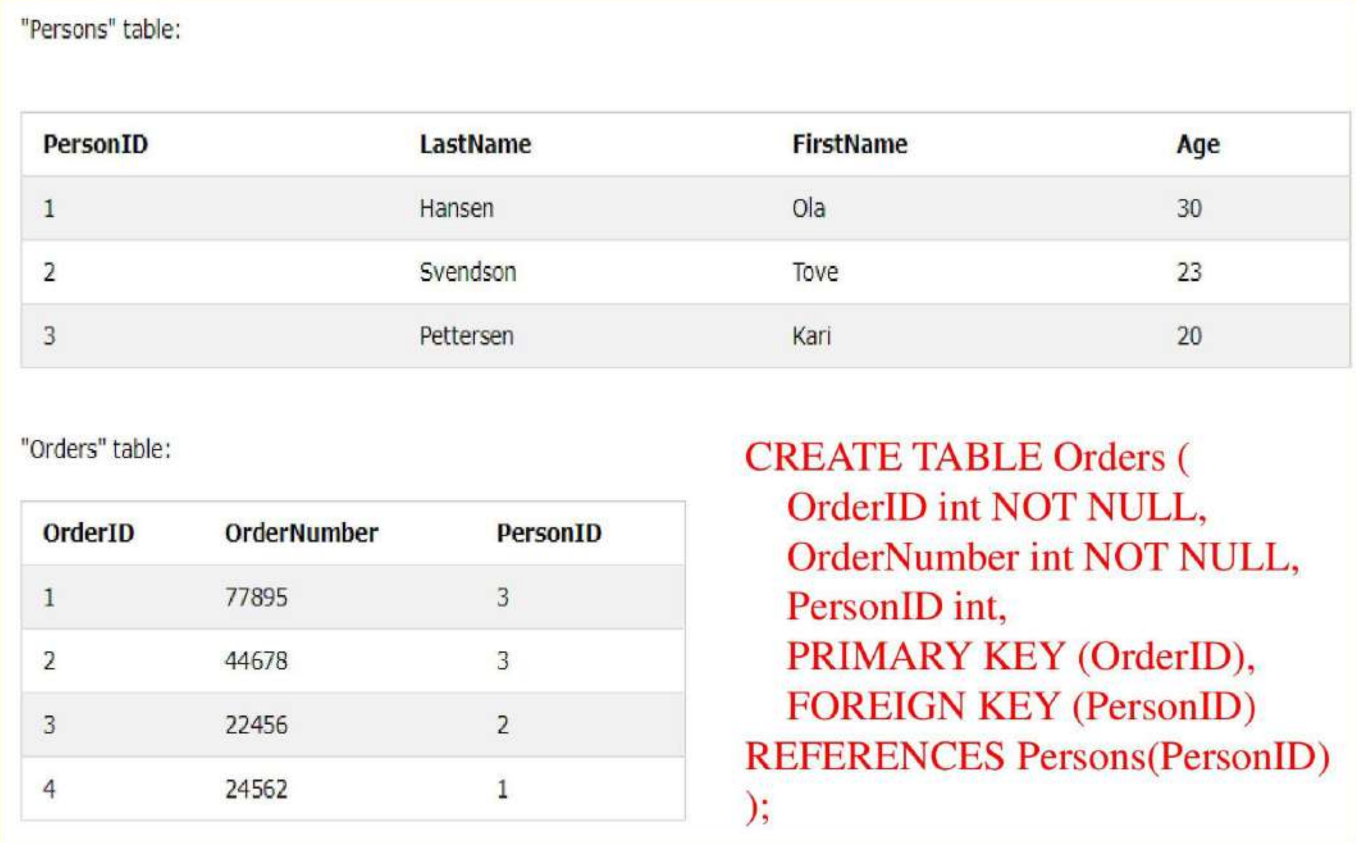
```
FOREIGN KEY (Col-Name) REFERENCES STUDENTS(Primary-Key);
```

Ex.

```
CREATE TABLE FEES (FEESID INT, STUDENTID INT,MONTH INT,YEAR INT,
FOREIGN KEY (STUDENTID) REFERENCES STUDENTS(ADMNO));
```

ONE MORE EXAMPLE

FOREIGN KEY



"Persons" table:

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

"Orders" table:

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

```
CREATE TABLE Orders (
   OrderID int NOT NULL,
   OrderNumber int NOT NULL,
   PersonID int,
   PRIMARY KEY (OrderID),
   FOREIGN KEY (PersonID)
REFERENCES Persons(PersonID)
);
```

Notice that the "PersonID" column in the "Orders" table points to the "PersonID" column in the "Persons" table.

The "PersonID" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.

