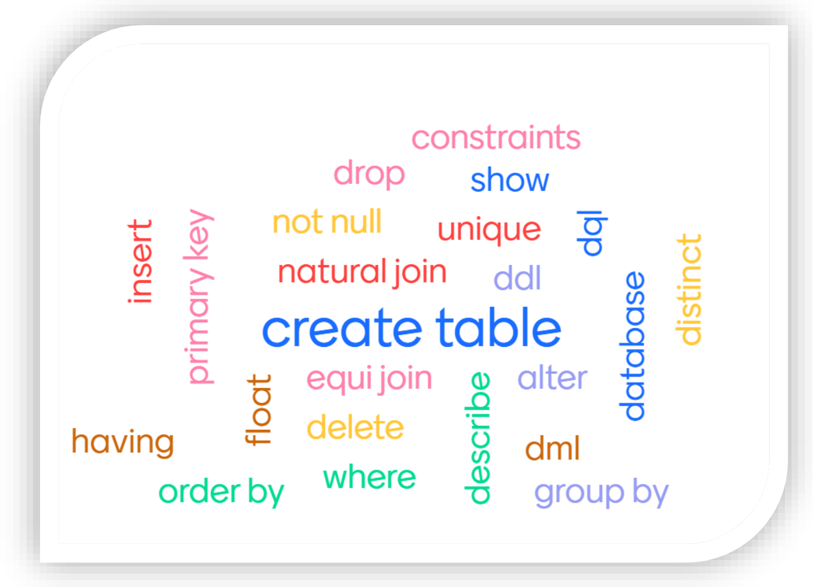
Database Management Systems



Topics to be covered

1. Database concepts
2. Relational data model
3. Structured Query Language
4. Interface of python with an sql database

UNIT-III

DATABASE MANAGEMENT:

Database Concept:

Introduction of Database:

Database is a word which composed of two words Data and Base. Data means raw facts and figures and base is place or location where data is being stored.

Or we can say that Database is collection of **interrelated data or record** in organized form so it can easily be accessed, managed and updated**. Interrelated data** means that the data is related or connected to each other with respect to the given attributes or column.

Database uses various fields to manage and store large amounts of information in organized and structured format

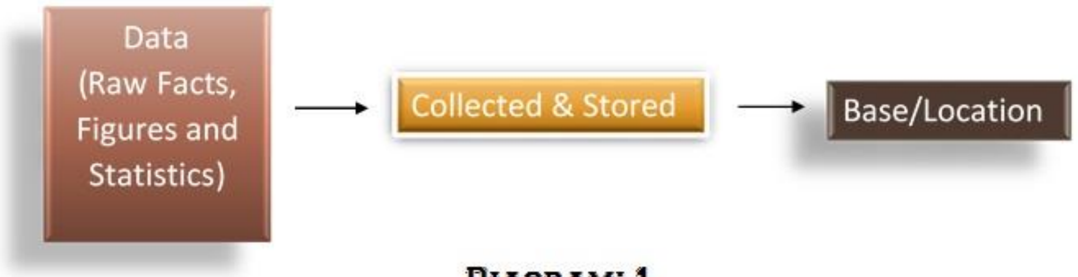


DIAGRAM: 1
HOW DATABASE WORKS

Introduction of Database Management System (DBMS):

DBMS is shorten name used for Database Management System. So, as we are aware of database now, we need to understand what DBMS is. Let's understand it.

DBMS is a software system which is used to manage Database. DBMS acts as an interface between a user and database which enables the user to create, insert, retrieve, update and delete the data.

Need of Database:

Centralized Storage: Storage of data in a single location or central database.

Data Integrity: Enforces data integrity rules which ensures that information stored is accurate, valid and consistent.

Data Security: Control access to sensitive data and protecting data from unauthorized access.

Data Retrieval: Authorized User/Application can access and retrieve the information as per their need.

Efficient Data Retrieval: Database helps user to retrieve data in an efficient way.

DBMS Model:

DBMS refers to that architecture/approach for how data is stored, organized and manipulated in a database. There are several types of DBMS Model.

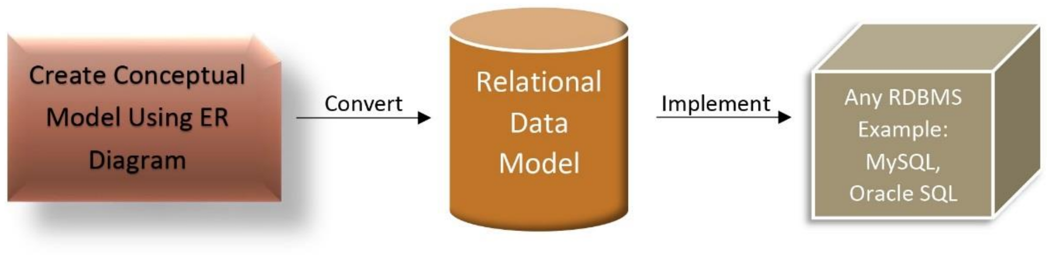
1. **Relational Model:**
   Data organized into tables with rows and columns.
2. **Hierarchical Model:**
   Data organized in Tree like structure with parent child relationship
3. **Network Model**:
   Similar to hierarchical model. It uses pointers to navigate through data.
4. **Object Oriented Model:**
   Data is represented as object. This model uses object oriented databases.

Now as per you CBSE syllabus we will discuss about Relational Data Model in detail.

Relational Data Model:

Relation Data Model is proposed by E.F. Codd in 1970.

In simple words, we can say that Relational data model is a model which uses **relation** to organize their **data**. Here, **Relation means table** and table is composed of **rows** and **columns**.



Create Conceptual Model Using ER Diagram
Convert
Relational Data Model
Implement
Any RDBMS Example: MySQL, Oracle SQL

DIAGRAM: 2

ABOVE DIAGRAM SHOWS THAT HOW TO IMPLEMENT RELATIONAL DATA MODEL

After creating conceptual data model using ER Diagram we need to convert it into Relational Data Model so that we can implement using any RDBMS language like MySQL, Oracle SQL.

Before we proceed further, let's discuss about some aspects of Conceptual Data Model

Conceptual Data Model is used to capture the meaning of data from the viewpoint of the user and try to represent it using Data Model tools like ER Diagram and Object Oriented Diagram

Basic Terminology of ER Model:

ER stands for **Entity-Relationship Model**. ER Model tools are used to represent Conceptual Data Model. Let's see some basic tools of ER Model.

Entity	Any real world object is known as entity like person, place, object and event	▭
Relationship	Show interconnection between two or more entity	◇
Attribute	Shows property/characteristics of an entity/relationship type	⬭

Relation/Table:

Relation is also known table. And table is a collection of related data and information in row and column. Relation is composed of rows and columns.

Row/Tuple/Record:

Row represent horizontal form of Table/Relation. Row is also known as tuple/record.

Column/Attributes:

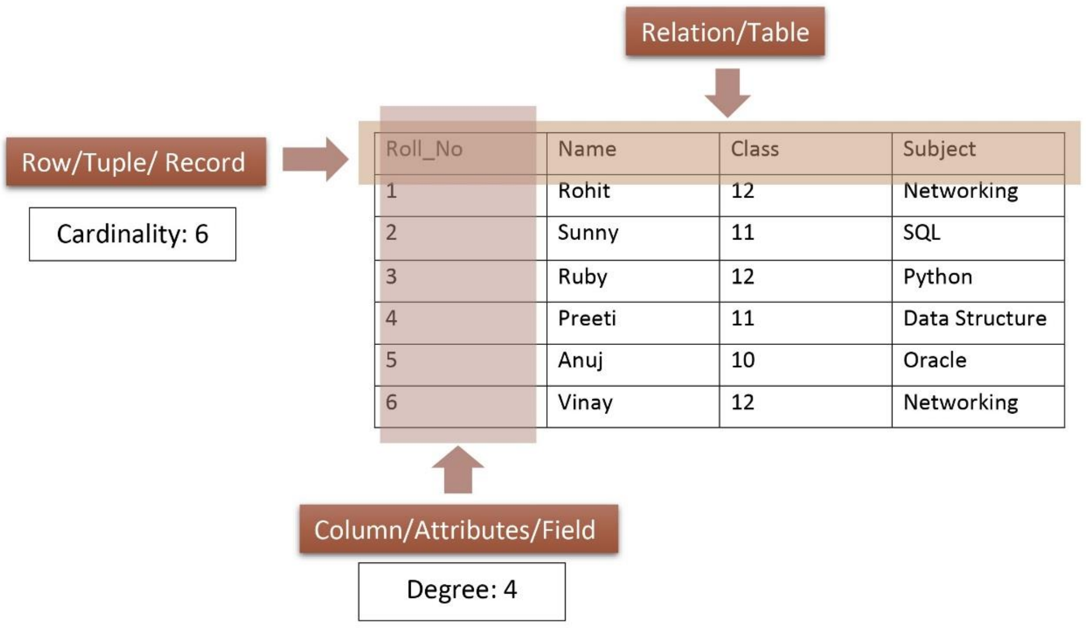
Column represent vertical form of Table/Relation. Column is also known as attributes.

Cardinality:

Total number of row/record/tuple in a relation is called as Cardinality.

Degree:

Total number of column/attributes in a relation is called as Degree.



Roll_No	Name	Class	Subject
1	Rohit	12	Networking
2	Sunny	11	SQL
3	Ruby	12	Python
4	Preeti	11	Data Structure
5	Anuj	10	Oracle
6	Vinay	12	Networking

Diagram: 3

Relation/Table composed of row/record and column/attributes

Domain:

Domain is **set of possible value** or **range of valid values** or **set of all unique values** that an attribute/column can hold.

A Domain of database is set of atomic value (which can't further be distributed) of a particular attribute/column.

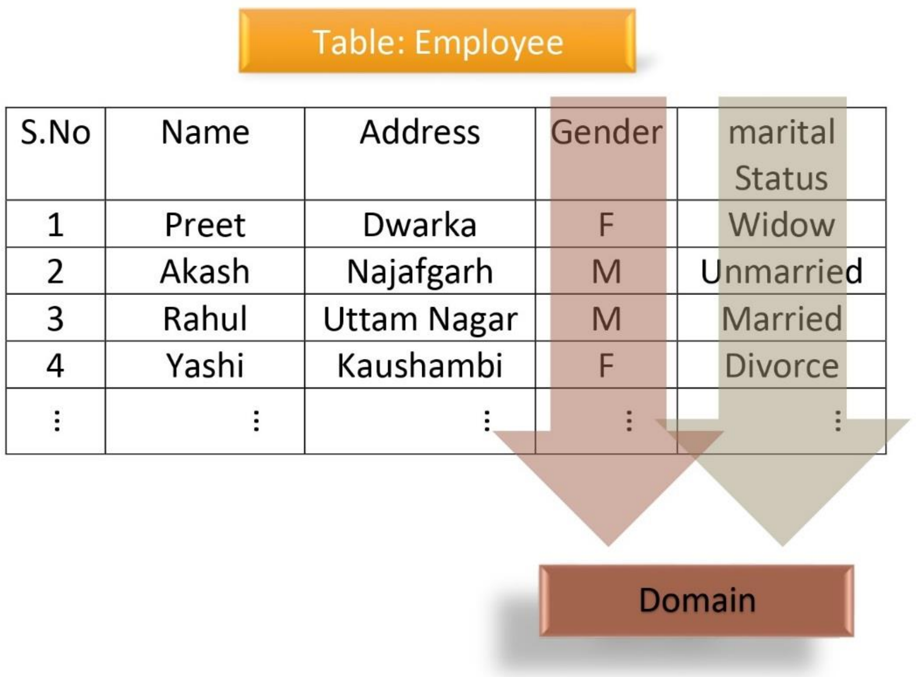
For example:

In table Employee, Attribute/column gender may have only M, F and T Domain. Only these value valid for that column.

Domain of S.No. Contain set of all possible roll numbers.

Domain of Martial Status contain set of all possible values like Married, Widow, Unmarried and Divorce.

In the below diagram 4, Table Employee contain S.No, Name, Address, Gender and Marital status. Two domain is showing name gender and marital status which contain set of possible values that an attribute can hold. Gender can only hold three possible values and marital status can only hold four possible values



S.No	Name	Address	Gender	marital Status
1	Preet	Dwarka	F	Widow
2	Akash	Najafgarh	M	Unmarried
3	Rahul	Uttam Nagar	M	Married
4	Yashi	Kaushambi	F	Divorce
⋮	⋮	⋮	⋮	⋮

Diagram: 4

Domain in a Relation

DATATYPES IN SQL:

Before discussing commands in detail we need to learn about datatype of column/attribute:

We need to assign datatype when we are declaring any column/attributes. Every column required name and datatype. This datatypes is used to declare what type of data that will be stored in particular column. There are lots of datatypes available in SQL we will discuss some important datatype.

Commonly used datatype in SQL:

1. Numeric Type:

- INT: Integer type
- FLOAT: Floating-point number
- DECIMAL or NUMERIC: Fixed-point number

2. Character String Type:

- CHAR(n): Fixed-length character string with maximum length of n
- VARCHAR(n): Variable-length character string with maximum length of n
- TEXT Type : Variable-length character string with no specified maximum length

3. Data and Time Type:

- DATE : for date only
- TIME: for time only
- DATETIME or TIMESTAMP: for date and time combined

4. Other Data type:

- NULL: to represent a missing/unknown/empty value
- ENUM: A enumeration type for a set of predefined values

Now let's learn datatype in detailed as per your syllabus

NUMERIC DATATYPE	INT/INTEGER	Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295
	SMALLINT(SIZE)	SIGNED RANGE IS FROM -32768 TO 32767. UNSIGNED RANGE IS FROM 0 TO 65535
	TINYINT(SIZE)	SIGNED RANGE IS FROM -128 TO 127.UNSINED RANGE IS FROM 0 TO 255
	MEDIUMINT(SIZE)	SIGNED RANGE IS FROM -8388608 TO 8388607. UNSIGNED RANGE IS FROM 0 TO 16777215
	BIGINT(SIZE)	Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615
	FLOAT(SZ,D)	SZ IS SIZE AND D IS NO. OF DIGIT AFTER DECIMAL
STRING DATATYPE	CHAR(SIZE)	CHAR IS FIXED SIZED STRING AS PER SIZE DEFINED IN PARANTHESIS. SIZE OF CHAR DATATYPE IN RANGE OF 0 TO 255. BYDEFAULT SIZE OF CHAR IS 1
	VARCHAR(SIZE)	VARCHAR STANDS FOR VARIABLE LENGTH CHARACTER STRING RANGE OF VARCHAR IS 0 TO 65535

DATE AND TIME DATATYPE	DATE	AS THE NAME SUGGEST IT IS USED TO STORE DATE IN ANY ATTRIBUTE SUPPORTED FORMAT : YYYY-MM-DD
	TIME	USED TO STORE TIME IN ANY ATTRIBUTE SUPPORTED FORMAT : HH:MM:SS

Keys:

In database, keys is column/attribute which is used to fetch/extract/retrieve row in a table. Or we can say that Keys are used to uniquely identify records in a table through column or combination of column. To extract any particular row/record from a table, we need a key attribute which contain unique values.

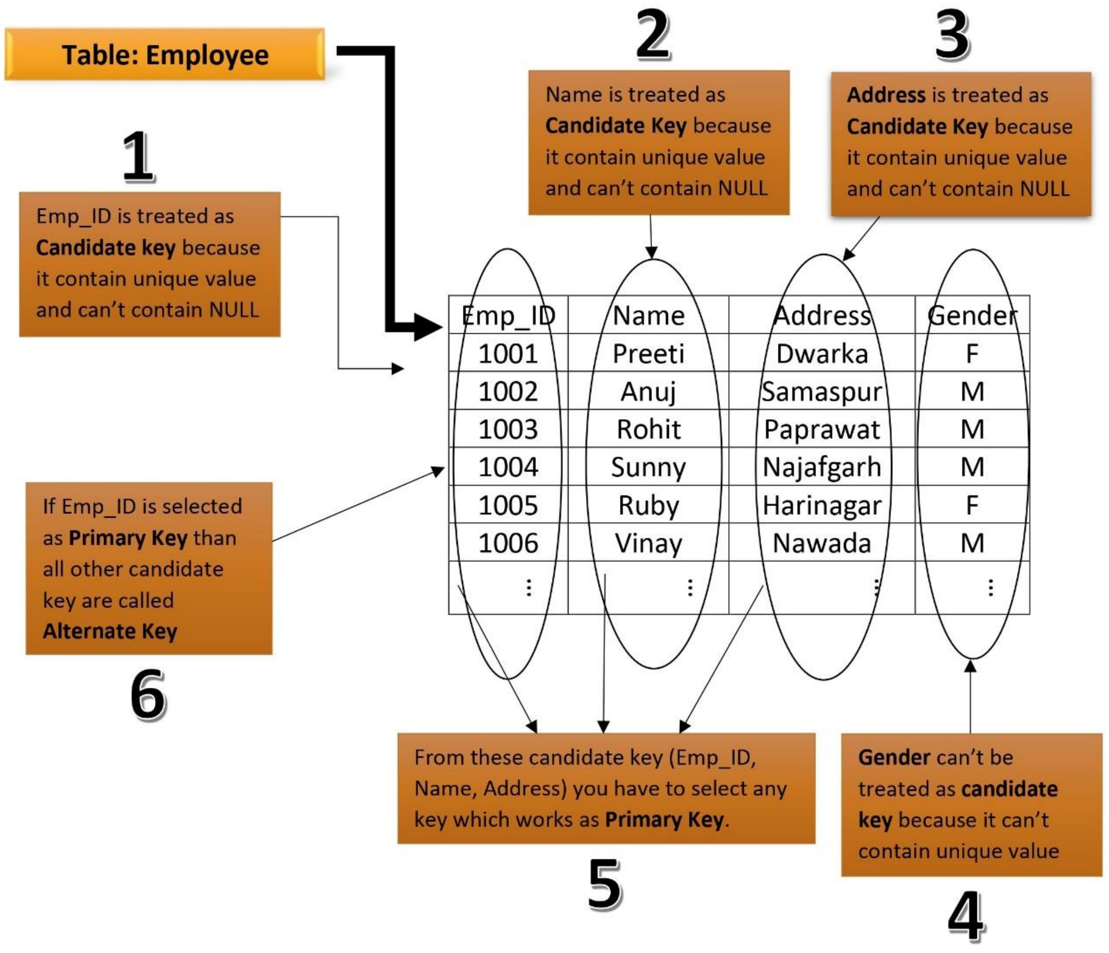


Table: Employee
1
Emp_ID is treated as **Candidate key** because it contain unique value and can't contain NULL
2
Name is treated as **Candidate Key** because it contain unique value and can't contain NULL
3
Address is treated as **Candidate Key** because it contain unique value and can't contain NULL
4
Gender can't be treated as **candidate key** because it can't contain unique value
5
From these candidate key (Emp_ID, Name, Address) you have to select any key which works as **Primary Key**.
6
If Emp_ID is selected as **Primary Key** than all other candidate key are called **Alternate Key**

Emp_ID	Name	Address	Gender
1001	Preeti	Dwarka	F
1002	Anuj	Samaspur	M
1003	Rohit	Paprawat	M
1004	Sunny	Najafgarh	M
1005	Ruby	Harinagar	F
1006	Vinay	Nawada	M
⋮	⋮	⋮	⋮

DIAGRAM: 5

KEYS IN TABLE (PLEASE READ IN ORDER OF NUMBER)

Primary Key:

Primary Key is a unique identifier which identify unique record in a particular table. It must contain unique values for each record. And Primary key attribute/column/field can't be NULL. A table can have only ONE primary key

Note: A Primary key must be a candidate key but not all candidate key are Primary key

Candidate Key:

Candidate key are those key which are eligible for primary key and can be used as primary key in a table.

Candidate key is a set of one or more column that could be used as primary key and from the set of these candidate key, one column is selected as primary key.

From Diagram 5, Candidate key can have more than one attribute like Emp_ID, Name, and Address etc.

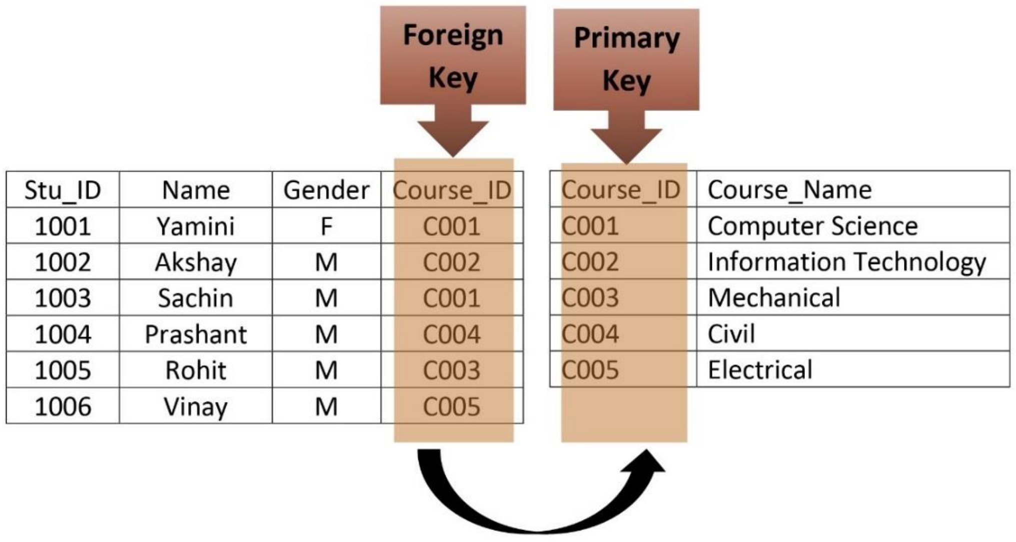
Alternate Key:

After selecting primary key from candidate key, the remaining keys (which are also eligible for primary key) are called Alternate Key.

Foreign Key:

A Foreign key is a column or group of columns in a table that provides a link between two tables.

Let's see how actually foreign key work in table.



Stu_ID	Name	Gender	Course_ID
1001	Yamini	F	C001
1002	Akshay	M	C002
1003	Sachin	M	C001
1004	Prashant	M	C004
1005	Rohit	M	C003
1006	Vinay	M	C005

Course_ID	Course_Name
C001	Computer Science
C002	Information Technology
C003	Mechanical
C004	Civil
C005	Electrical

DIAGRAM: 6

HOW FOREIGN KEY WORKS BETWEEN TWO TABLES

It is an attribute whose values can be derived from primary key of some other table.

It ensure referential integrity. Referential Integrity is protocol which is used to ensure that relationship between record/row is valid and can't change in related data.

Structured Query Language:

Introduction:

SQL is shortened of Structured Query Language. And it is pronounced as See-Quell. SQL is used to manage database. SQL was developed in 1970 in IBM Laboratory and it became a standard of the ANSI (American National Standard Institute) in 1986.

SQL is query language not a database system. You are required to install DBMS software in your system to perform SQL Language operation with help to query Example – Oracle, MySQL, MongoDB, PostgreSQL, SQL Server, DB2 etc.

SQL is mainly used for maintaining the data in relational database management system. SQL provide interaction between user and database via set of standard commands.

In simple words, SQL is language that help user to communicate with databases. SQL is not case sensitive language means you can type your query in small or capital letter as per user.

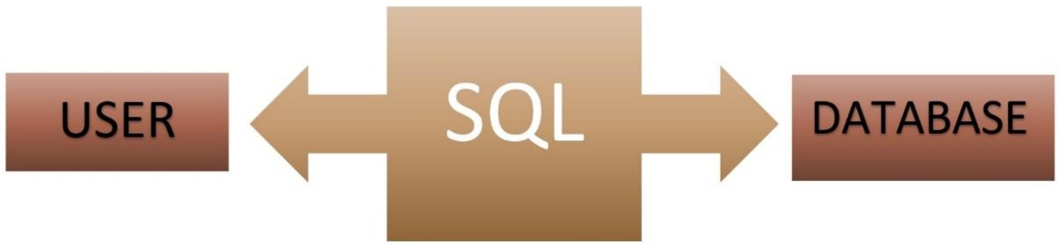


DIAGRAM: 7

HOW SQL PROVIDE INTERACTION BETWEEN USER AND DATABASE

You may write comments in SQL using "--" (Double hyphen)

User may get information from a database file from required query. Query is a request in the form of SQL command to retrieve information with some condition. You will see lots of query performing different type of operations. SQL is query language (Query based language) which works on structured data (data in structured form).

Now let's discuss all the SQL Commands in categorized way.

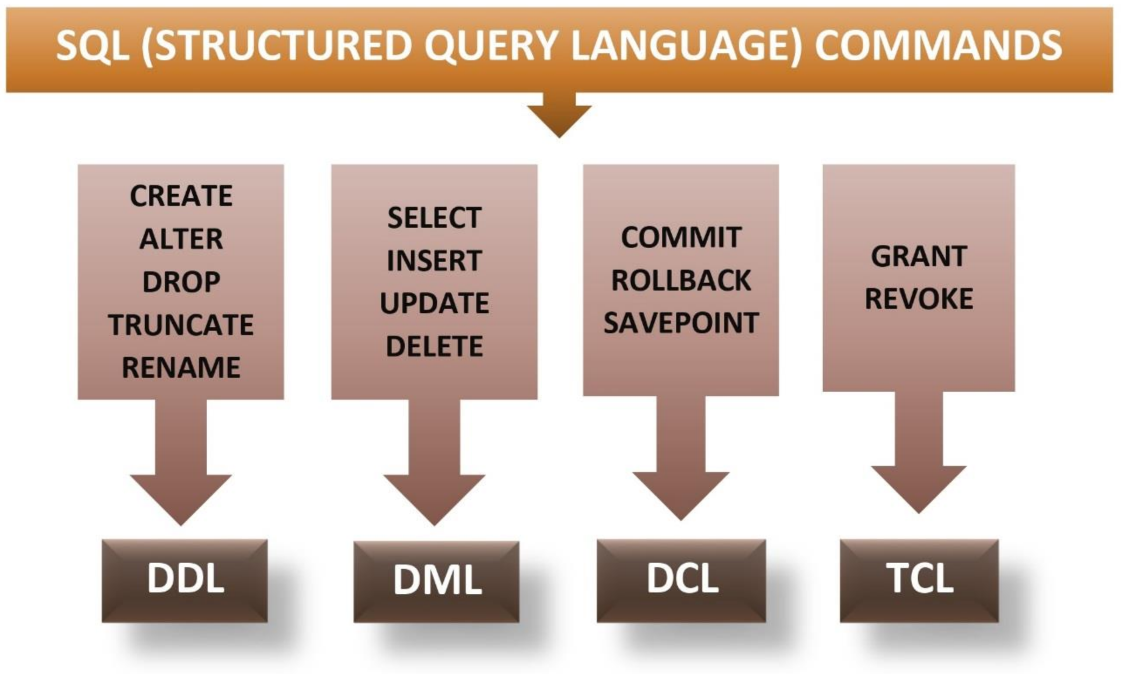


DIAGRAM: 8

SQL COMMANDS CAN BE CLASSIFIED INTO FOLLOWING CATEGORY

SQL perform following operation:

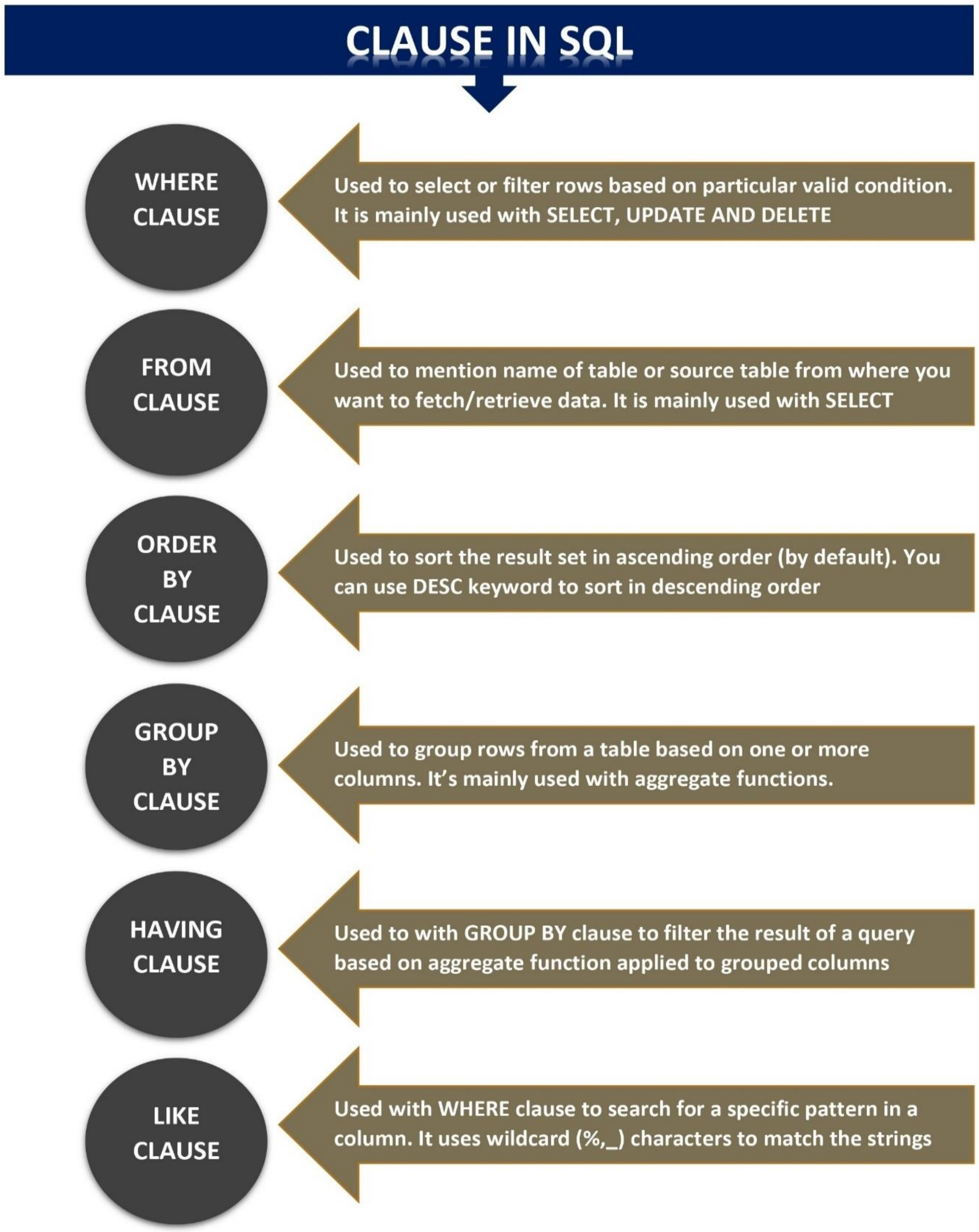
- Create a database
- Create a table
- Create view
- Insert data
- Update data
- Delete data
- Execute Query
- Set Permission or Constraints in table

SQL Commands:

SQL commands are predefined set of commands which are already defined in SQL. Commands are basically combination of keyword and statement you want to execute. Keywords are reserved words that has special meaning for SQL, you don't need to define they are already defined in SQL. All you need to use these keywords with you particular statements.

CLAUSE IN SQL:

Clause: Clause are built in functions which is use to deal with data inside the table that help SQL to filter and analyses data quickly.


CLAUSE IN SQL
WHERE CLAUSE
Used to select or filter rows based on particular valid condition. It is mainly used with SELECT, UPDATE AND DELETE
FROM CLAUSE
Used to mention name of table or source table from where you want to fetch/retrieve data. It is mainly used with SELECT
ORDER BY CLAUSE
Used to sort the result set in ascending order (by default). You can use DESC keyword to sort in descending order
GROUP BY CLAUSE
Used to group rows from a table based on one or more columns. It's mainly used with aggregate functions.
HAVING CLAUSE
Used to with GROUP BY clause to filter the result of a query based on aggregate function applied to grouped columns
LIKE CLAUSE
Used with WHERE clause to search for a specific pattern in a column. It uses wildcard (%,_) characters to match the strings

Any SQL statements is composed of two or more clause.

These clause are used with your SQL statements to filter commands you may learn in detail in further section. Some mainly used clause are discussed below.

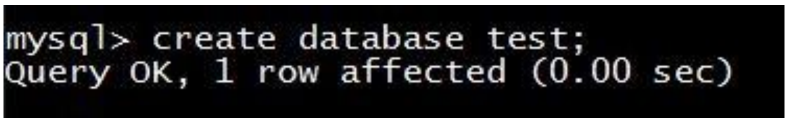
NOTE: All SQL statements are terminated with (;) semicolon
SQL statements are not case sensitive means SQL treats both upper and lower case commands as same.

We are going to use these above clauses in our all types of commands in SQL.

Few basic SQL Commands:

Create databases:

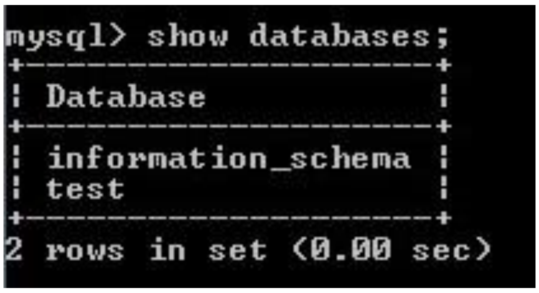
To create a new database, create databases command is used.
SQL Syntax: create database <database name>;



```
mysql> create database test;
Query OK, 1 row affected (0.00 sec)
```

Show databases:

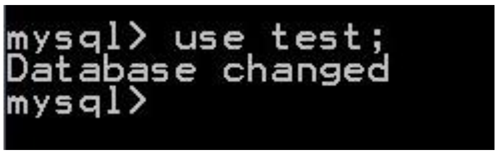
To view the list of available/created databases in SQL, *show databases* command is used.
SQL Syntax:
show databases;



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| test               |
+--------------------+
2 rows in set (0.00 sec)
```

Use database:

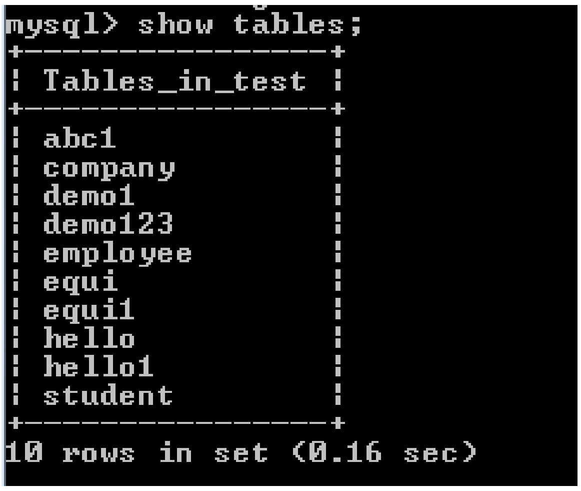
To select a database among available databases, *use database* command is used.
SQL Syntax:
use <database_name>;



```
mysql> use test;
Database changed
mysql>
```

Show tables:

To view the list of available/created databases in SQL, *show tables* command is used.
SQL Syntax:
show tables;



```
mysql> show tables;
+-----------------+
| Tables_in_test  |
+-----------------+
| abc1            |
| company         |
| demo1           |
| demo123         |
| employee        |
| equi            |
| equi1           |
| hello           |
| hello1          |
| student         |
+-----------------+
10 rows in set (0.16 sec)
```

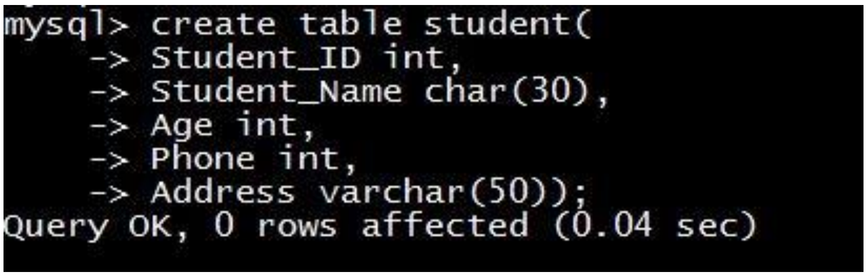
Create table:

To create a new table in the selected database. For example, if I want to create a table Student with following attributes and data types:

SQL Syntax:

create table <table name>(<attribute name> <data type> (size), <attribute name> <data type> (size) …);

Name of attribute	Data Type
Student_ID	int
Student_Name	char(30)
Age	int
Phone	int
Address	varchar(50)



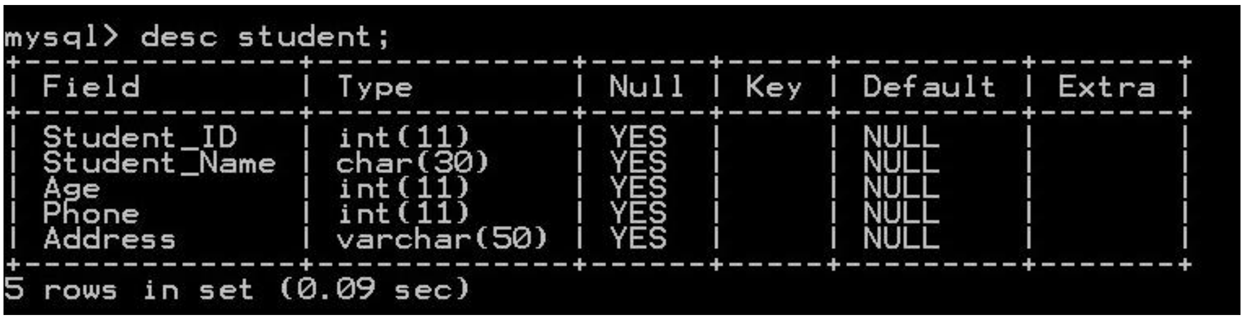
```
mysql> create table student(
    -> Student_ID int,
    -> Student_Name char(30),
    -> Age int,
    -> Phone int,
    -> Address varchar(50));
Query OK, 0 rows affected (0.04 sec)
```

Describe table:

To view the structure of table (like attributes and its data types, keys, constraints, default values), desc command is used.

SQL Syntax:

desc <table name>;   **OR**   describe <table name>;

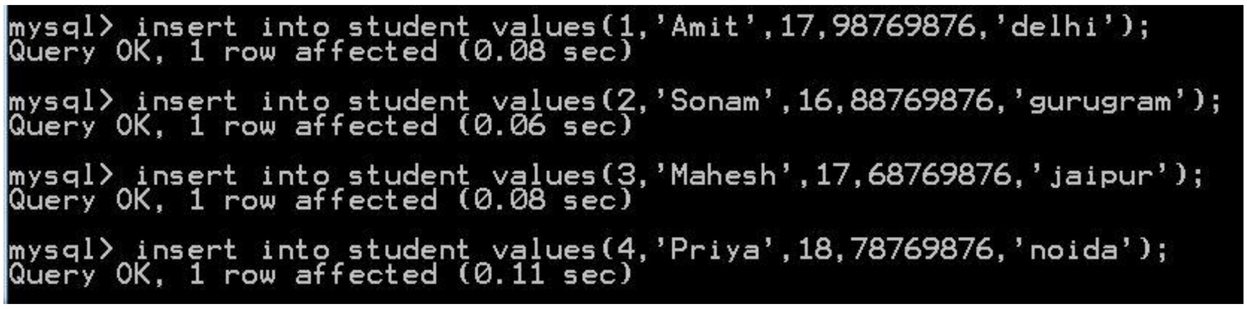