The "PersonID" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

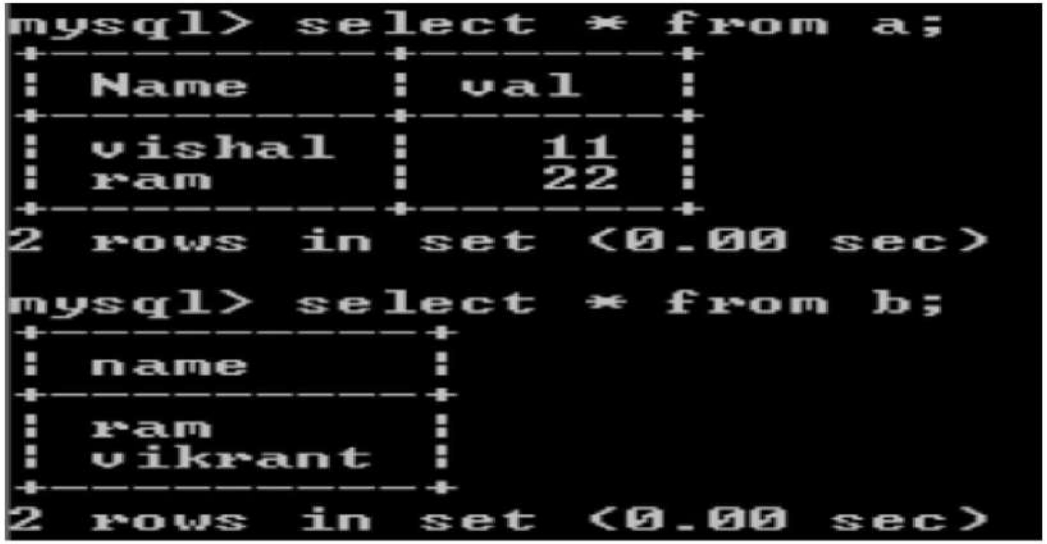
The FOREIGN KEY constraint also prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the table it points to.

KAPIL SEHGAL

CARTESIAN PRODUCT (X) / CROSS JOIN

Cartesian Product is denoted by X symbol. Lets say we have two relations R1 and R2 then the Cartesian product of these two relations (R1 X R2) would combine each tuple of first relation R1 with the each tuple of second relation R2.

Cartesian product (X) example Table a and Table b as shown below



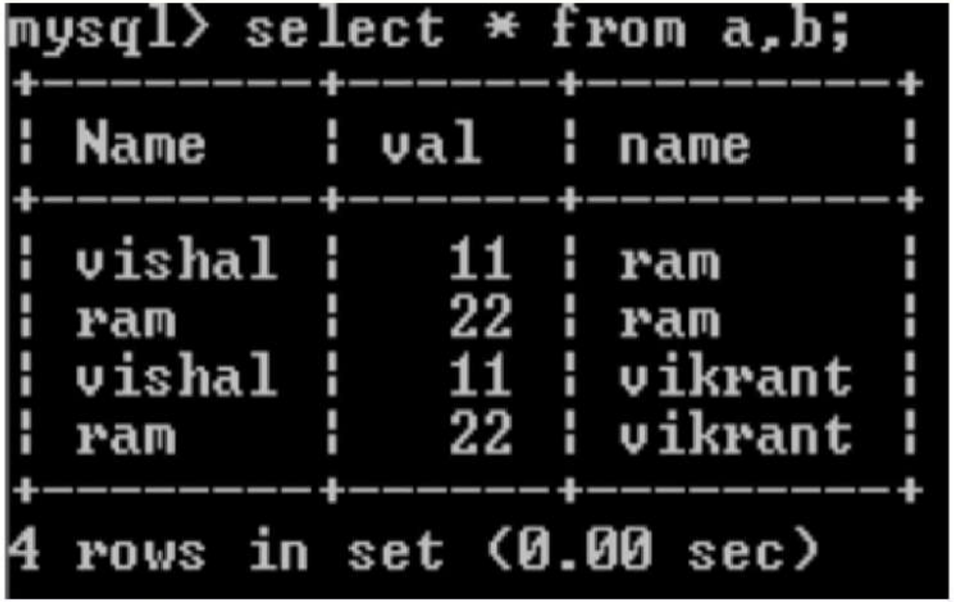
```
mysql> select * from a;
+----------+--------+
| Name     | val    |
+----------+--------+
| vishal   |     11 |
| ram      |     22 |
+----------+--------+
2 rows in set (0.00 sec)

mysql> select * from b;
+-----------+
| name      |
+-----------+
| ram       |
| vikrant   |
+-----------+
2 rows in set (0.00 sec)
```

Select * from a,b;

OR

Select * from a cross join b;



```
mysql> select * from a,b;
+--------+------+---------+
| Name   | val  | name    |
+--------+------+---------+
| vishal |   11 | ram     |
| ram    |   22 | ram     |
| vishal |   11 | vikrant |
| ram    |   22 | vikrant |
+--------+------+---------+
4 rows in set (0.00 sec)
```

Degree of Cartesian product is 3 and cardinality is 4=(2 rows of a X 2 rows of b)

Degree will be add & Cardinality will be Multiply

VARIOUS TYPES OF JOIN

Join – Join is used to fetch data from two or more tables, which is joined to appear as single set of data. It is used for combining column from two or more tables by using values common to both tables.

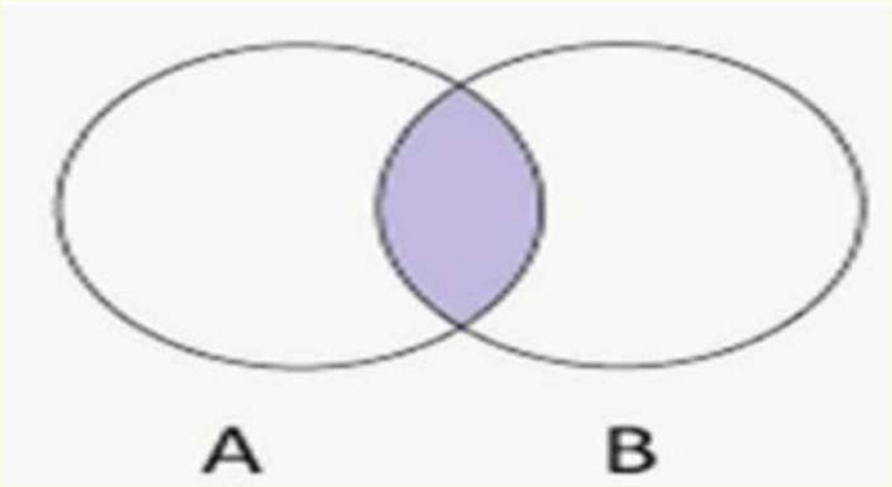
Types of JOIN Following are the types of JOIN that we can use in SQL:

- **Inner**
- **Outer**
- **Left**
- **Right**

JOINS

- **Inner join**

INNER Join or EQUI Join⋈ This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the SQL query.

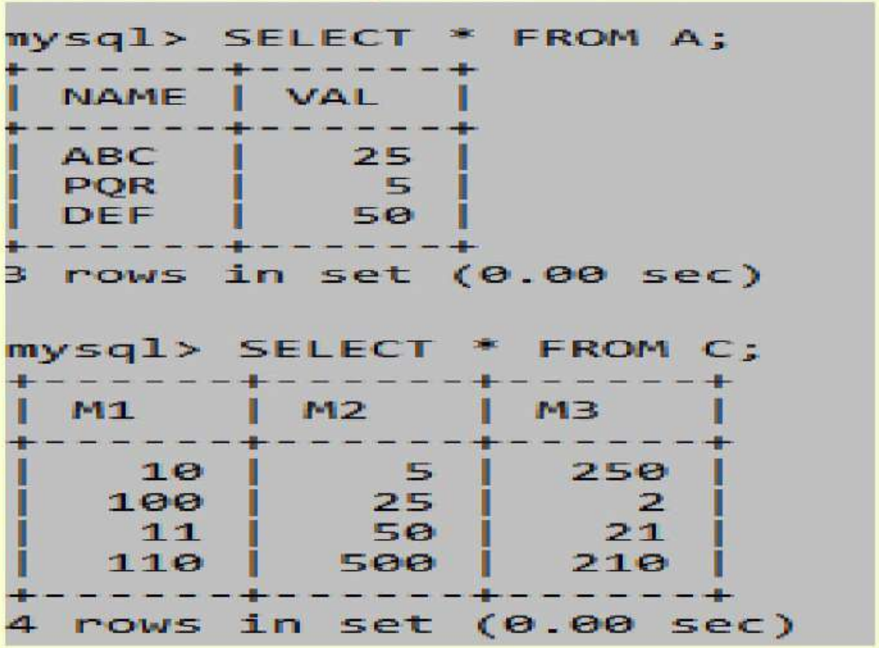


KAPIL SEHGAL

INNER OR EQUI JOIN

An **equi join** is a type of **join** that combines tables based on matching values in specified columns. Please remember that: The column names do not need to be the same. The resultant table contains repeated columns. It is possible to perform an **equi join** on more than two tables.