```
mysql> desc student;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Student_ID   | int(11)     | YES  |     | NULL    |       |
| Student_Name | char(30)    | YES  |     | NULL    |       |
| Age          | int(11)     | YES  |     | NULL    |       |
| Phone        | int(11)     | YES  |     | NULL    |       |
| Address      | varchar(50) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.09 sec)
```

Insert command:

To insert data in table, insert command is used (one row at a time). Here in this example, data of 4 students are inserted in table student.

SQL Syntax:

insert into <table name> values (<value>, <value> , <value> …);

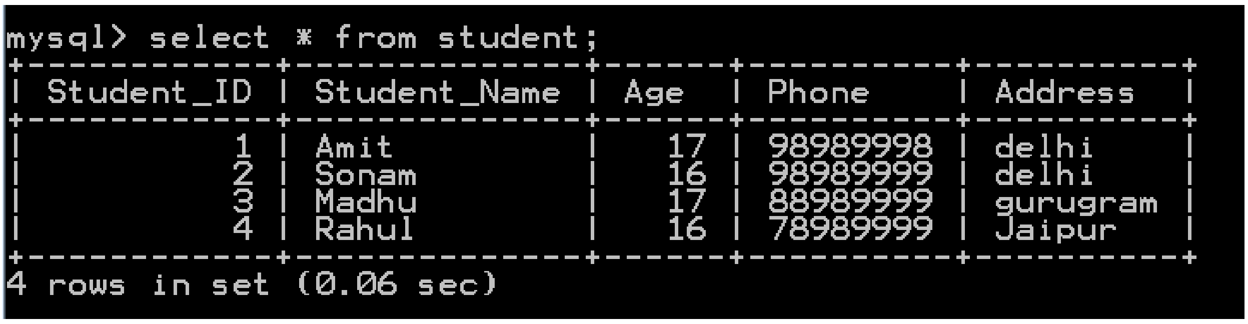

```
mysql> insert into student values(1,'Amit',17,98769876,'delhi');
Query OK, 1 row affected (0.08 sec)

mysql> insert into student values(2,'Sonam',16,88769876,'gurugram');
Query OK, 1 row affected (0.06 sec)

mysql> insert into student values(3,'Mahesh',17,68769876,'jaipur');
Query OK, 1 row affected (0.08 sec)

mysql> insert into student values(4,'Priya',18,78769876,'noida');
Query OK, 1 row affected (0.11 sec)
```

Select command:

To show the data of a table, select command is used. Let's show the data of 4 students in student table that was inserted in the previous command.

SQL Syntax:

select * from <table name>;

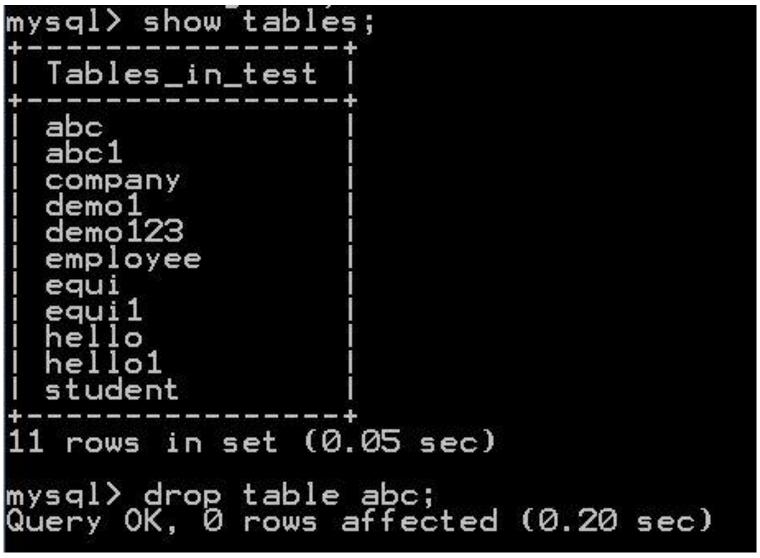

```
mysql> select * from student;
+------------+--------------+------+----------+----------+
| Student_ID | Student_Name | Age  | Phone    | Address  |
+------------+--------------+------+----------+----------+
|          1 | Amit         |   17 | 98989998 | delhi    |
|          2 | Sonam        |   16 | 98989999 | delhi    |
|          3 | Madhu        |   17 | 88989999 | gurugram |
|          4 | Rahul        |   16 | 78989999 | Jaipur   |
+------------+--------------+------+----------+----------+
4 rows in set (0.06 sec)
```

Drop table command:

To delete data as well as structure of a table, drop command is used.

SQL Syntax:

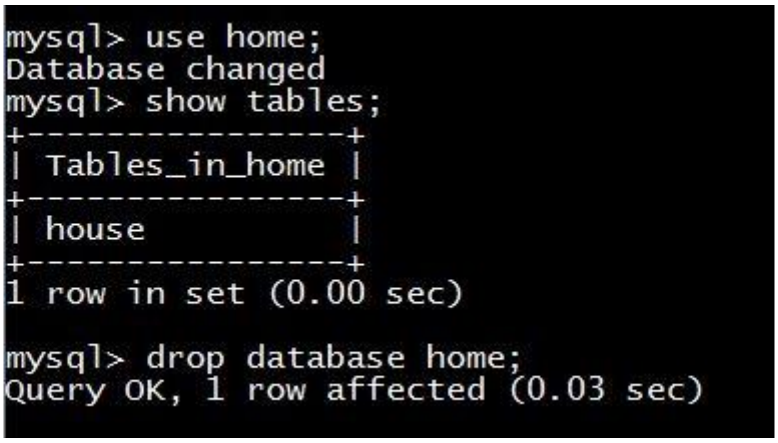
drop table <table name>;



```
mysql> show tables;
+-----------------+
| Tables_in_test  |
+-----------------+
| abc             |
| abc1            |
| company         |
| demo1           |
| demo123         |
| employee        |
| equi            |
| equi1           |
| hello           |
| hello1          |
| student         |
+-----------------+
11 rows in set (0.05 sec)

mysql> drop table abc;
Query OK, 0 rows affected (0.20 sec)
```

Drop database command:

To delete a database along with all the tables present in the database, drop command is used.
SQL Syntax:
drop database <database name>;



```
mysql> use home;
Database changed
mysql> show tables;
+----------------+
| Tables_in_home |
+----------------+
| house          |
+----------------+
1 row in set (0.00 sec)

mysql> drop database home;
Query OK, 1 row affected (0.03 sec)
```

Constraints:

Constraints in SQL are set of rules that are applied to the data in a relation/table. Constraints are used to ensure the accuracy and reliability of the data. Constraints can be at column level or at table level. Column level constraints apply to a column, and table level constraints apply to the whole table. After applying constraints to the table, if any violation happens then the data can't be inserted or action can't be completed.

Types of constraints:

1. **Primary key**
2. **Unique**
3. **Not Null**
4. **Foreign Key**

Let’s discuss in Details:

1. Primary Key Constraints:

NOT NULL+UNIQUE=Primary Key

This key applies unique and not null constraint to the column.

This constraints ensure that each column/attribute of table contain unique value for each row.

Let’s learn with the help of example:

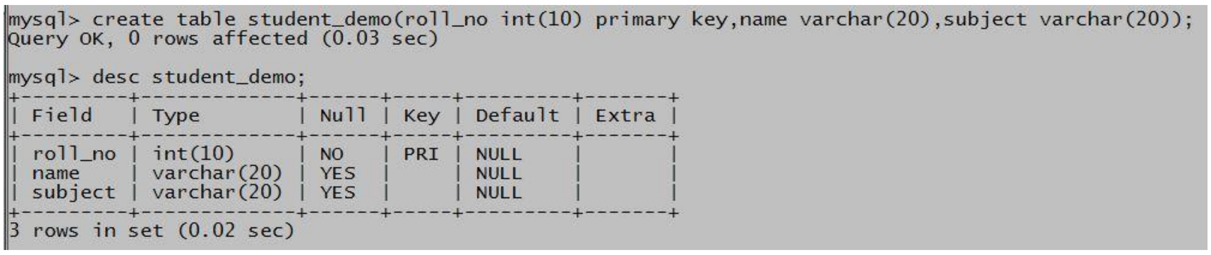
First we have created a table named student_demo and applying primary key constraints on roll_no attribute.

SQL Syntax:

create table <table_name>(<column_name> <datatype> <size> <constraint>);

Command:

create table student_demo(roll_no int(10) primary key,name varchar(20),subject varchar(20));



```
mysql> create table student_demo(roll_no int(10) primary key,name varchar(20),subject varchar(20));
Query OK, 0 rows affected (0.03 sec)

mysql> desc student_demo;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| roll_no | int(10)     | NO   | PRI | NULL    |       |
| name    | varchar(20) | YES  |     | NULL    |       |
| subject | varchar(20) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```

2. Unique Constraints :

NULL VALUES or (NOT NULL VALUES+UNIQUE) =UNIQUE

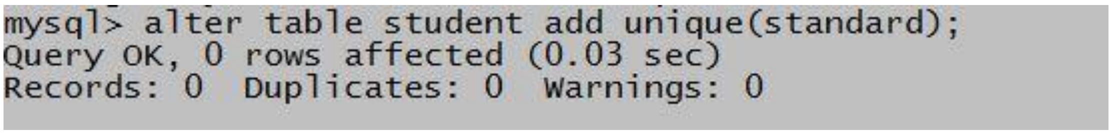
Unique constraint make sure that all values present or inserted in a column are different.

And this constraints follows all property of primary key constraints except that Primary key can’t contain NULL values but unique constraints allows NULL values. It ensures that values in the column are unique across all rows.

NOTE: You can have more than one unique key but only one primary key

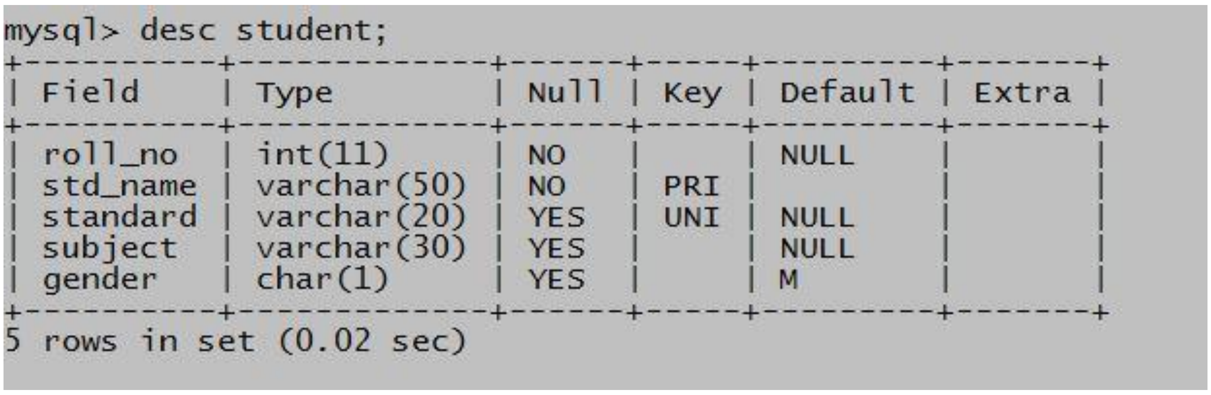
SQL Syntax: alter table <table_name> add <constraints_name>(<column_name>);

Command: alter table student add unique(standard);



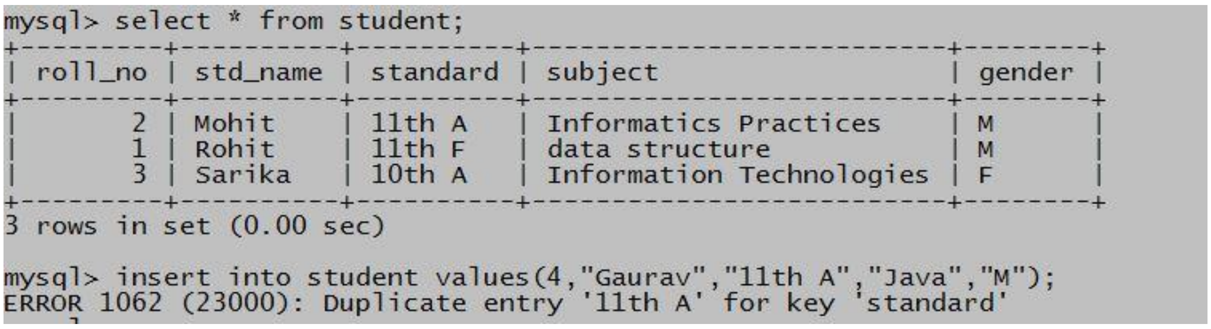
```
mysql> alter table student add unique(standard);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Let’s see table schema after applying above code:



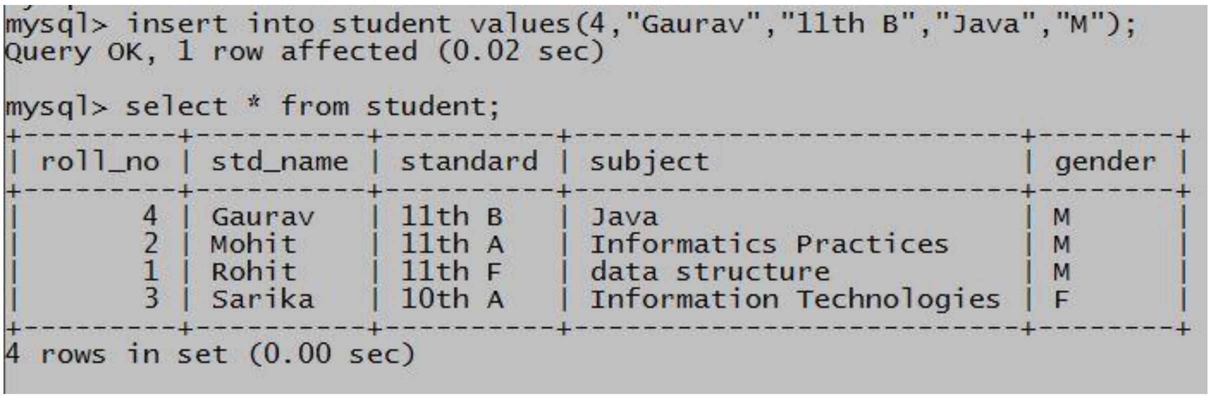
```
mysql> desc student;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| roll_no  | int(11)     | NO   |     | NULL    |       |
| std_name | varchar(50) | NO   | PRI |         |       |
| standard | varchar(20) | YES  | UNI | NULL    |       |
| subject  | varchar(30) | YES  |     | NULL    |       |
| gender   | char(1)     | YES  |     | M       |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.02 sec)
```

As you can see Unique Key constraints assigned to attribute/Field named standard. Which means standard column hold unique value.
Now as we know standard attribute can only hold unique value. If we attempt to insert duplicate values in standard attribute. This may occur an error. Let's see how



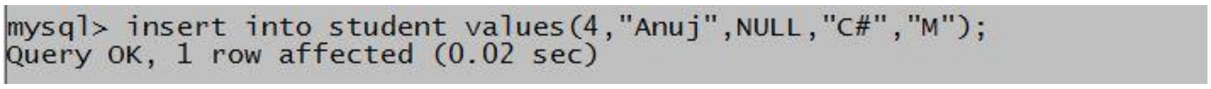
```
mysql> select * from student;
+---------+----------+----------+-------------------------+--------+
| roll_no | std_name | standard | subject                 | gender |
+---------+----------+----------+-------------------------+--------+
|       2 | Mohit    | 11th A   | Informatics Practices   | M      |
|       1 | Rohit    | 11th F   | data structure          | M      |
|       3 | Sarika   | 10th A   | Information Technologies | F      |
+---------+----------+----------+-------------------------+--------+
3 rows in set (0.00 sec)

mysql> insert into student values(4,"Gaurav","11th A","Java","M");
ERROR 1062 (23000): Duplicate entry '11th A' for key 'standard'
```

Now we are going to enter a valid entry in student table. Let's see

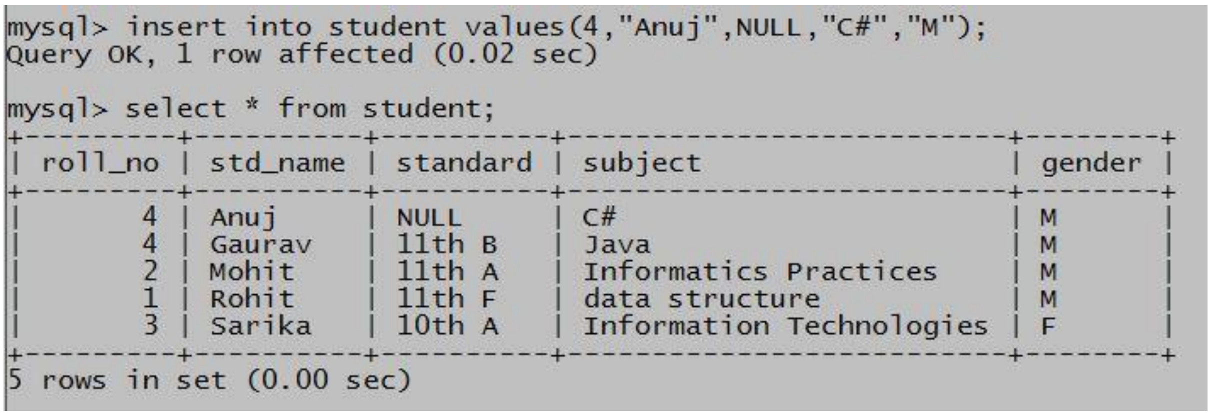


```
mysql> insert into student values(4,"Gaurav","11th B","Java","M");
Query OK, 1 row affected (0.02 sec)

mysql> select * from student;
+---------+----------+----------+-------------------------+--------+
| roll_no | std_name | standard | subject                 | gender |
+---------+----------+----------+-------------------------+--------+
|       4 | Gaurav   | 11th B   | Java                    | M      |
|       2 | Mohit    | 11th A   | Informatics Practices   | M      |
|       1 | Rohit    | 11th F   | data structure          | M      |
|       3 | Sarika   | 10th A   | Information Technologies | F      |
+---------+----------+----------+-------------------------+--------+
4 rows in set (0.00 sec)
```

As you can see all entry are clearly affected in table student. Now we need to check that standard attribute allows NULL values. Let's understand with SQL command.