INNER Join or EQUI Join Table A and Table C



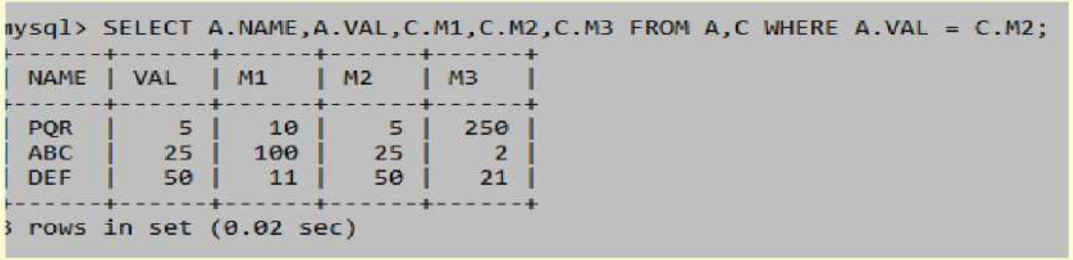
```
mysql> SELECT * FROM A;
+-------+-------+
| NAME  | VAL   |
+-------+-------+
| ABC   |    25 |
| PQR   |     5 |
| DEF   |    50 |
+-------+-------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM C;
+-------+-------+-------+
| M1    | M2    | M3    |
+-------+-------+-------+
|    10 |     5 |   250 |
|   100 |    25 |     2 |
|    11 |    50 |    21 |
|   110 |   500 |   210 |
+-------+-------+-------+
4 rows in set (0.00 sec)
```

SELECT A.NAME,A.VAL,C.M1,C.M2,C.M3 FROM A,C WHERE A.VAL = C.M2;

INNER Join Using "ON" Clause

SELECT A.NAME,A.VAL,C.M1,C.M2,C.M3 FROM A INNER JOIN C ON A.VAL = C.M2;



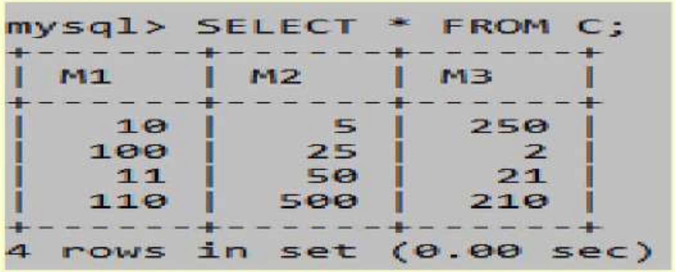
```
mysql> SELECT A.NAME,A.VAL,C.M1,C.M2,C.M3 FROM A,C WHERE A.VAL = C.M2;
+------+------+------+------+------+
| NAME | VAL  | M1   | M2   | M3   |
+------+------+------+------+------+
| PQR  |    5 |   10 |    5 |  250 |
| ABC  |   25 |  100 |   25 |    2 |
| DEF  |   50 |   11 |   50 |   21 |
+------+------+------+------+------+
3 rows in set (0.02 sec)
```

NATURAL JOIN

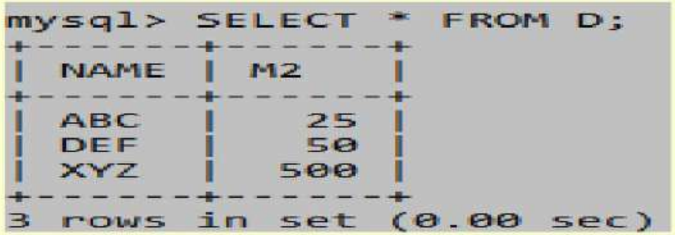
Natural JOIN(⋈) Natural Join is a type of Inner join which is based on column having same name and same data type present in both the tables to be joined .

Ex. Select * from a natural join b;

Table C and **Table D**



```
mysql> SELECT * FROM C;
+--------+--------+--------+
| M1     | M2     | M3     |
+--------+--------+--------+
|     10 |      5 |    250 |
|    100 |     25 |      2 |
|     11 |     50 |     21 |
|    110 |    500 |    210 |
+--------+--------+--------+
4 rows in set (0.00 sec)
```



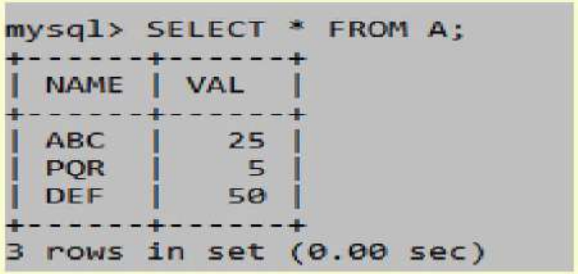
```
mysql> SELECT * FROM D;
+--------+--------+
| NAME   | M2     |
+--------+--------+
| ABC    |     25 |
| DEF    |     50 |
| XYZ    |    500 |
+--------+--------+
3 rows in set (0.00 sec)
```

KAPIL SEHGAL

LIMITATION OF NATURAL JOIN

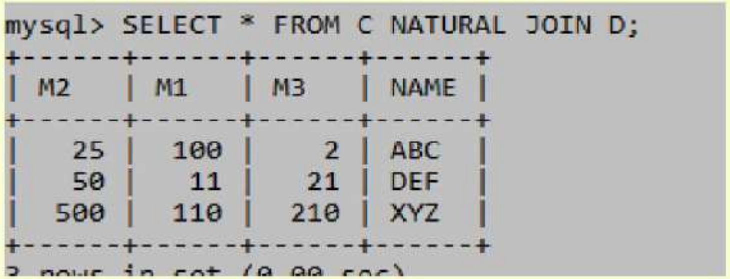
Both Table have one column with same name and same data type.

TABLE : A



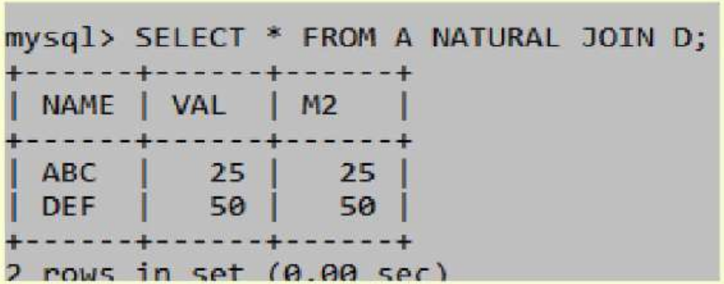
```
mysql> SELECT * FROM A;
+------+------+
| NAME | VAL  |
+------+------+
| ABC  |   25 |
| PQR  |    5 |
| DEF  |   50 |
+------+------+
3 rows in set (0.00 sec)
```

Select * from C natural join D;



```
mysql> SELECT * FROM C NATURAL JOIN D;
+------+------+------+------+
| M2   | M1   | M3   | NAME |
+------+------+------+------+
|   25 |  100 |    2 | ABC  |
|   50 |   11 |   21 | DEF  |
|  500 |  110 |  210 | XYZ  |
+------+------+------+------+
3 rows in set (0.00 sec)
```

Select * from A natural join D;



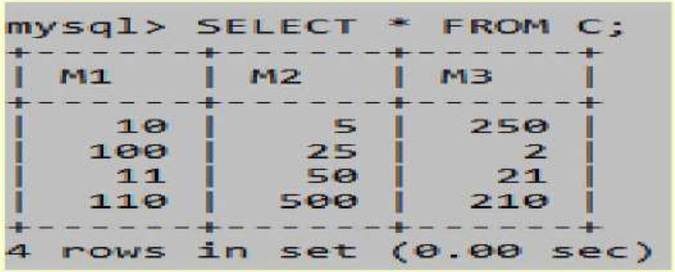
```
mysql> SELECT * FROM A NATURAL JOIN D;
+------+------+------+
| NAME | VAL  | M2   |
+------+------+------+
| ABC  |   25 |   25 |
| DEF  |   50 |   50 |
+------+------+------+
2 rows in set (0.00 sec)
```

LEFT OUTER JOIN

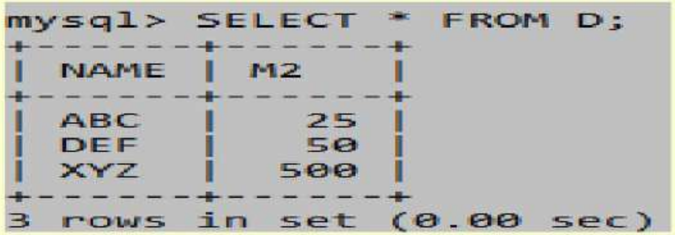
The left outer join returns a result set table with the matched data from the two tables and then the remaining rows of the left table and null from the right table's columns..

Ex. Select * from a left outer join b;

Table C and **Table D**



```
mysql> SELECT * FROM C;
+--------+--------+--------+
| M1     | M2     | M3     |
+--------+--------+--------+
|     10 |      5 |    250 |
|    100 |     25 |      2 |
|     11 |     50 |     21 |
|    110 |    500 |    210 |
+--------+--------+--------+
4 rows in set (0.00 sec)
```

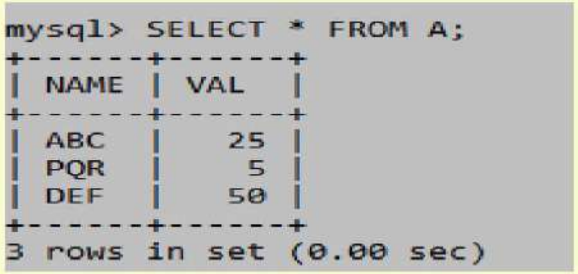


```
mysql> SELECT * FROM D;
+--------+--------+
| NAME   | M2     |
+--------+--------+
| ABC    |     25 |
| DEF    |     50 |
| XYZ    |    500 |
+--------+--------+
3 rows in set (0.00 sec)
```

LIMITATION OF LEFT OUTER JOIN

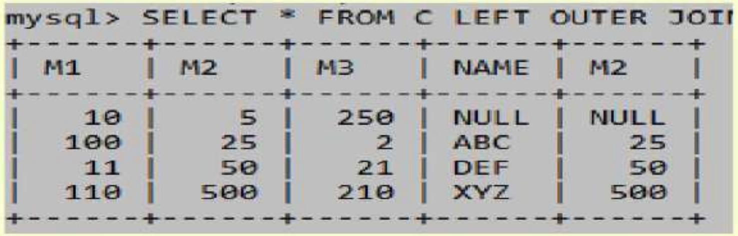
Result table contains only number of row as the left table.