```
mysql> insert into student values(4,"Anuj",NULL,"C#","M");
Query OK, 1 row affected (0.02 sec)
```

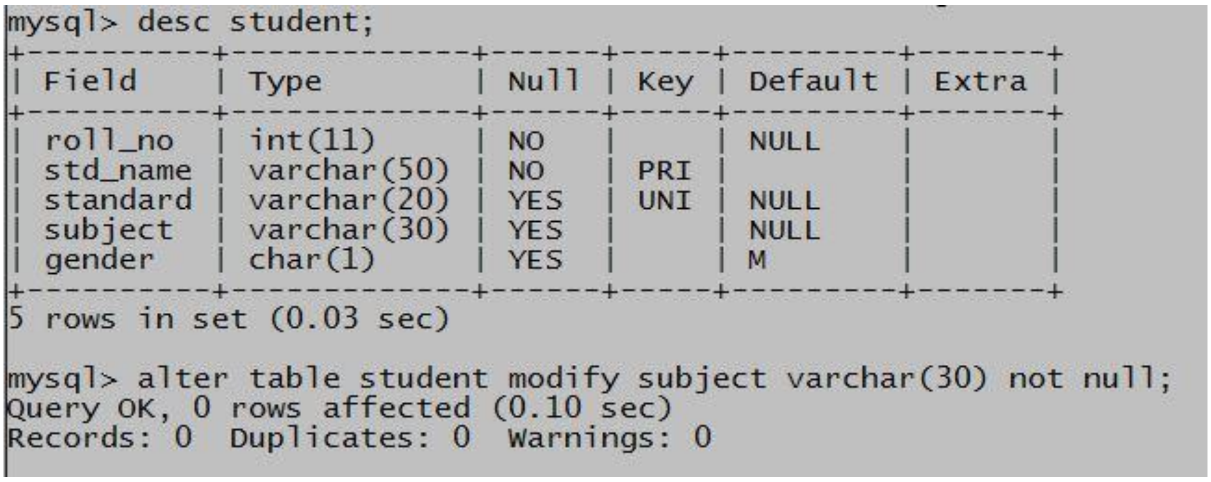


```
mysql> insert into student values(4,"Anuj",NULL,"C#","M");
Query OK, 1 row affected (0.02 sec)

mysql> select * from student;
+---------+----------+----------+-------------------------+--------+
| roll_no | std_name | standard | subject                 | gender |
+---------+----------+----------+-------------------------+--------+
|       4 | Anuj     | NULL     | C#                      | M      |
|       4 | Gaurav   | 11th B   | Java                    | M      |
|       2 | Mohit    | 11th A   | Informatics Practices   | M      |
|       1 | Rohit    | 11th F   | data structure          | M      |
|       3 | Sarika   | 10th A   | Information Technologies | F      |
+---------+----------+----------+-------------------------+--------+
5 rows in set (0.00 sec)
```

3. **NOT NULL Constraints :** Which never accept NULL values

NOT NULL constraints ensures that columns in table does not contain any NULL values. NULL values means missing or unknown values. If you enforce this NOT NULL constraints to any attribute than you are not able to insert any NULL values in it. Let’s see how.

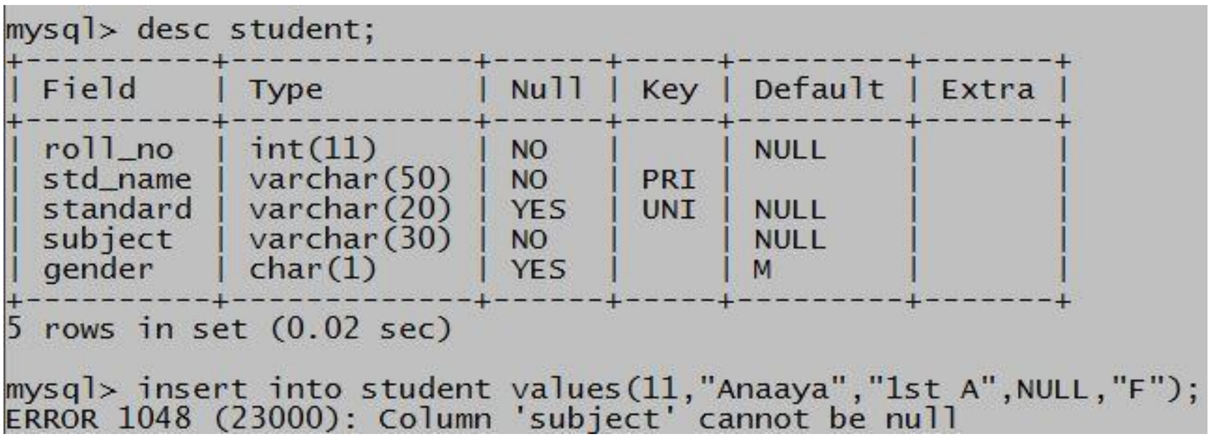
In student table, only two attribute assigned with NOT NULL constraint, first one is roll_no and std_name which means that these column can’t accept NULL values
Now we enforce NOT NULL constraint to another column named subject in existing table student.



```
mysql> desc student;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| roll_no  | int(11)     | NO   |     | NULL    |       |
| std_name | varchar(50) | NO   | PRI |         |       |
| standard | varchar(20) | YES  | UNI | NULL    |       |
| subject  | varchar(30) | YES  |     | NULL    |       |
| gender   | char(1)     | YES  |     | M       |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.03 sec)

mysql> alter table student modify subject varchar(30) not null;
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

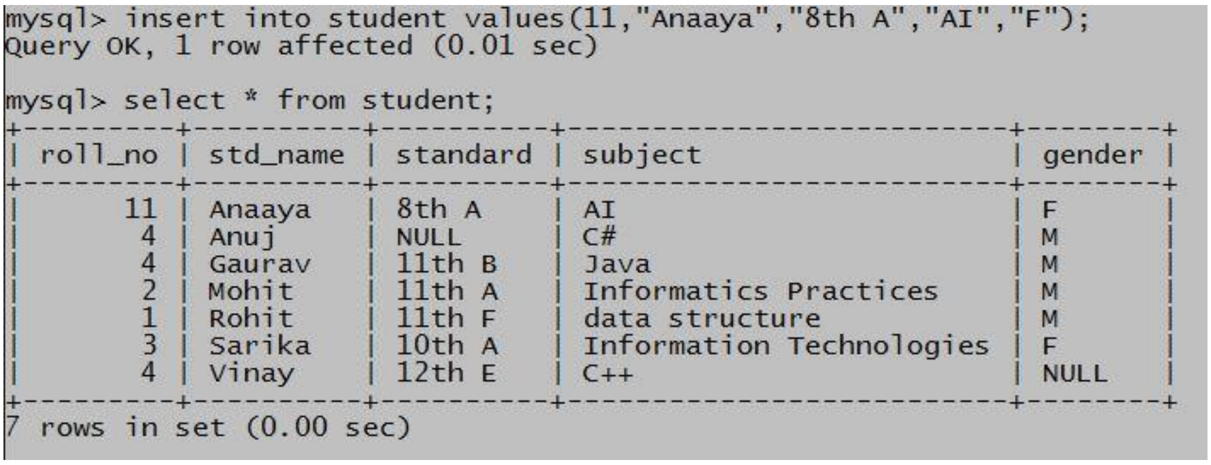
As you can command is successfully executed. Let’s see the schema of the table to check NOT NULL constraint successfully applied on subject column.
In below diagram it is clearly shows that subject NULL Type is set to NO means you can’t insert NULL value.

After that we have to check by insert a command with NULL values whether the subject column is accepting NULL value or not. Let’s see



```
mysql> desc student;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| roll_no  | int(11)     | NO   |     | NULL    |       |
| std_name | varchar(50) | NO   | PRI |         |       |
| standard | varchar(20) | YES  | UNI | NULL    |       |
| subject  | varchar(30) | NO   |     | NULL    |       |
| gender   | char(1)     | YES  |     | M       |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.02 sec)

mysql> insert into student values(11,"Anaaya","1st A",NULL,"F");
ERROR 1048 (23000): Column 'subject' cannot be null
```

As you can there is error occur that column ‘subject’ cannot be null
Now we try to insert a another command in the student table and check whether the insertion is successful or not



```
mysql> insert into student values(11,"Anaaya","8th A","AI","F");
Query OK, 1 row affected (0.01 sec)

mysql> select * from student;
+---------+----------+----------+-------------------------+--------+
| roll_no | std_name | standard | subject                 | gender |
+---------+----------+----------+-------------------------+--------+
|      11 | Anaaya   | 8th A    | AI                      | F      |
|       4 | Anuj     | NULL     | C#                      | M      |
|       4 | Gaurav   | 11th B   | Java                    | M      |
|       2 | Mohit    | 11th A   | Informatics Practices   | M      |
|       1 | Rohit    | 11th F   | data structure          | M      |
|       3 | Sarika   | 10th A   | Information Technologies | F      |
|       4 | Vinay    | 12th E   | C++                     | NULL   |
+---------+----------+----------+-------------------------+--------+
7 rows in set (0.00 sec)
```

4. Foreign Key:

In Foreign Key constraints, unique, not null and primary key constraint applies to a single table where as foreign key constraint applies to two tables.

For example: we have two tables’ student and awards as follows:

Table: Student

ID	Name	Age	City
1	Amit	15	Delhi
2	Madhu	14	Gurugram
3	Manoj	15	Noida
4	Asif	15	Faridabad

Table: Awards

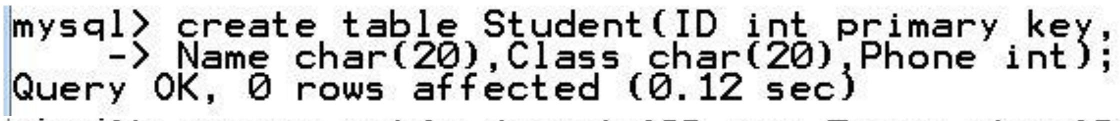
ID	Award	Sport
1	Gold	Badminton
2	Silver	Tennis
1	Silver	Hockey
4	Bronze	Badminton

Here we will establish foreign key constraint on column **ID** of **student table** and column **id** of **awards table.** Here we will consider **Student table** as **parent table** and **Awards table** as **child table.** Following rules must be followed:

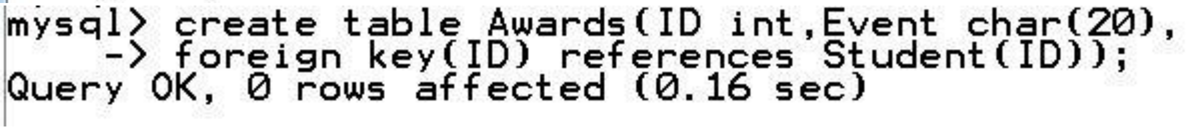
(a) Column ID of Student table must be its primary key.
(b) Column ID of Awards table may or may not be primary key of Awards table.

- Foreign key constraint ensures that only that data can be inserted in column ID of Awards table which is present in column ID of Student table.

Example: We will create two tables. **Student table** as **parent table** and **Awards table** as **child table**. Now we will establish foreign key constraint on column **ID** of **student table** and column **id** of **awards table.**

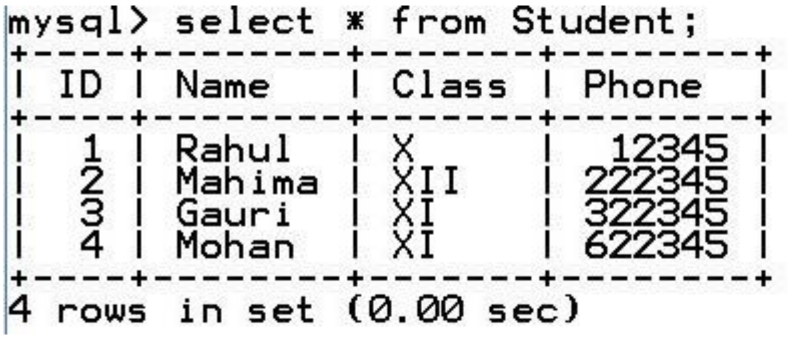


```
mysql> create table Student(ID int primary key,
    -> Name char(20),Class char(20),Phone int);
Query OK, 0 rows affected (0.12 sec)
```



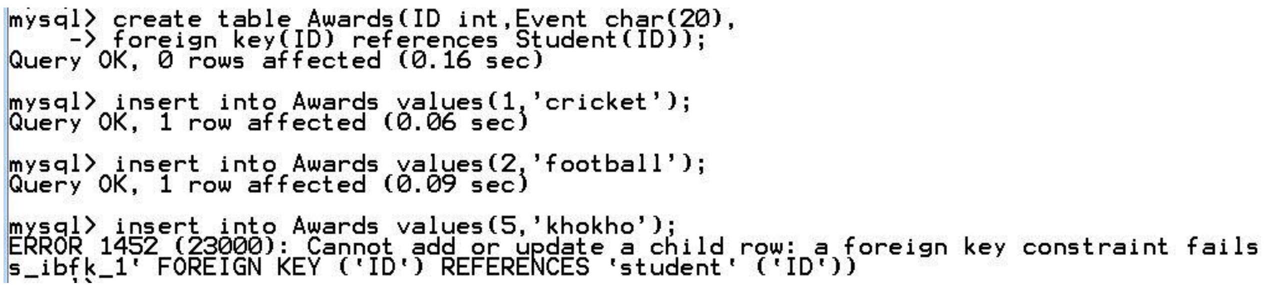
```
mysql> create table Awards(ID int,Event char(20),
    -> foreign key(ID) references Student(ID));
Query OK, 0 rows affected (0.16 sec)
```

Now let's check how foreign key constraint works. We have already added data in Student table.



```
mysql> select * from Student;
+----+--------+-------+--------+
| ID | Name   | Class | Phone  |
+----+--------+-------+--------+
|  1 | Rahul  | X     |  12345 |
|  2 | Mahima | XII   | 222345 |
|  3 | Gauri  | XI    | 322345 |
|  4 | Mohan  | XI    | 622345 |
+----+--------+-------+--------+
4 rows in set (0.00 sec)
```

Now let's check how foreign key constraint enforces referential integrity.



```
mysql> create table Awards(ID int,Event char(20),
    -> foreign key(ID) references Student(ID));
Query OK, 0 rows affected (0.16 sec)

mysql> insert into Awards values(1,'cricket');
Query OK, 1 row affected (0.06 sec)

mysql> insert into Awards values(2,'football');
Query OK, 1 row affected (0.09 sec)

mysql> insert into Awards values(5,'khokho');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails
s_ibfk_1' FOREIGN KEY ('ID') REFERENCES 'student' ('ID'))
```

Here we can see that ID 5 is not present in parent table (Student). So, ID 5 can't be added in child table (Awards).

DDL (Data Definition Language) Commands:

These commands are used to make any changes in the structure of the table/database. These command don't change the data of the table.

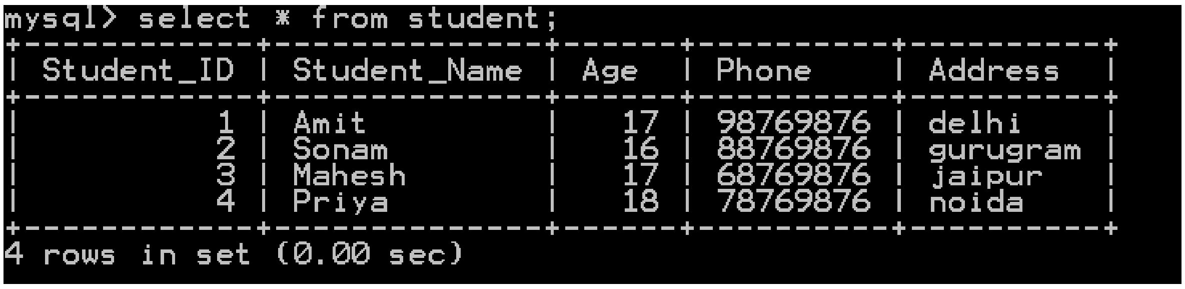
Example: create table, alter table, drop table, create database, create view etc.

We have already covered few DDL Commands like create database, create table, drop database, drop table. Few more DDL commands like alter table will be discussed now.

Alter Table:
This is a DDL command and it is used to modify a table. This command can be used to add, delete, or modify columns, add or drop constraints etc.
SQL Syntax:
alter table <table name> [alter option];

Add a column to the table:

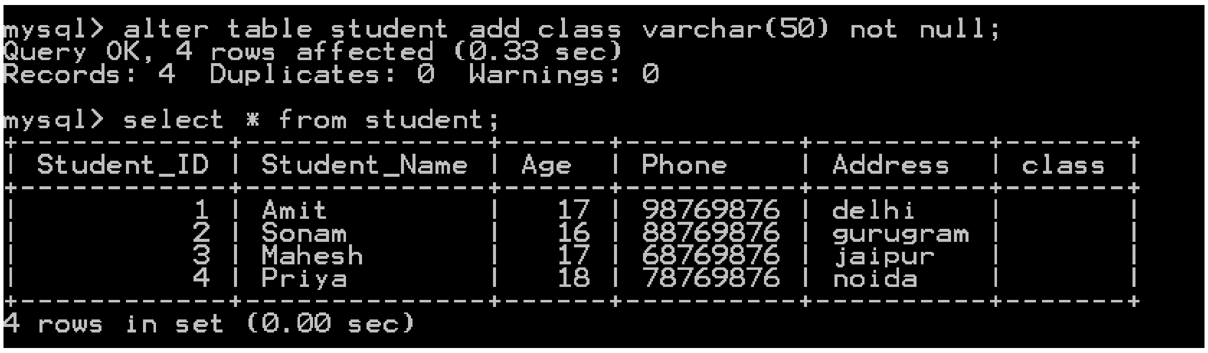
We have table student which was created in previous section.



```
mysql> select * from student;
+------------+--------------+------+----------+----------+
| Student_ID | Student_Name | Age  | Phone    | Address  |
+------------+--------------+------+----------+----------+
|          1 | Amit         |   17 | 98769876 | delhi    |
|          2 | Sonam        |   16 | 88769876 | gurugram |
|          3 | Mahesh       |   17 | 68769876 | jaipur   |
|          4 | Priya        |   18 | 78769876 | noida    |
+------------+--------------+------+----------+----------+
4 rows in set (0.00 sec)
```

SQL Syntax:
alter table <table name> add <column name><data type> [constraint];

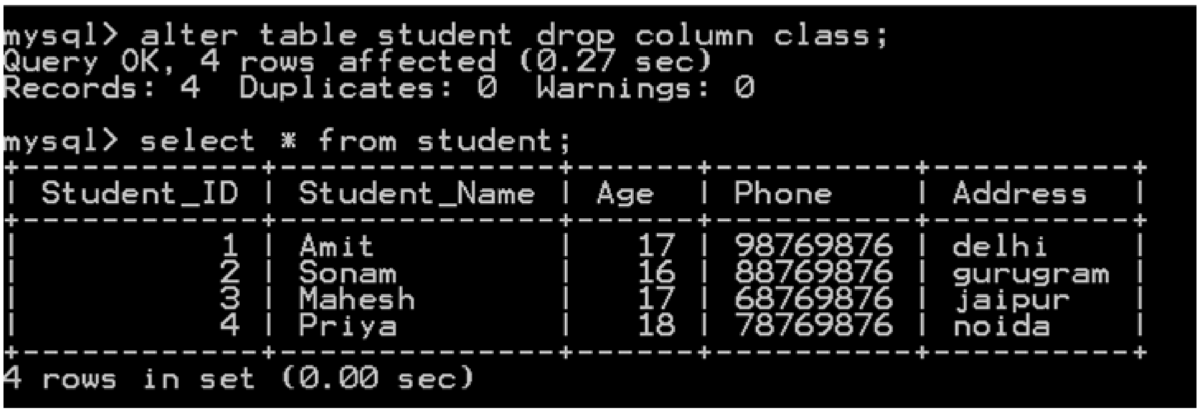
Example: If we want to add a column class with data type varchar and size 50 and nulls are not allowed.



```
mysql> alter table student add class varchar(50) not null;
Query OK, 4 rows affected (0.33 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from student;
+------------+--------------+------+----------+----------+-------+
| Student_ID | Student_Name | Age  | Phone    | Address  | class |
+------------+--------------+------+----------+----------+-------+
|          1 | Amit         |   17 | 98769876 | delhi    |       |
|          2 | Sonam        |   16 | 88769876 | gurugram |       |
|          3 | Mahesh       |   17 | 68769876 | jaipur   |       |
|          4 | Priya        |   18 | 78769876 | noida    |       |
+------------+--------------+------+----------+----------+-------+
4 rows in set (0.00 sec)
```

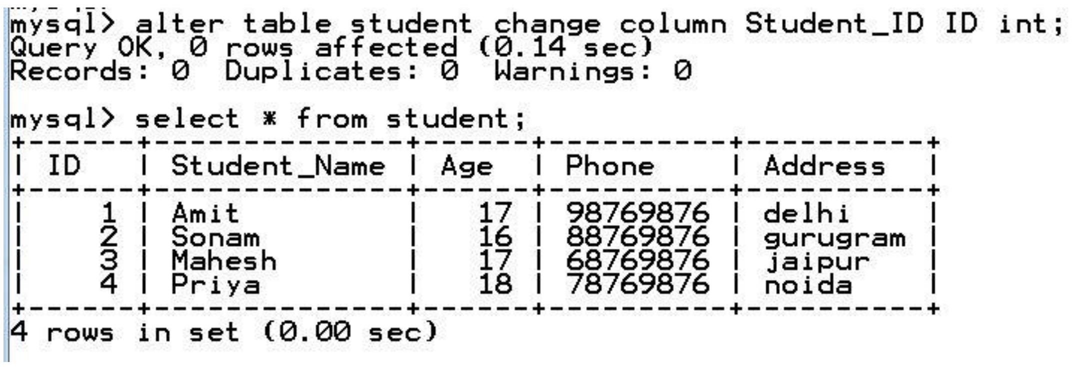
Drop a column from the table:

Let’s delete a column class from table student which we added in the previous section.
SQL Syntax:
alter table <table name> drop column<column name>;

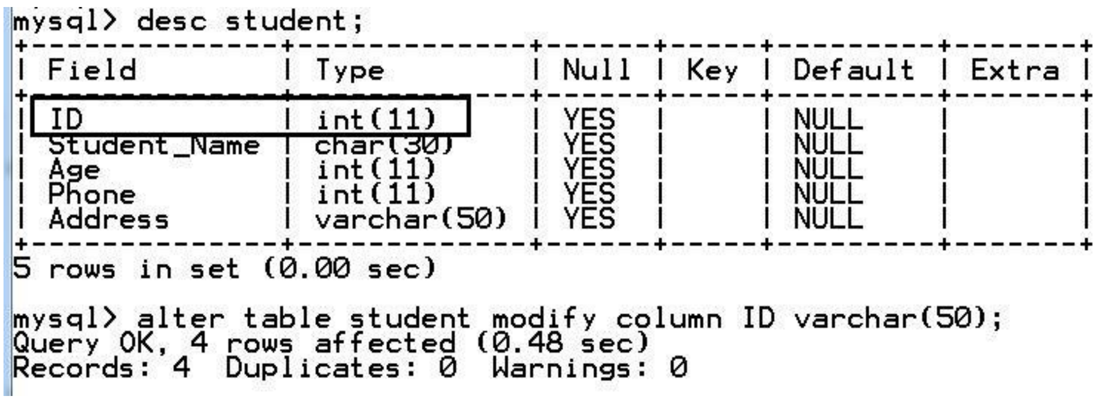

```
mysql> alter table student drop column class;
Query OK, 4 rows affected (0.27 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from student;
+------------+--------------+------+----------+----------+
| Student_ID | Student_Name | Age  | Phone    | Address  |
+------------+--------------+------+----------+----------+
|          1 | Amit         |   17 | 98769876 | delhi    |
|          2 | Sonam        |   16 | 88769876 | gurugram |
|          3 | Mahesh       |   17 | 68769876 | jaipur   |
|          4 | Priya        |   18 | 78769876 | noida    |
+------------+--------------+------+----------+----------+
4 rows in set (0.00 sec)
```

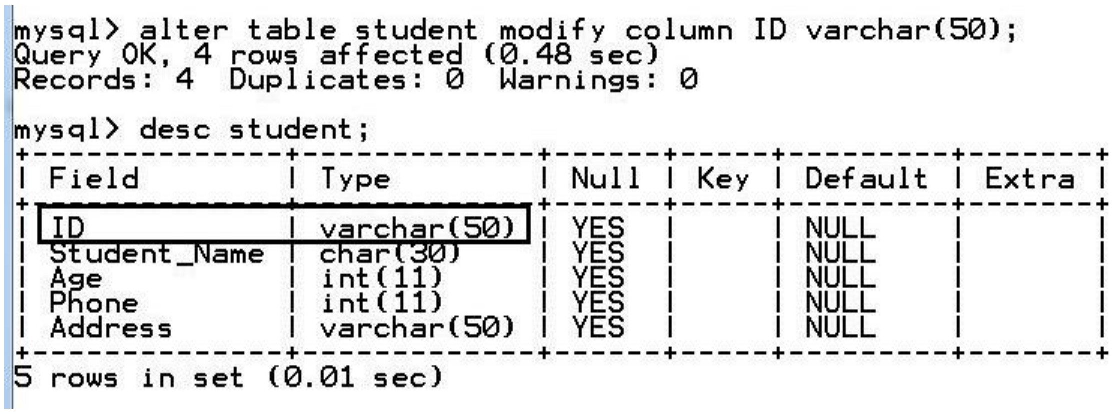
Modifying column of a table: We have different ways to modify a table like column name, data type, default value, size, order of column, constraints.

1. **Changing column name:** We can change the column name of a table using alter command. For example, in table student, we are going to change column name Student_ID to ID.
   SQL Syntax:
   alter table <table name> change column <old column name> <new column name> <data type>;


```
mysql> alter table student change column Student_ID ID int;
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from student;
+------+--------------+------+----------+----------+
| ID   | Student_Name | Age  | Phone    | Address  |
+------+--------------+------+----------+----------+
|    1 | Amit         |   17 | 98769876 | delhi    |
|    2 | Sonam        |   16 | 88769876 | gurugram |
|    3 | Mahesh       |   17 | 68769876 | jaipur   |
|    4 | Priya        |   18 | 78769876 | noida    |
+------+--------------+------+----------+----------+
4 rows in set (0.00 sec)
```

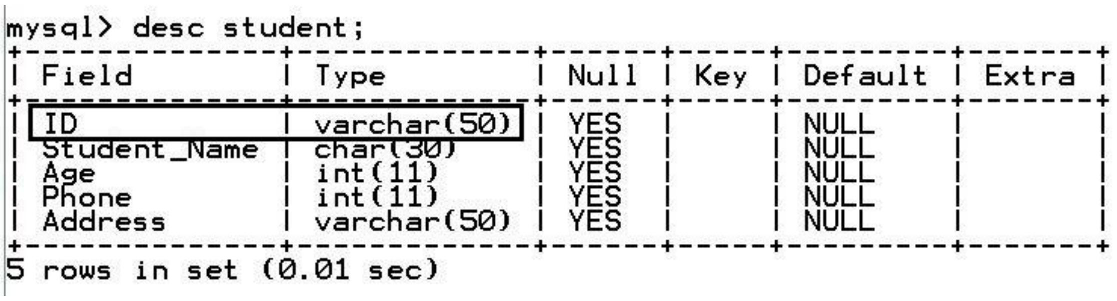
2. **Changing column data type:** We can change the column data type from varchar to char or int to varchar etc. of a table using alter command. For example, in table student, we are going to change datatype of column ID from int to varchar.
   SQL Syntax:
   alter table <table name> change column <column name><new data type>;


```
mysql> desc student;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| ID           | int(11)     | YES  |     | NULL    |       |
| Student_Name | char(30)    | YES  |     | NULL    |       |
| Age          | int(11)     | YES  |     | NULL    |       |
| Phone        | int(11)     | YES  |     | NULL    |       |
| Address      | varchar(50) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> alter table student modify column ID varchar(50);
Query OK, 4 rows affected (0.48 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

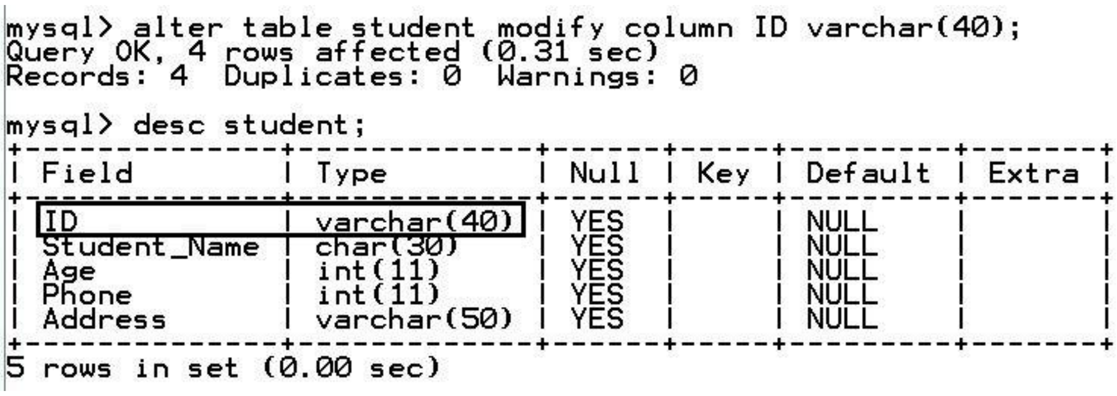


```
mysql> alter table student modify column ID varchar(50);
Query OK, 4 rows affected (0.48 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> desc student;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| ID           | varchar(50) | YES  |     | NULL    |       |
| Student_Name | char(30)    | YES  |     | NULL    |       |
| Age          | int(11)     | YES  |     | NULL    |       |
| Phone        | int(11)     | YES  |     | NULL    |       |
| Address      | varchar(50) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

3. **<u>Changing maximum size of the data in a column:</u>** We can change the maximum size of the data in a column of a table using alter command. For example, in table student, we are going to change size of column ID from varchar(50) to varchar(40).
   SQL Syntax:
   alter table <table name> change column <column name> <data type with size>;

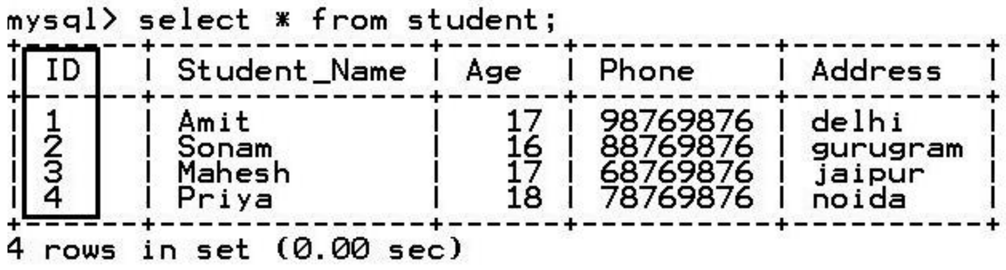


```
mysql> desc student;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| ID           | varchar(50) | YES  |     | NULL    |       |
| Student_Name | char(30)    | YES  |     | NULL    |       |
| Age          | int(11)     | YES  |     | NULL    |       |
| Phone        | int(11)     | YES  |     | NULL    |       |
| Address      | varchar(50) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

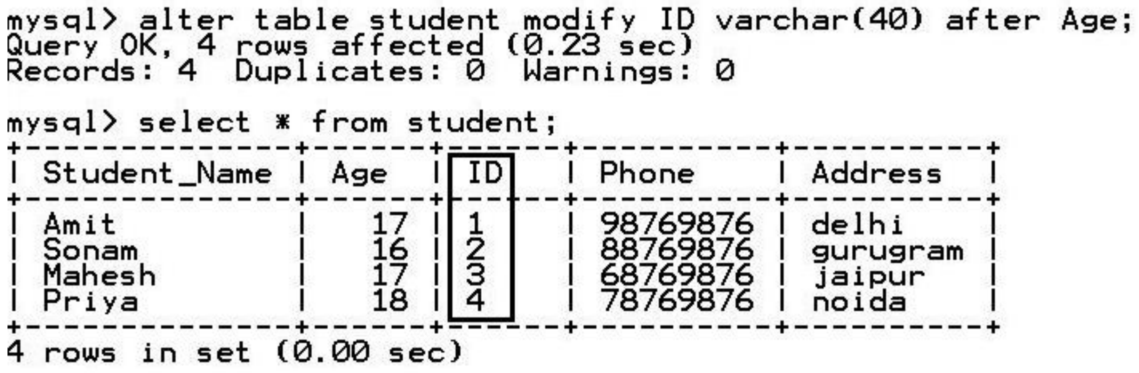


```
mysql> alter table student modify column ID varchar(40);
Query OK, 4 rows affected (0.31 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> desc student;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| ID           | varchar(40) | YES  |     | NULL    |       |
| Student_Name | char(30)    | YES  |     | NULL    |       |
| Age          | int(11)     | YES  |     | NULL    |       |
| Phone        | int(11)     | YES  |     | NULL    |       |
| Address      | varchar(50) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

4. **<u>Changing order of the column:</u>** We can change the order of the column of a table using alter command. For example, in table student, we are going to place column ID after column Age.
   SQL Syntax:
   alter table <table name> modify <column name> <data type with size>[first|after <column name>];

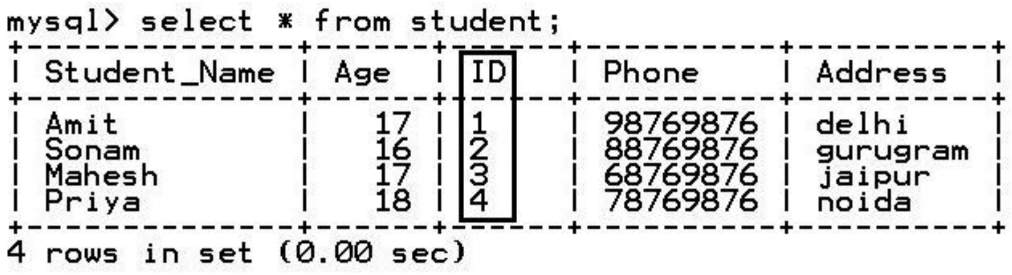


```
mysql> select * from student;
+----+--------------+-----+----------+----------+
| ID | Student_Name | Age | Phone    | Address  |
+----+--------------+-----+----------+----------+
| 1  | Amit         |  17 | 98769876 | delhi    |
| 2  | Sonam        |  16 | 88769876 | gurugram |
| 3  | Mahesh       |  17 | 68769876 | jaipur   |
| 4  | Priya        |  18 | 78769876 | noida    |
+----+--------------+-----+----------+----------+
4 rows in set (0.00 sec)
```

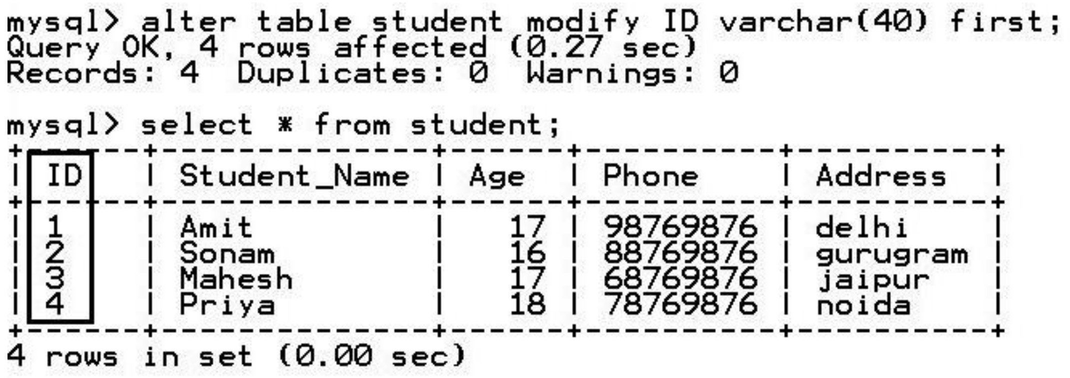


```
mysql> alter table student modify ID varchar(40) after Age;
Query OK, 4 rows affected (0.23 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from student;
+--------------+-----+----+----------+----------+
| Student_Name | Age | ID | Phone    | Address  |
+--------------+-----+----+----------+----------+
| Amit         |  17 | 1  | 98769876 | delhi    |
| Sonam        |  16 | 2  | 88769876 | gurugram |
| Mahesh       |  17 | 3  | 68769876 | jaipur   |
| Priya        |  18 | 4  | 78769876 | noida    |
+--------------+-----+----+----------+----------+
4 rows in set (0.00 sec)
```

Now we are going to put column ID back to first position.

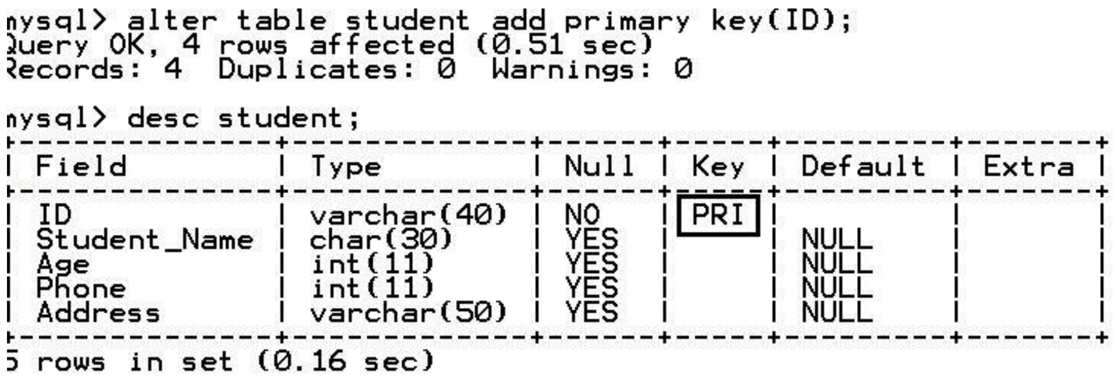


```
mysql> select * from student;
+--------------+-----+----+----------+----------+
| Student_Name | Age | ID | Phone    | Address  |
+--------------+-----+----+----------+----------+
| Amit         |  17 | 1  | 98769876 | delhi    |
| Sonam        |  16 | 2  | 88769876 | gurugram |
| Mahesh       |  17 | 3  | 68769876 | jaipur   |
| Priya        |  18 | 4  | 78769876 | noida    |
+--------------+-----+----+----------+----------+
4 rows in set (0.00 sec)
```



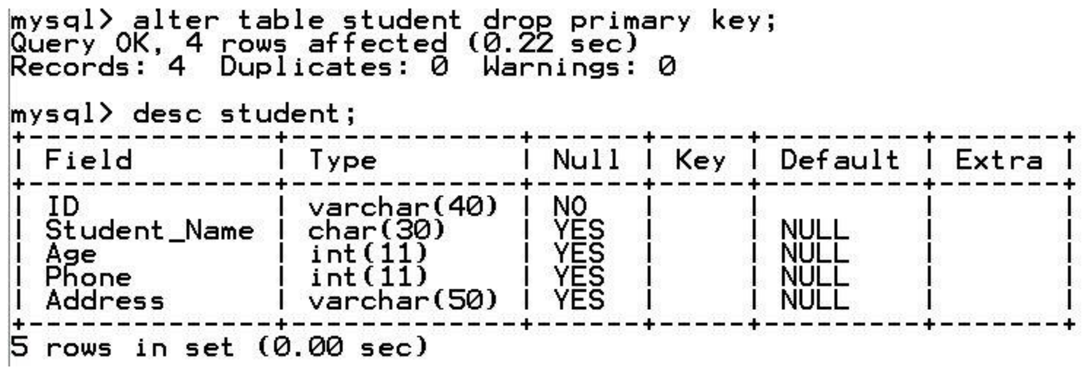
```
mysql> alter table student modify ID varchar(40) first;
Query OK, 4 rows affected (0.27 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from student;
+----+--------------+-----+----------+----------+
| ID | Student_Name | Age | Phone    | Address  |
+----+--------------+-----+----------+----------+
| 1  | Amit         |  17 | 98769876 | delhi    |
| 2  | Sonam        |  16 | 88769876 | gurugram |
| 3  | Mahesh       |  17 | 68769876 | jaipur   |
| 4  | Priya        |  18 | 78769876 | noida    |
+----+--------------+-----+----------+----------+
4 rows in set (0.00 sec)
```

5. **<u>Add/drop constraints/column:</u>** We can add/drop constraints in a table using alter command.
    - **<u>Adding primary key</u>**: We are going to add primary key at column ID using alter command.
      <u>Command</u>: alter table <table name> add primary key(<column name>);