TABLE : A



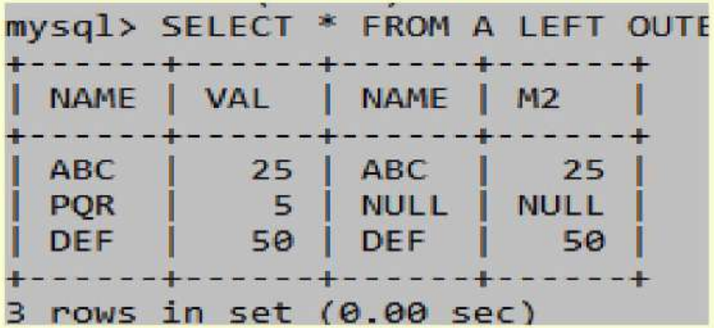
```
mysql> SELECT * FROM A;
+------+-----+
| NAME | VAL |
+------+-----+
| ABC  |  25 |
| PQR  |   5 |
| DEF  |  50 |
+------+-----+
3 rows in set (0.00 sec)
```

SELECT * FROM C LEFT OUTER JOIN D ON C.M2 = D.M2;



```
mysql> SELECT * FROM C LEFT OUTER JOI
+------+------+------+------+------+
| M1   | M2   | M3   | NAME | M2   |
+------+------+------+------+------+
|   10 |    5 |  250 | NULL | NULL |
|  100 |   25 |    2 | ABC  |   25 |
|   11 |   50 |   21 | DEF  |   50 |
|  110 |  500 |  210 | XYZ  |  500 |
+------+------+------+------+------+
```

SELECT * FROM A LEFT OUTER JOIN D ON A.NAME = D.NAME;



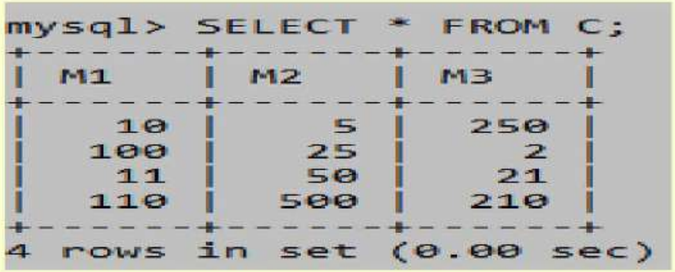
```
mysql> SELECT * FROM A LEFT OUTE
+------+-----+------+------+
| NAME | VAL | NAME | M2   |
+------+-----+------+------+
| ABC  |  25 | ABC  |   25 |
| PQR  |   5 | NULL | NULL |
| DEF  |  50 | DEF  |   50 |
+------+-----+------+------+
3 rows in set (0.00 sec)
```

RIGHT OUTER JOIN

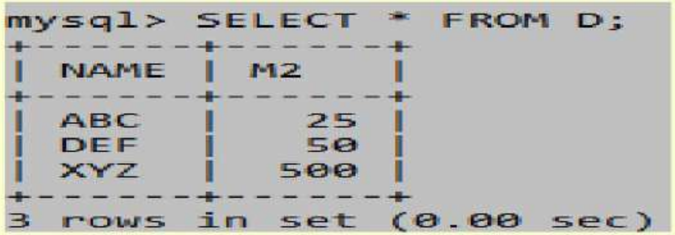
RIGHT JOIN: **RIGHT JOIN** is similar to LEFT **JOIN**. This **join** returns all the rows of the table on the **right** side of the **join** and matching rows for the table on the left side of **join**. The rows for which there is no matching row on left side, the result-set will contain null. **RIGHT JOIN** is also known as **RIGHT OUTER JOIN**

Ex. Select * from a right outer join b;

Table C and **Table D**



```
mysql> SELECT * FROM C;
+-------+-------+-------+
| M1    | M2    | M3    |
+-------+-------+-------+
|    10 |     5 |   250 |
|   100 |    25 |     2 |
|    11 |    50 |    21 |
|   110 |   500 |   210 |
+-------+-------+-------+
4 rows in set (0.00 sec)
```

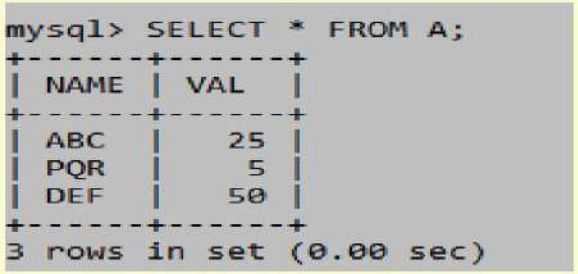


```
mysql> SELECT * FROM D;
+-------+-------+
| NAME  | M2    |
+-------+-------+
| ABC   |    25 |
| DEF   |    50 |
| XYZ   |   500 |
+-------+-------+
3 rows in set (0.00 sec)
```