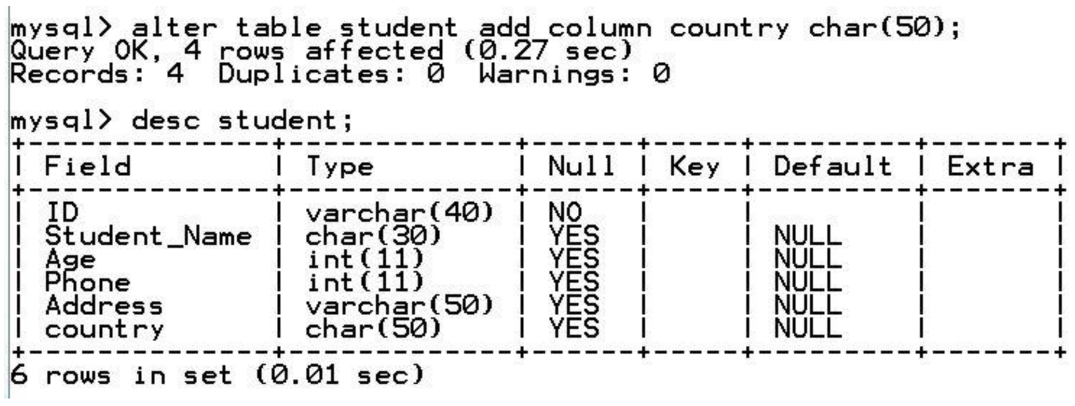
```
nysql> alter table student add primary key(ID);
Query OK, 4 rows affected (0.51 sec)
Records: 4  Duplicates: 0  Warnings: 0

nysql> desc student;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| ID           | varchar(40) | NO   | PRI |         |       |
| Student_Name | char(30)    | YES  |     | NULL    |       |
| Age          | int(11)     | YES  |     | NULL    |       |
| Phone        | int(11)     | YES  |     | NULL    |       |
| Address      | varchar(50) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.16 sec)
```

➢ **<u>Dropping primary key:</u>** We are going to remove primary key at column ID which we added in the previous section.
<u>Command:</u> alter table <table name> drop primary key;



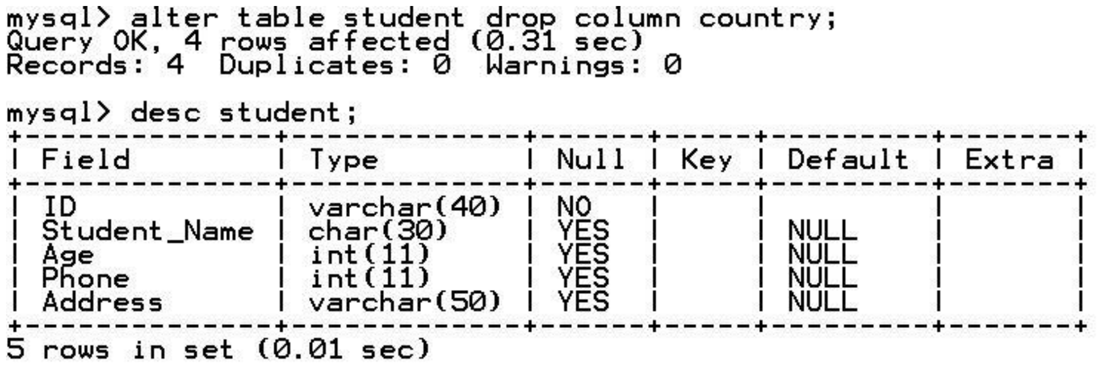
```
mysql> alter table student drop primary key;
Query OK, 4 rows affected (0.22 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> desc student;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| ID           | varchar(40) | NO   |     |         |       |
| Student_Name | char(30)    | YES  |     | NULL    |       |
| Age          | int(11)     | YES  |     | NULL    |       |
| Phone        | int(11)     | YES  |     | NULL    |       |
| Address      | varchar(50) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

➢ **<u>Adding a new column:</u>** We are going to add a column country with data type char of size 50 to the table student using alter command.
<u>Command:</u> alter table <table name> add column <column name> <data type with size>;



```
mysql> alter table student add column country char(50);
Query OK, 4 rows affected (0.27 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> desc student;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| ID           | varchar(40) | NO   |     |         |       |
| Student_Name | char(30)    | YES  |     | NULL    |       |
| Age          | int(11)     | YES  |     | NULL    |       |
| Phone        | int(11)     | YES  |     | NULL    |       |
| Address      | varchar(50) | YES  |     | NULL    |       |
| country      | char(50)    | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
6 rows in set (0.01 sec)
```

➢ **<u>Dropping a column:</u>** We are going to remove a column 'country' which we added in the last section using alter command.
<u>Command:</u> alter table <table name> drop column <column name>;



```
mysql> alter table student drop column country;
Query OK, 4 rows affected (0.31 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> desc student;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| ID           | varchar(40) | NO   |     |         |       |
| Student_Name | char(30)    | YES  |     | NULL    |       |
| Age          | int(11)     | YES  |     | NULL    |       |
| Phone        | int(11)     | YES  |     | NULL    |       |
| Address      | varchar(50) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

DML (Data Manipulation Language) Commands:

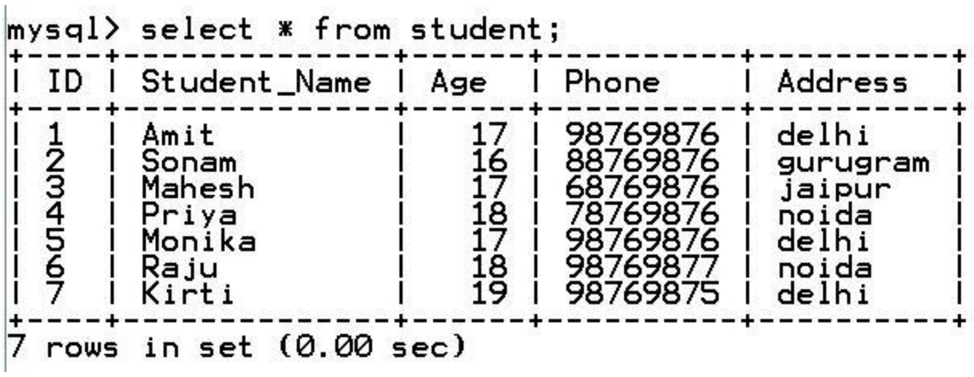
These commands are used to make any changes in the data of the table.

DML commands: Insert, delete, update, select etc.

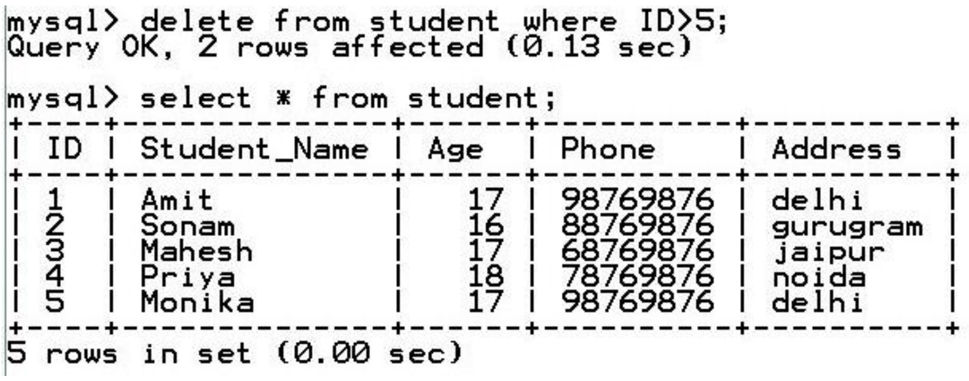
We have already covered few DML Commands like insert and select. Now we will discuss delete and update command.

Delete command:

Delete command is used to delete data from the table. **Where clause is used to give condition in a SQL query.** All those tuples which satisfies the condition will be deleted from the table.
SQL Syntax:
delete from <table name> where <condition>;
Now let's delete data of all those students from student table whose ID is greater than 5.



```
mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   17 | 98769876 | delhi    |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 4  | Priya        |   18 | 78769876 | noida    |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 6  | Raju         |   18 | 98769877 | noida    |
| 7  | Kirti        |   19 | 98769875 | delhi    |
+----+--------------+------+----------+----------+
7 rows in set (0.00 sec)
```



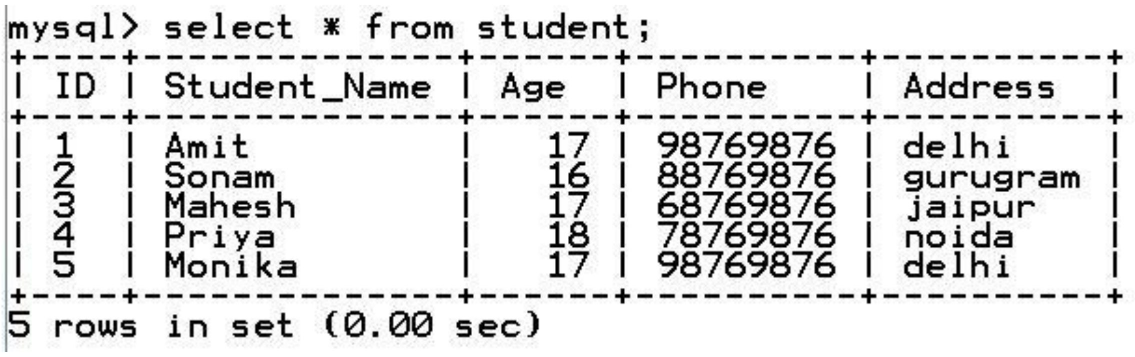
```
mysql> delete from student where ID>5;
Query OK, 2 rows affected (0.13 sec)

mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   17 | 98769876 | delhi    |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 4  | Priya        |   18 | 78769876 | noida    |
| 5  | Monika       |   17 | 98769876 | delhi    |
+----+--------------+------+----------+----------+
5 rows in set (0.00 sec)
```

<u>Update command</u>: Update command is used to update data from the table. **Where clause is used to give condition in a SQL query.** All those tuples which satisfies the condition will be update from the table.

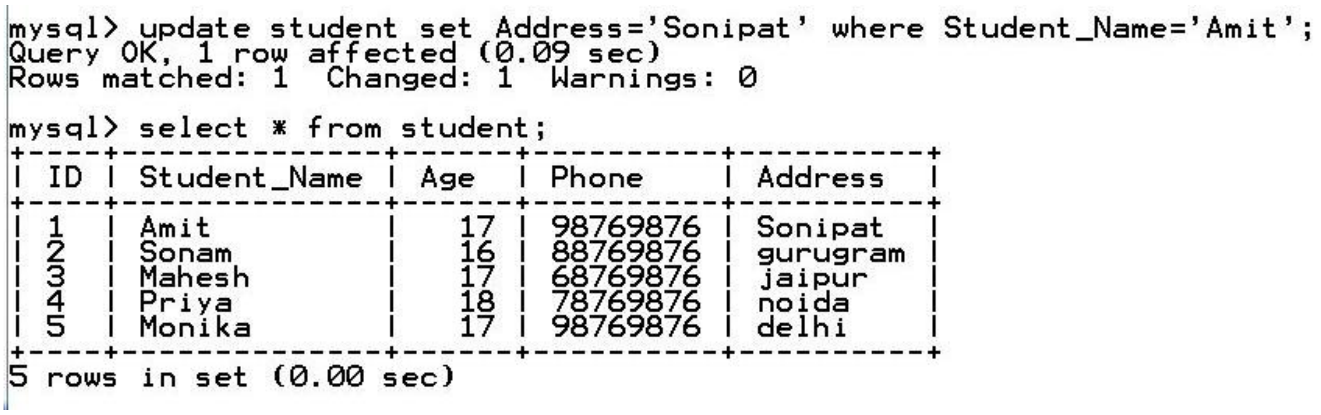
SQL Syntax:

update <table name> set <column name>=<new data> where <condition>;

Now let's update the Address from 'Delhi' to 'Sonipat' of that student whose name is 'Amit'.



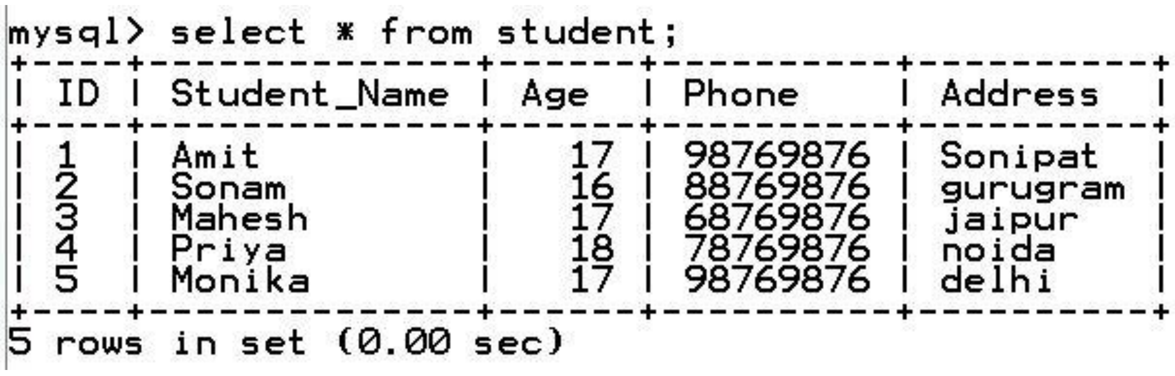
```
mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   17 | 98769876 | delhi    |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 4  | Priya        |   18 | 78769876 | noida    |
| 5  | Monika       |   17 | 98769876 | delhi    |
+----+--------------+------+----------+----------+
5 rows in set (0.00 sec)
```



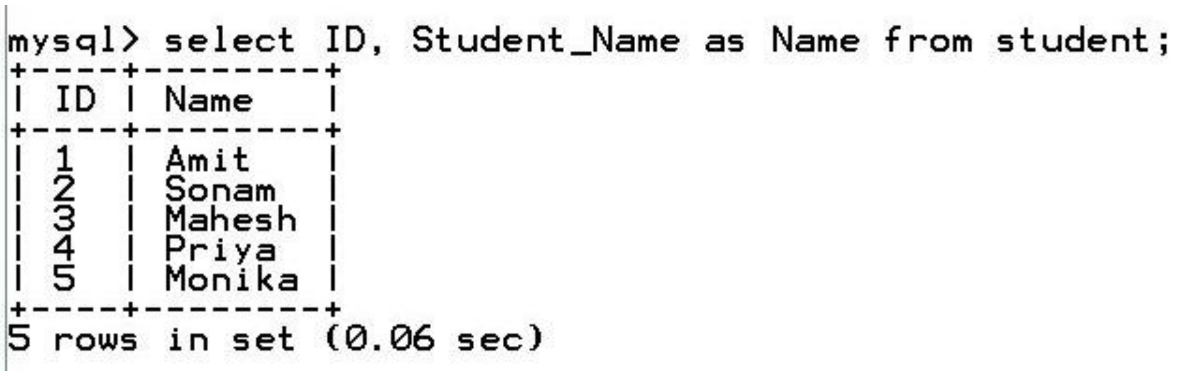
```
mysql> update student set Address='Sonipat' where Student_Name='Amit';
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   17 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 4  | Priya        |   18 | 78769876 | noida    |
| 5  | Monika       |   17 | 98769876 | delhi    |
+----+--------------+------+----------+----------+
5 rows in set (0.00 sec)
```

Aliasing:

Aliasing in SQL is the process of assigning a nick name or a temporary name to a table or column. We create aliases to make queries more readable and easier to use. Alias created using **as** keyword. Creating aliases don't change name of any table or column permanently.



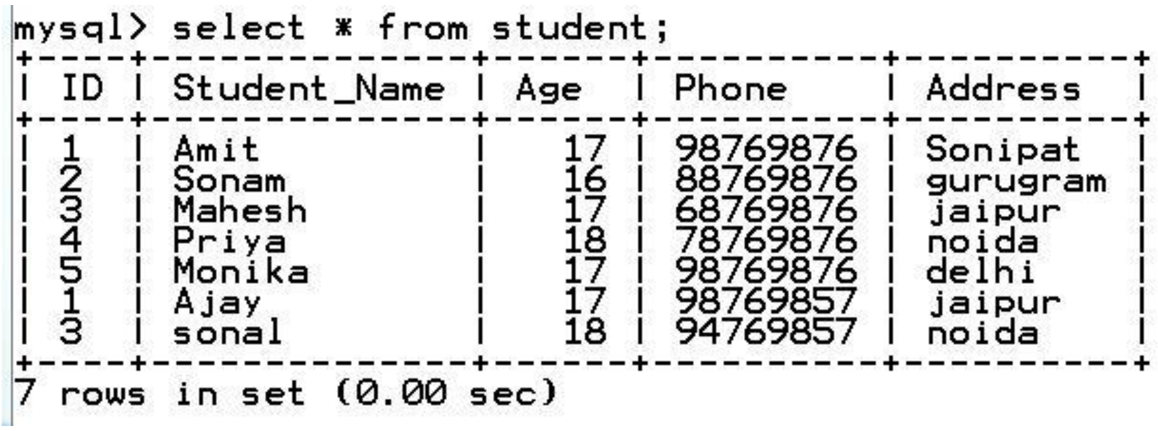
```
mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   17 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 4  | Priya        |   18 | 78769876 | noida    |
| 5  | Monika       |   17 | 98769876 | delhi    |
+----+--------------+------+----------+----------+
5 rows in set (0.00 sec)
```



```
mysql> select ID, Student_Name as Name from student;
+----+--------+
| ID | Name   |
+----+--------+
| 1  | Amit   |
| 2  | Sonam  |
| 3  | Mahesh |
| 4  | Priya  |
| 5  | Monika |
+----+--------+
5 rows in set (0.06 sec)
```

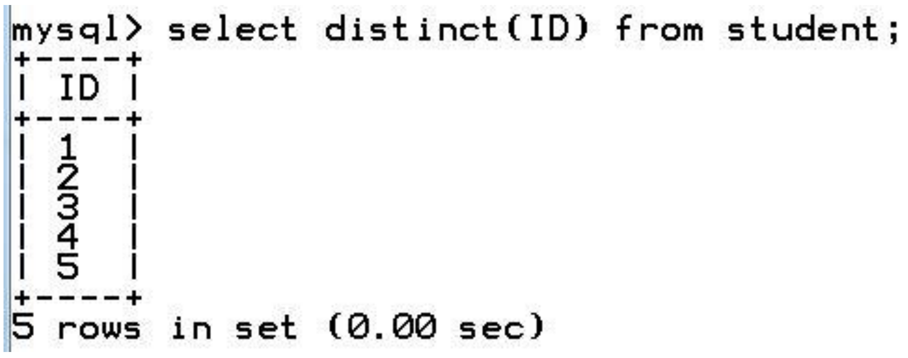
Distinct clause:

Distinct clause is used to remove duplicate values from the table. As we studied earlier, changes in data of a table is done using delete and update command. So, removing duplicate values using distinct clause is temporary and only reflected during output. Distinct clause can be used for more than one column.



```
mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   17 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 4  | Priya        |   18 | 78769876 | noida    |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 1  | Ajay         |   17 | 98769857 | jaipur   |
| 3  | sonal        |   18 | 94769857 | noida    |
+----+--------------+------+----------+----------+
7 rows in set (0.00 sec)
```

Here in student table, two duplicate ID’s 1 and 3 are present. Now using distinct clause we can get only unique values.



```
mysql> select distinct(ID) from student;
+----+
| ID |
+----+
| 1  |
| 2  |
| 3  |
| 4  |
| 5  |
+----+
5 rows in set (0.00 sec)
```

As we can see that all duplicate ID’s are removed but it is temporary. Duplicate values are not removed and still present in the table.

Where clause:

The WHERE clause in SQL is used to filter the results of a SELECT statement by specifying one or more conditions. All those tuples which meets the condition will be included in the final result.

The WHERE clause is a very powerful technique to select particular rows from a table. It can be used to filter by the values in a column, by the values in multiple columns, or by the outcome of any calculation.

Uses of where clause:

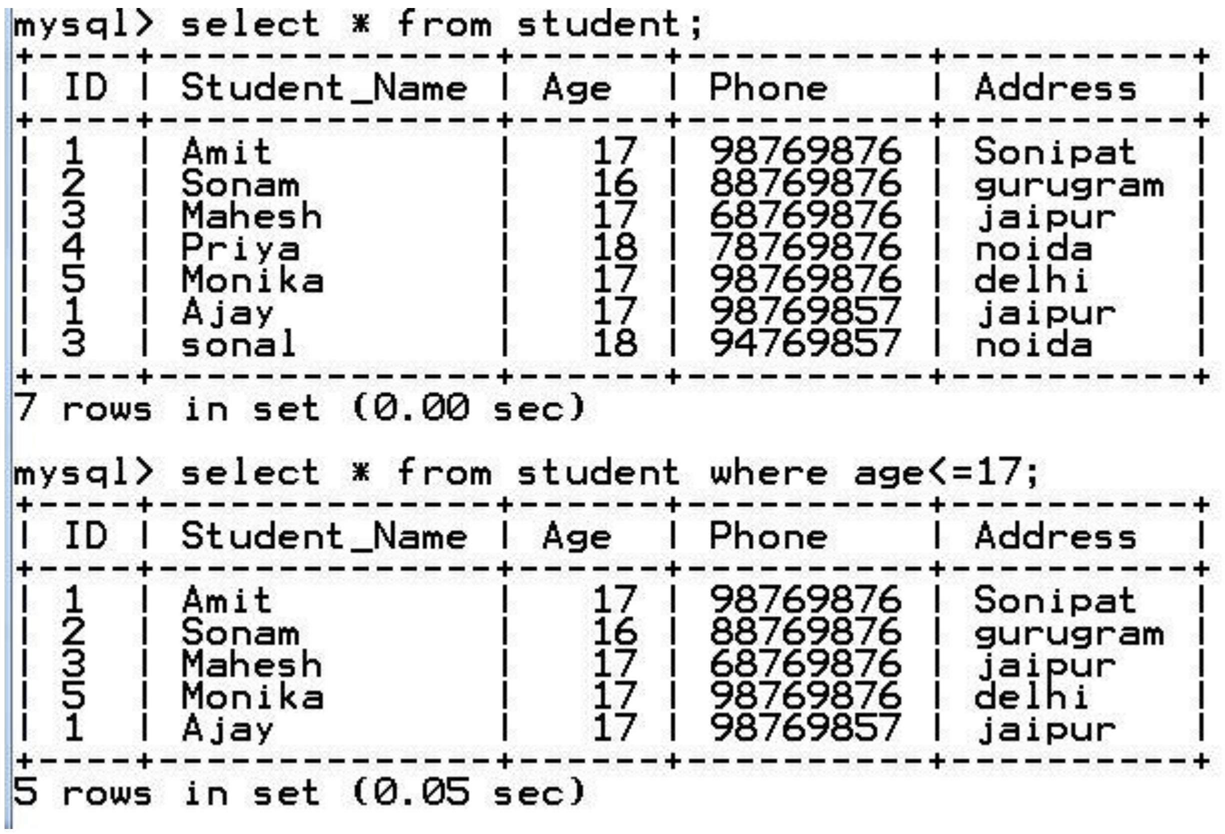
- Where clause can be used with select statement to filter the result.
- Where clause can be used with update statement to update the data of table that matches with the condition.
- Where clause can be used with delete statement to delete the rows of table that matches with the condition.

SQL Syntax:

where <condition>;

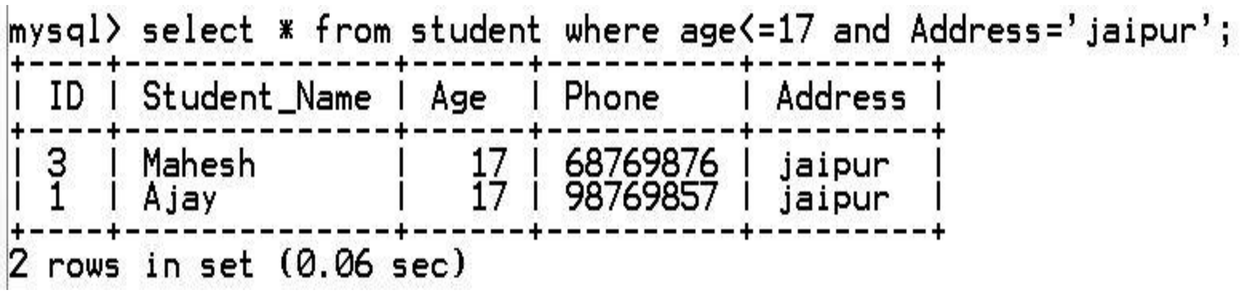
Examples:

1. To filter the result based on only one condition:



```
mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   17 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 4  | Priya        |   18 | 78769876 | noida    |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 1  | Ajay         |   17 | 98769857 | jaipur   |
| 3  | sonal        |   18 | 94769857 | noida    |
+----+--------------+------+----------+----------+
7 rows in set (0.00 sec)

mysql> select * from student where age<=17;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   17 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 1  | Ajay         |   17 | 98769857 | jaipur   |
+----+--------------+------+----------+----------+
5 rows in set (0.05 sec)
```

2. To filter the result based on multiple condition:


```
mysql> select * from student where age<=17 and Address='jaipur';
+----+--------------+------+----------+---------+
| ID | Student_Name | Age  | Phone    | Address |
+----+--------------+------+----------+---------+
| 3  | Mahesh       |   17 | 68769876 | jaipur  |
| 1  | Ajay         |   17 | 98769857 | jaipur  |
+----+--------------+------+----------+---------+
2 rows in set (0.06 sec)
```

In clause and not in clause:

In sql, **in clause** and **not in clause** is used to filter the rows in output based on list of values.

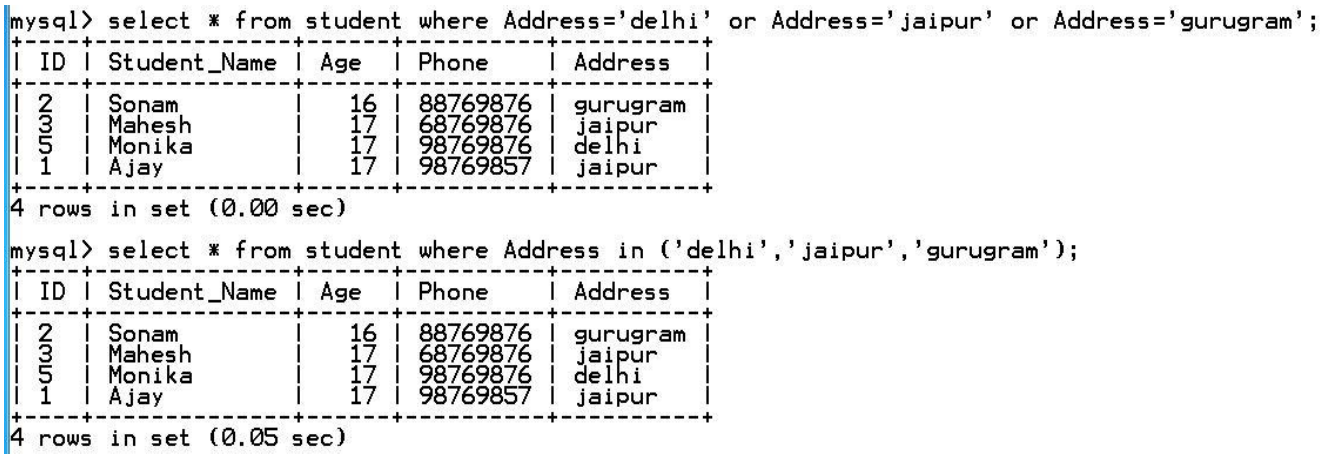
Syntax for in clause:

where <column name> in (item1, item2,…);

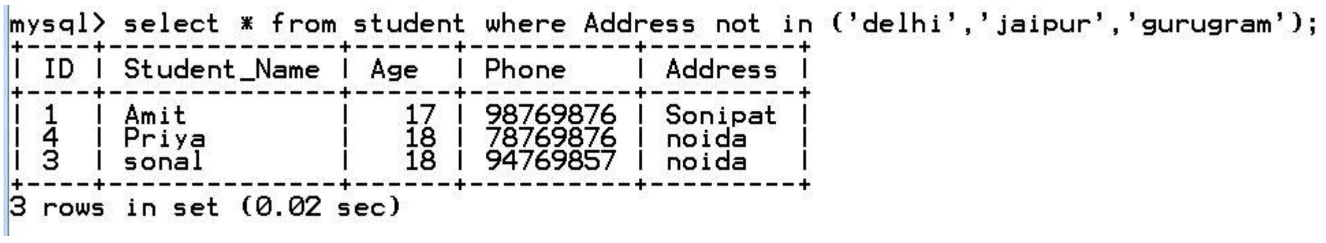
Syntax for not in clause:

where <column name> not in (item1, item2,…);

Example of in clause: if we want to find the data of those students who lives in either delhi or Jaipur or gurugram. Now to solve this problem, we have two ways. Either we write multiple comparison using **or keyword** or we can use **in clause**. Now you will see that using in clause for comparing with a list of items is an easy option.


```
mysql> select * from student where Address='delhi' or Address='jaipur' or Address='gurugram';
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 1  | Ajay         |   17 | 98769857 | jaipur   |
+----+--------------+------+----------+----------+
4 rows in set (0.00 sec)

mysql> select * from student where Address in ('delhi','jaipur','gurugram');
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 1  | Ajay         |   17 | 98769857 | jaipur   |
+----+--------------+------+----------+----------+
4 rows in set (0.05 sec)
```

Example of not in clause: if we want to find the data of those students who don't lives in delhi or Jaipur or gurugram.


```
mysql> select * from student where Address not in ('delhi','jaipur','gurugram');
+----+--------------+------+----------+---------+
| ID | Student_Name | Age  | Phone    | Address |
+----+--------------+------+----------+---------+
| 1  | Amit         |   17 | 98769876 | Sonipat |
| 4  | Priya        |   18 | 78769876 | noida   |
| 3  | sonal        |   18 | 94769857 | noida   |
+----+--------------+------+----------+---------+
3 rows in set (0.02 sec)
```

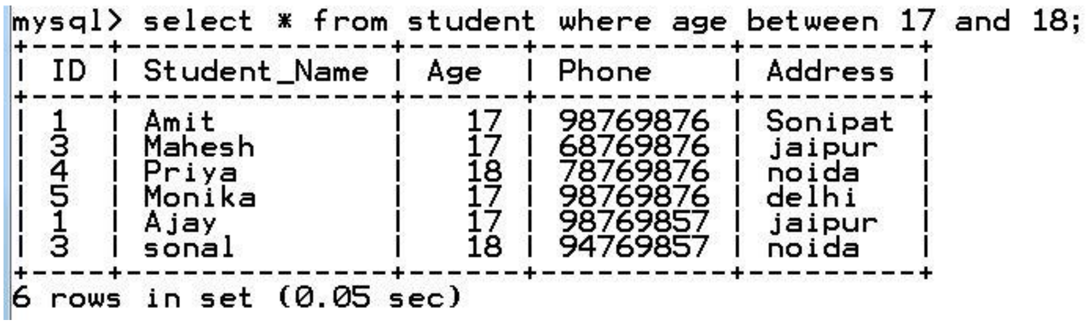
Between Clause:

It is used to filter the rows in output based on the range of values.

SQL Syntax:

where <column name> between <starting value> and <ending value>;

Note: The final result of between clause filters the rows of the table based on range of values including starting and ending value.



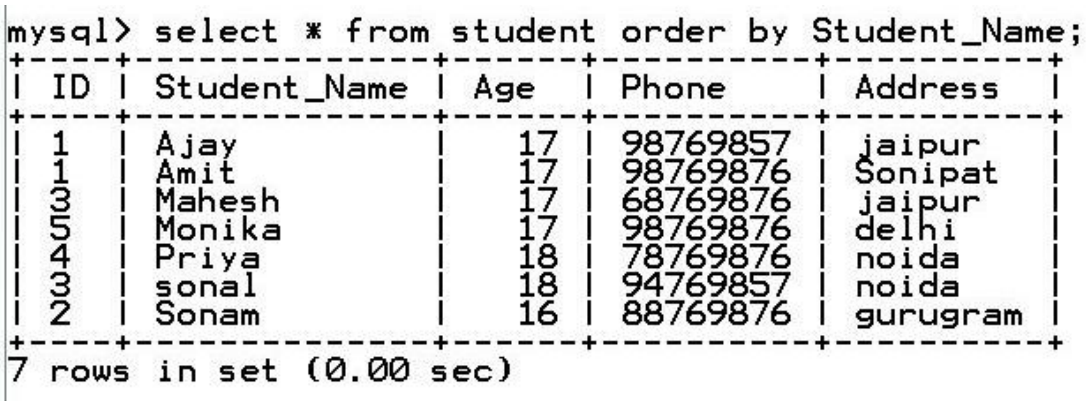
```
mysql> select * from student where age between 17 and 18;
+----+--------------+------+----------+---------+
| ID | Student_Name | Age  | Phone    | Address |
+----+--------------+------+----------+---------+
| 1  | Amit         |   17 | 98769876 | Sonipat |
| 3  | Mahesh       |   17 | 68769876 | jaipur  |
| 4  | Priya        |   18 | 78769876 | noida   |
| 5  | Monika       |   17 | 98769876 | delhi   |
| 1  | Ajay         |   17 | 98769857 | jaipur  |
| 3  | sonal        |   18 | 94769857 | noida   |
+----+--------------+------+----------+---------+
6 rows in set (0.05 sec)
```

Order by Clause: It is used to sort the output of the select statement in ascending or descending order.

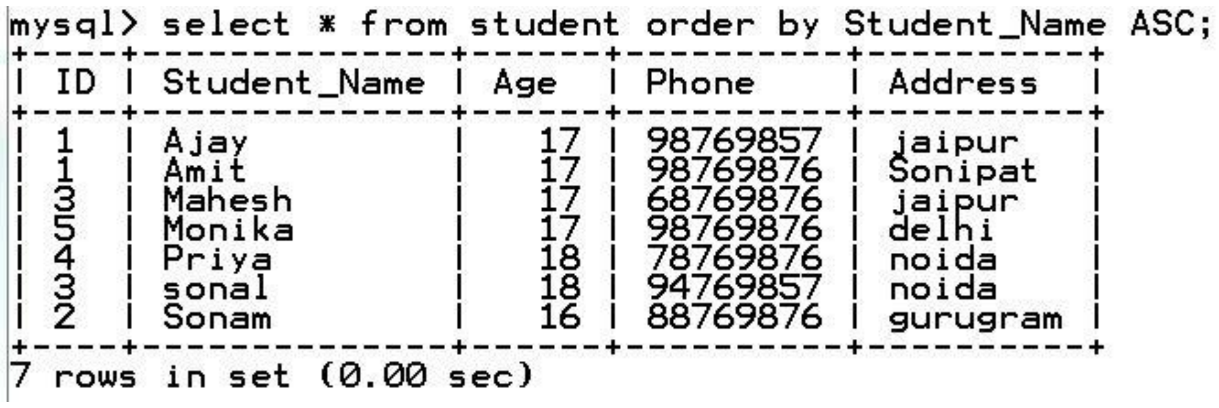
SQL Syntax:

order by <column name> [ASC|DESC];

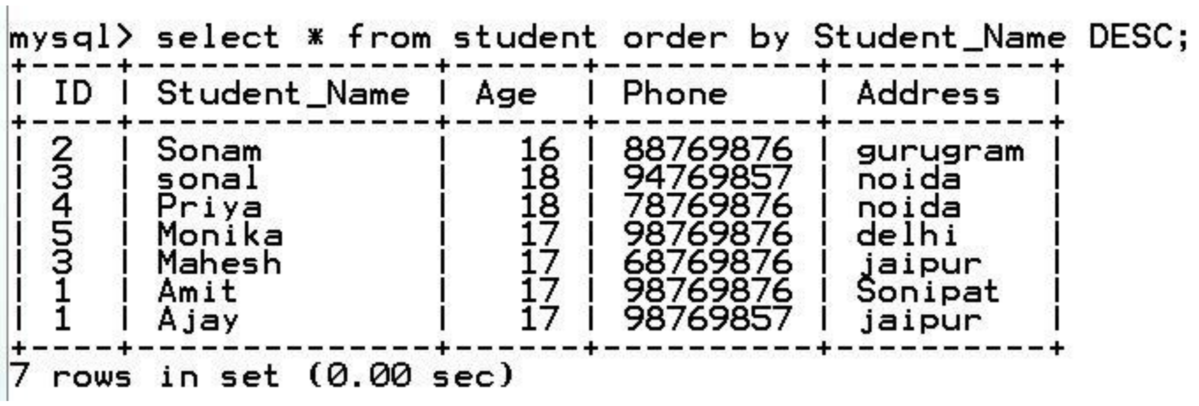
Note: If not mentioned, by default it will sort the output in ascending order. So, if you want to sort the data in ascending order, you need not to mention the order of sorting.



```
mysql> select * from student order by Student_Name;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Ajay         |   17 | 98769857 | jaipur   |
| 1  | Amit         |   17 | 98769876 | Sonipat  |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 4  | Priya        |   18 | 78769876 | noida    |
| 3  | sonal        |   18 | 94769857 | noida    |
| 2  | Sonam        |   16 | 88769876 | gurugram |
+----+--------------+------+----------+----------+
7 rows in set (0.00 sec)
```

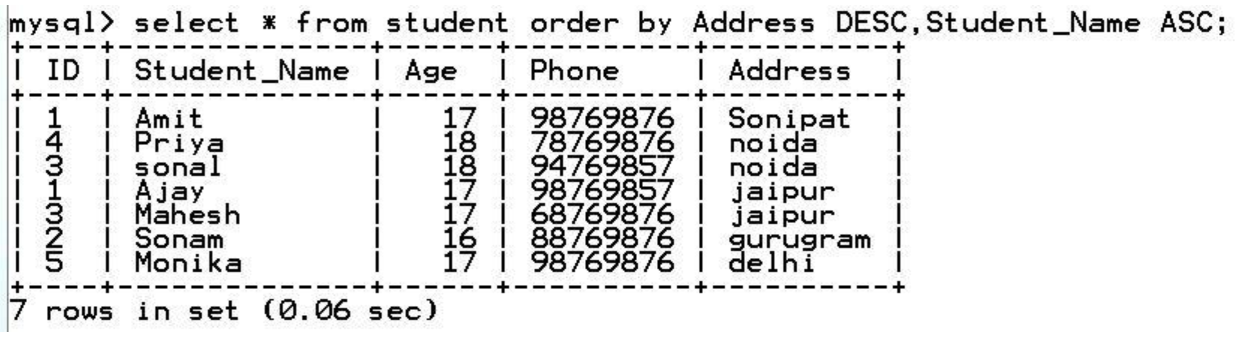


```
mysql> select * from student order by Student_Name ASC;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Ajay         |   17 | 98769857 | jaipur   |
| 1  | Amit         |   17 | 98769876 | Sonipat  |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 4  | Priya        |   18 | 78769876 | noida    |
| 3  | sonal        |   18 | 94769857 | noida    |
| 2  | Sonam        |   16 | 88769876 | gurugram |
+----+--------------+------+----------+----------+
7 rows in set (0.00 sec)
```



```
mysql> select * from student order by Student_Name DESC;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | sonal        |   18 | 94769857 | noida    |
| 4  | Priya        |   18 | 78769876 | noida    |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 1  | Amit         |   17 | 98769876 | Sonipat  |
| 1  | Ajay         |   17 | 98769857 | jaipur   |
+----+--------------+------+----------+----------+
7 rows in set (0.00 sec)
```

We can sort multiple columns together in ASC and DESC order.