LIMITATION OF RIGHT OUTER JOIN

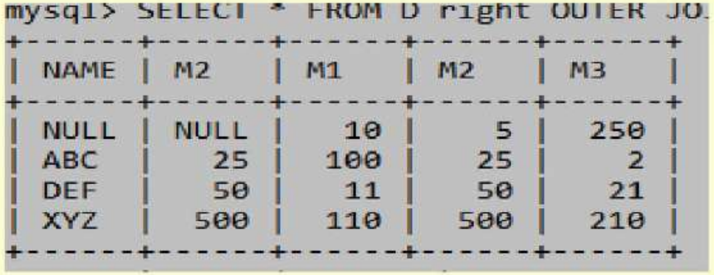
Result table contains only number of row as the RIGHT table.

TABLE : A



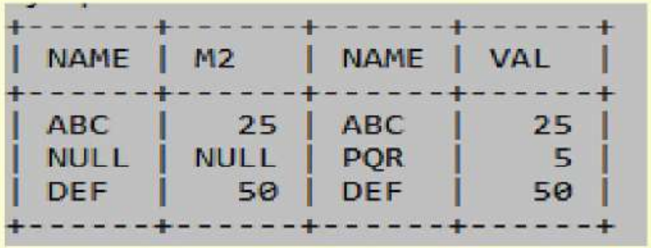
```
mysql> SELECT * FROM A;
+------+-----+
| NAME | VAL |
+------+-----+
| ABC  |  25 |
| PQR  |   5 |
| DEF  |  50 |
+------+-----+
3 rows in set (0.00 sec)
```

SELECT * FROM D right OUTER JOIN C ON C.M2 = D.M2;



```
mysql> SELECT * FROM D right OUTER JO
+------+------+------+------+------+
| NAME | M2   | M1   | M2   | M3   |
+------+------+------+------+------+
| NULL | NULL |   10 |    5 |  250 |
| ABC  |   25 |  100 |   25 |    2 |
| DEF  |   50 |   11 |   50 |   21 |
| XYZ  |  500 |  110 |  500 |  210 |
+------+------+------+------+------+
```

SELECT * FROM D RIGHT OUTER JOIN A ON A.NAME = D.NAME;



```
+------+------+------+------+
| NAME | M2   | NAME | VAL  |
+------+------+------+------+
| ABC  |   25 | ABC  |   25 |
| NULL | NULL | PQR  |    5 |
| DEF  |   50 | DEF  |   50 |
+------+------+------+------+
```

PYTHON-MYSQL CONNECTIVITY

MYSQL CONNECTOR

- MySQL connector is an interface for connecting to a MySQL database server from Python.
- It implements the Python Database API and is built on top of the MySQL C API.

import mysql.connector

OR

Import mysql.connector as con

WHAT IS CONNECTION

The next step to using MySQL in your Python scripts is to make a connection to the database that you wish to use. All Python DB-API modules implement a function

'module_name.connect()‘

This is the function that is used to connect to the database, in our case MySQL..

Steps to use mysql-connector

1. Download Mysql API ,exe file and install it.
2. Install Mysql-Python Connector (Open command prompt and execute command) >pip install mysql-connector
3. Now connect Mysql server using python
4. Write python statement in python shell import mysql.connector If no error message is shown means mysql connector is properly installed

KAPIL SEHGAL

PYTHON-MYSQL CONNECTIVITY

CURSOR

CREATE CURSOR

The next step is to create a Cursor object. It will let you execute all the queries you need. In order to put our new connection to good use we need to create a cursor object.

The cursor object is an abstraction specified in the Python DB-API

It gives us the ability to have multiple separate working environments through the same connection to the database.

We can create a cursor by executing the 'cursor' function of your database object.

Ex.

mycursor = mydb.cursor()

KAPIL SEHGAL

How to create cursor object and use it

```
import mysql.connector

mydb=mysql.connector.connect(host="localhost",user="root",passwd="root")

mycursor=mydb.cursor()

mycursor.execute("create database if not exists school")

mycursor.execute("show databases")

for x in mycursor:

print(x)
```

EXAMPLES

PYTHON-MYSQL CONNECTIVITY

Simple code to connect to MySQL

```
import mysql.connector as con
mydb=con.connect(host=‘localhost’,
user=‘root’, passwd=‘rootj’)
mycursor=mydb.cursor()
sql=‘show databases’
mycursor.execute(sql)

myresult     =     mycursor.fetchall()

for data in myresult:
print(data)
mydb.close()
```

Some function and data

We can use fetchone() method to fetch single record and fetchall() method to fetch multiple values from a database table.

fetchone() − It fetches the next row of a query result set. A resultset is an object that is returned when a cursor object is used to query a table.

fetchall() − It fetches all the rows in a result set. If some rows have already been extracted from the resultset, then it retrieves the remaining rows from the resultset.

rowcount − This is a read-only attribute and returns the number of rows that were affected by an execute() method.

How to create table at run time

Table creation is very easy ,if we are already well versed in sql table creation then we have to just pass the create table query in execute() method of cursor object. But before table creation we must open the data base. Here we are opening database school(through connect() method) before student table creation

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root",dat abase="school")
mycursor=mydb.cursor()
mycursor.execute("create table student(rollno int(3) primary key,name varchar(20),age int(2))")
```

On successful execution of above program a table named student with three fields rollno, name, age will be created in school database. We can check student table in mysql shell also if required.

How to insert record in a table at run time

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root",databa se="school")
mycursor=mydb.cursor()
while 1==1:
ch=int(input("enter -1 to exit any other no to insert record into student table"))
if ch==-1: break rollno=int(input("Enter rollno"))
class1=int(input("Enter Class"))
name=input("Enter name")
marks=int(input("Enter marks"))
mycursor.execute("insert into student
values('"+str(rollno)+"','"+name+"','"+str(class1)+"','"+str(marks)+"')") mydb.commit()
```

EXAMPLES

PYTHON-MYSQL CONNECTIVITY

How to change table structure/(add, edit, remove column of a table) at run time

To modify the structure of the table we just have to use alter table query. Below program will add a column mark in the student table.

```
import mysql.connector

mydb=mysql.connector.connect(host="localhost",user="root",passwd ="root",database="school")

 mycursor=mydb.cursor()

mycursor.execute("alter table student add (marks int(3))")

mycursor.execute("desc student") for x in mycursor: print(x)
```

Above program will add a column marks in the table student and will display the structure of the table

How to search records of a table at run time

Statement demonstrate the use of select query for searching specific record from a table.

```
import mysql.connector

mydb=mysql.connector.connect(host="localhost",user="root",passwd="ro ot",database="school")

mycursor=mydb.cursor()

nm=input("enter name") mycursor.execute("select * from student where name='"+nm+"'")

for x in mycursor:

    print (x)
```

EXAMPLES

PYTHON-MYSQL CONNECTIVITY

How to fetch all records of a table at run time

```
import mysql.connector

mydb=mysql.connector.connect(host="localhost",user
="root",passwd="ro ot",database="school")

mycursor=mydb.cursor()

mycursor.execute("select * from student")

myrecords=mycursor.fetchall()

for x in myrecords:

    print (x)
```

KAPIL SEHGAL

How to fetch one record of a table at run time

```
import mysql.connector

mydb=mysql.connector.connect(host="localhost",
user="root",passwd="root",database="school")

mycursor=mydb.cursor()

mycursor.execute("select * from student")

row=mycursor.fetchone()

while row is not None:
    print(row)
    row = mycursor.fetchone()
```

MySQLCursor.fetchone() Method

This method retrieves the next row of a query result set and returns a single sequence, or None if no more rows are available. By default, the returned tuple consists of data returned by the MySQL server, converted to Python objects.

MySQLCursor.fetchmany() Method

```
rows = cursor.fetchmany(size=1)
```

This method fetches the next set of rows of a query result and returns a list of tuples. If no more rows are available, it returns an empty list.

EXAMPLES

PYTHON-MYSQL CONNECTIVITY

rowcount

Rows affected by Query. We can get number of rows affected by the query by using rowcount. We will use one SELECT query here.

```
import mysql.connector
mydb=mysql.connector.connect(host="localhost",user="root",passwd="root",database="s chool")
mycursor=mydb.cursor()
mycursor = mydb.cursor(buffered=True) mycursor.execute("select * from student")
noofrows=mycursor.rowcount print("No of rows in student table are",noofrows)
```

buffered=True
We have used my_cursor as buffered cursor.
my_cursor = my_connect.cursor(buffered=True)
This type cursor fetches rows and buffers them after getting output from MySQL database. We can use such cursor as iterator. There is no point in using buffered cursor for single fetching of rows.If we don’t use buffered cursor then we will get -1 as output from rowcount

EXAMPLES

PYTHON-MYSQL CONNECTIVITY

How to delete record of a table at run time import

```
mysql.connector

mydb=mysql.connector.connect(host="localhost",user="root",passwd="root",database="sc hool")

mycursor=mydb.cursor()

mycursor.execute("delete from student where rollno=1")

mydb.commit()
```

In above program delete query will delete a record with rollno=1.commit() method is necessary to call for database transaction.

EXAMPLES

PYTHON-MYSQL CONNECTIVITY

How to update record of a table at run time

```
import mysql.connector

mydb=mysql.connector.connect(host="localhost",user="root",passwd="root",database="sc hool")

mycursor=mydb.cursor()

mycursor.execute("update student set marks=99 where rollno=2")

mydb.commit().
```

In above program update query update the marks with 99 of rollno=2 Students are advised to develop menu driven program using above concepts for better understating of python mysql database interface