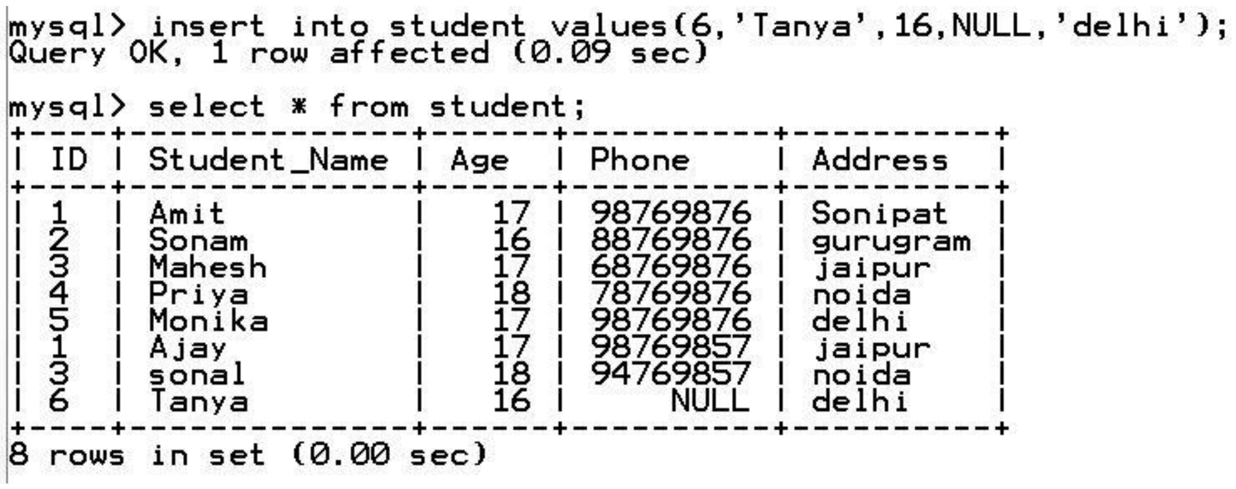


```
mysql> select * from student order by Address DESC,Student_Name ASC;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   17 | 98769876 | Sonipat  |
| 4  | Priya        |   18 | 78769876 | noida    |
| 3  | sonal        |   18 | 94769857 | noida    |
| 1  | Ajay         |   17 | 98769857 | jaipur   |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 5  | Monika       |   17 | 98769876 | delhi    |
+----+--------------+------+----------+----------+
7 rows in set (0.06 sec)
```

NULL:

In SQL, null is a special value which means absence of value or a field doesn't has a value. Null doesn't mean zero. Null also doesn't mean empty string. Null is a kind of placeholder of that value which is not present or not known.

Example: If the phone number of a student is not known at present, so we can store NULL instead of zero or make it empty.

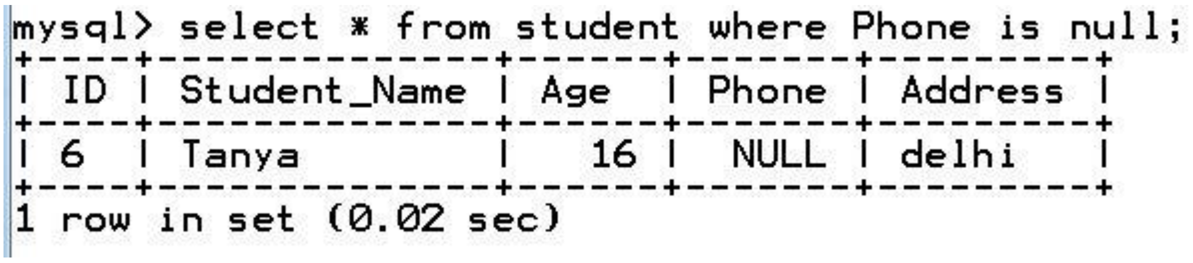


```
mysql> insert into student values(6,'Tanya',16,NULL,'delhi');
Query OK, 1 row affected (0.09 sec)

mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   17 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 4  | Priya        |   18 | 78769876 | noida    |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 1  | Ajay         |   17 | 98769857 | jaipur   |
| 3  | sonal        |   18 | 94769857 | noida    |
| 6  | Tanya        |   16 |     NULL | delhi    |
+----+--------------+------+----------+----------+
8 rows in set (0.00 sec)
```

IS NULL Clause:

IS NULL clause is used to check that value in particular column is NULL value

Example: To find out name of students whose phone number is NULL.



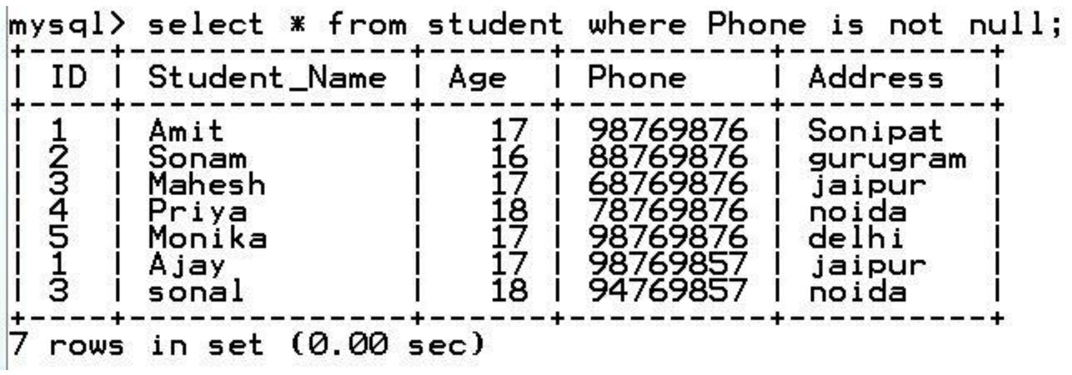
```
mysql> select * from student where Phone is null;
+----+--------------+------+-------+---------+
| ID | Student_Name | Age  | Phone | Address |
+----+--------------+------+-------+---------+
| 6  | Tanya        |   16 |  NULL | delhi   |
+----+--------------+------+-------+---------+
1 row in set (0.02 sec)
```

Note: is keyword is used to compare values of a column with **NULL**.

IS NOT NULL Clause:

IS NULL clause is used to check that value in particular column is not NULL value

Example: To find out name of students whose phone number is not NULL.



```
mysql> select * from student where Phone is not null;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   17 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 4  | Priya        |   18 | 78769876 | noida    |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 1  | Ajay         |   17 | 98769857 | jaipur   |
| 3  | sonal        |   18 | 94769857 | noida    |
+----+--------------+------+----------+----------+
7 rows in set (0.00 sec)
```

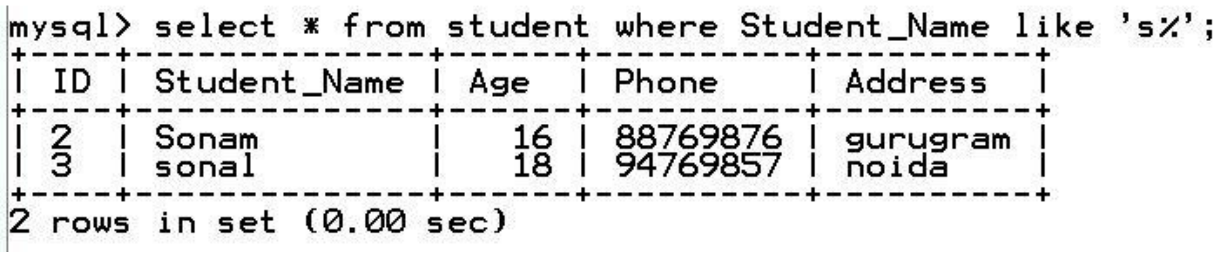
Like operator:

Like operator is used to match a pattern. The like operator is used with were clause. Like operator has 2 wildcards:

1. **_ (underscore):** It is used to match one character.

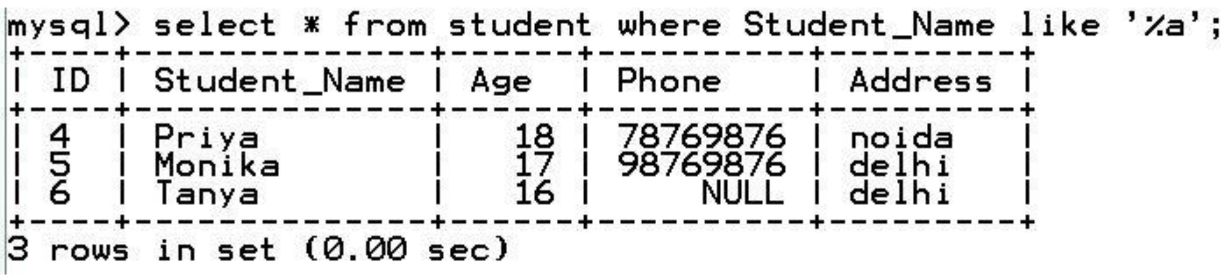
2. **% (percentage sign)**: It is used to match zero or more characters.

Example 1: To match a string that starts with ‘s’, its pattern will be ‘s%’. As we don’t know how many characters are there after ‘s’, so ‘%’ sign is used after ’s’.



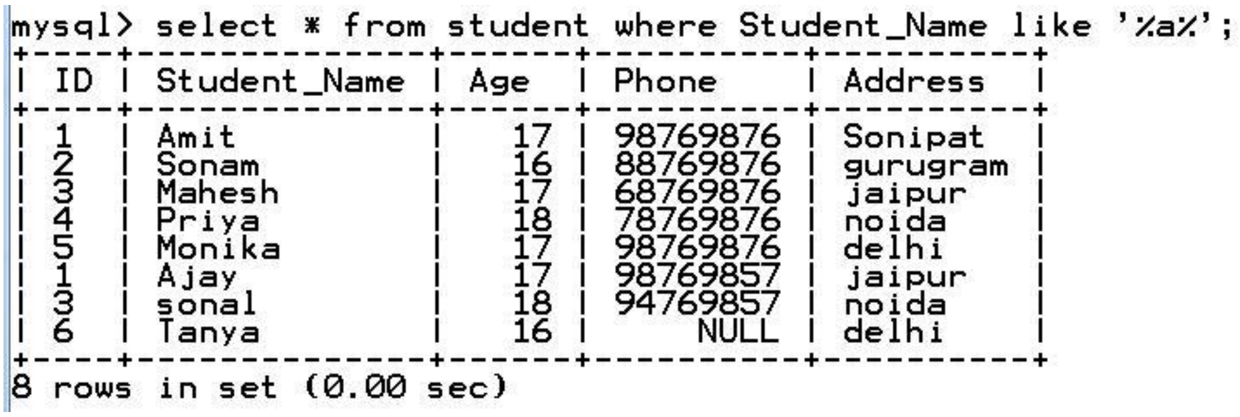
```
mysql> select * from student where Student_Name like 's%';
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | sonal        |   18 | 94769857 | noida    |
+----+--------------+------+----------+----------+
2 rows in set (0.00 sec)
```

Example 2: To match a string that ends with ‘a’, its pattern will be ‘%a’. As we don’t know how many characters are there before ‘a’, so ‘%’ sign is used before ’a’.



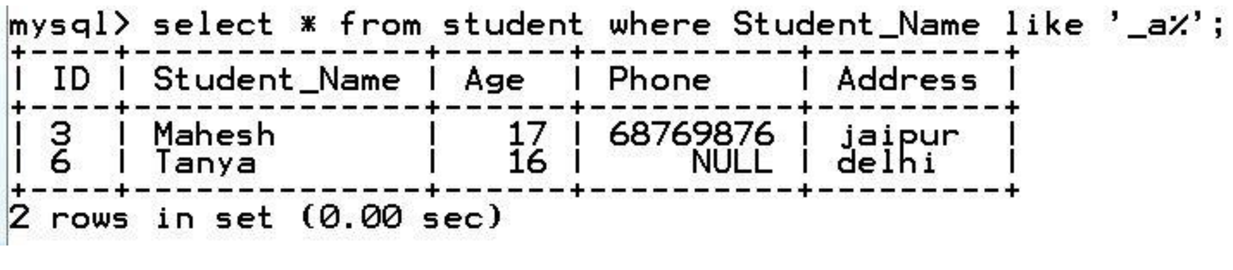
```
mysql> select * from student where Student_Name like '%a';
+----+--------------+------+----------+---------+
| ID | Student_Name | Age  | Phone    | Address |
+----+--------------+------+----------+---------+
| 4  | Priya        |   18 | 78769876 | noida   |
| 5  | Monika       |   17 | 98769876 | delhi   |
| 6  | Tanya        |   16 |     NULL | delhi   |
+----+--------------+------+----------+---------+
3 rows in set (0.00 sec)
```

Example 3: To match a string that contains with ‘a’, its pattern will be ‘%a%’. As we don’t know how many characters are there before or after ‘a’, so ‘%’ sign is used before and after ’a’.



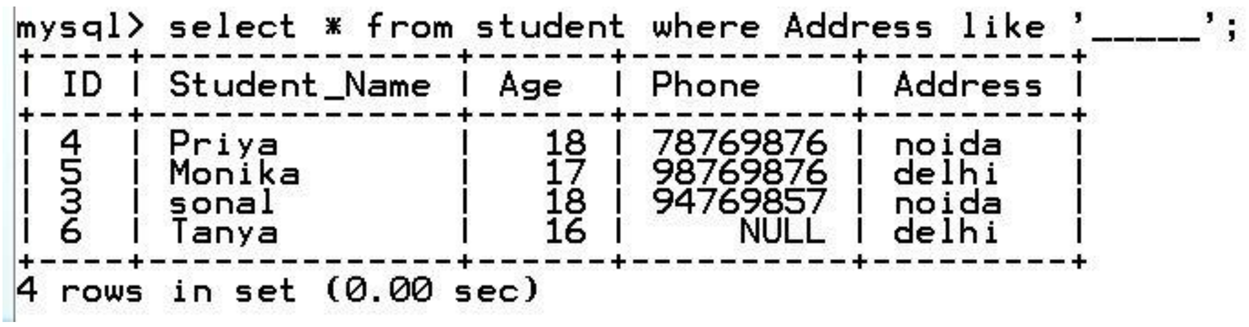
```
mysql> select * from student where Student_Name like '%a%';
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   17 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 4  | Priya        |   18 | 78769876 | noida    |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 1  | Ajay         |   17 | 98769857 | jaipur   |
| 3  | sonal        |   18 | 94769857 | noida    |
| 6  | Tanya        |   16 |     NULL | delhi    |
+----+--------------+------+----------+----------+
8 rows in set (0.00 sec)
```

Example 4: To match a string that has letter ‘a’ at second position, its pattern will be ‘_a%’. As we know there must be exact one character before ‘a’ and we don’t know how many characters are there after ‘a’, so ‘_’ sign is used before ‘a’ and ‘%’ sign is used after ’a’.



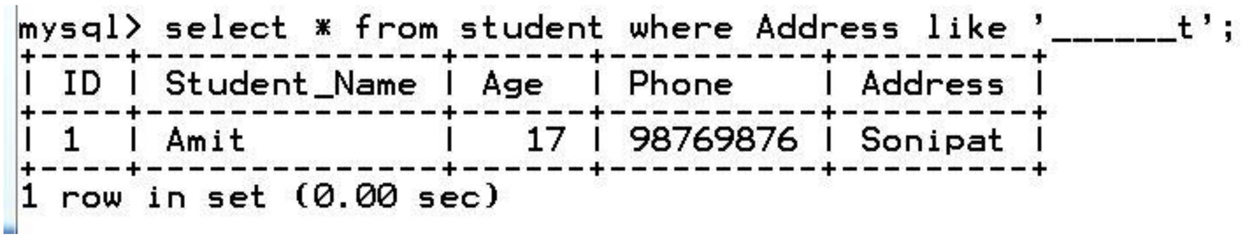
```
mysql> select * from student where Student_Name like '_a%';
+----+--------------+------+----------+---------+
| ID | Student_Name | Age  | Phone    | Address |
+----+--------------+------+----------+---------+
| 3  | Mahesh       |   17 | 68769876 | jaipur  |
| 6  | Tanya        |   16 |     NULL | delhi   |
+----+--------------+------+----------+---------+
2 rows in set (0.00 sec)
```

Example 5: To match a string that has exactly 5 character, its pattern will be '_ _ _ _ _'. As we know there must be exact 5 character, so '_' sign is used 5 times.



```
mysql> select * from student where Address like '_____';
+----+--------------+------+----------+---------+
| ID | Student_Name | Age  | Phone    | Address |
+----+--------------+------+----------+---------+
| 4  | Priya        |   18 | 78769876 | noida   |
| 5  | Monika       |   17 | 98769876 | delhi   |
| 3  | sonal        |   18 | 94769857 | noida   |
| 6  | Tanya        |   16 |     NULL | delhi   |
+----+--------------+------+----------+---------+
4 rows in set (0.00 sec)
```

Example 6: To match a string that has exactly 7 character and ends with 't', its pattern will be '_ _ _ _ _ _ t'. As we know there must be exact 7 character, so '_' sign is used 6 times before 't'.



```
mysql> select * from student where Address like '______t';
+----+--------------+------+----------+---------+
| ID | Student_Name | Age  | Phone    | Address |
+----+--------------+------+----------+---------+
| 1  | Amit         |   17 | 98769876 | Sonipat |
+----+--------------+------+----------+---------+
1 row in set (0.00 sec)
```

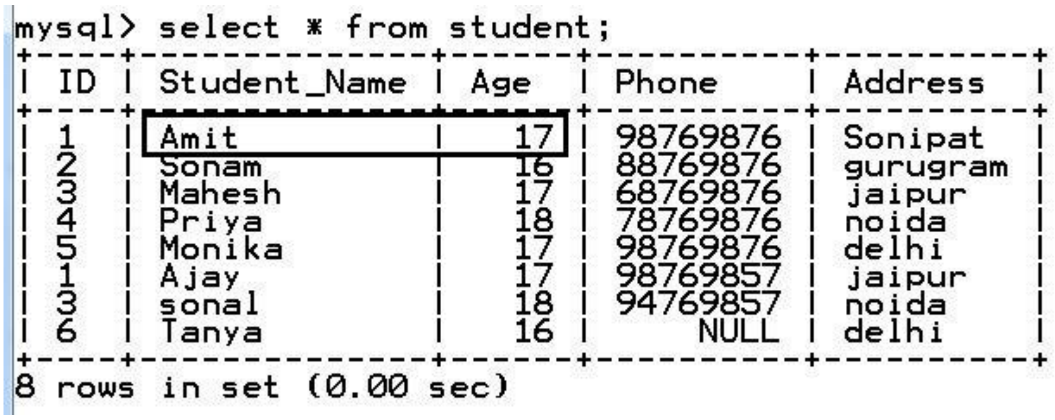
Update Command:

It is used to update the existing data in a table.

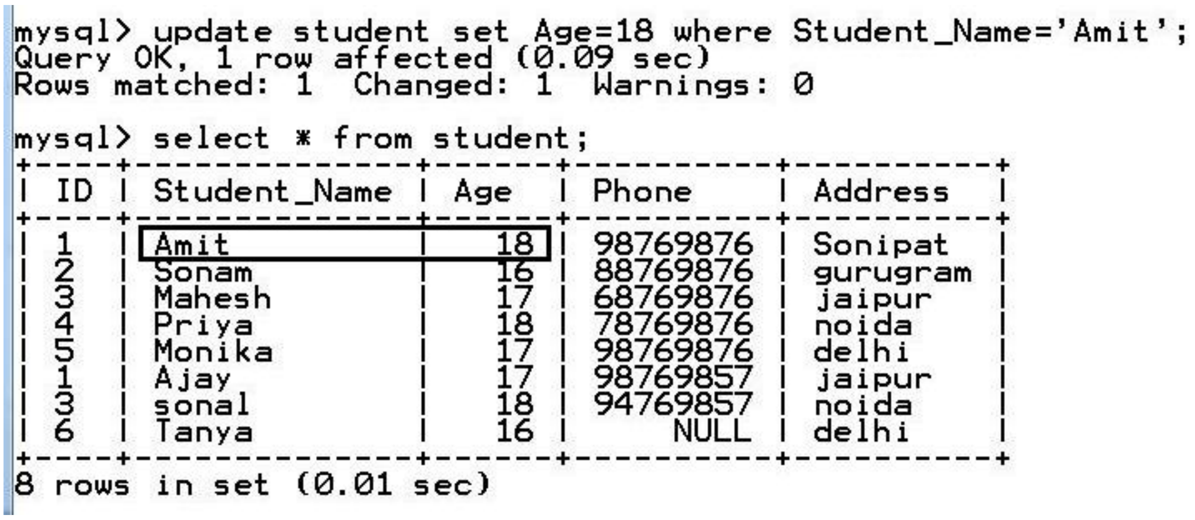
SQL Syntax:

update <table name> set <column name> = <new data> where <condition>;

Example 1: Let's update the Age to 18 of that student whose name is Amit.

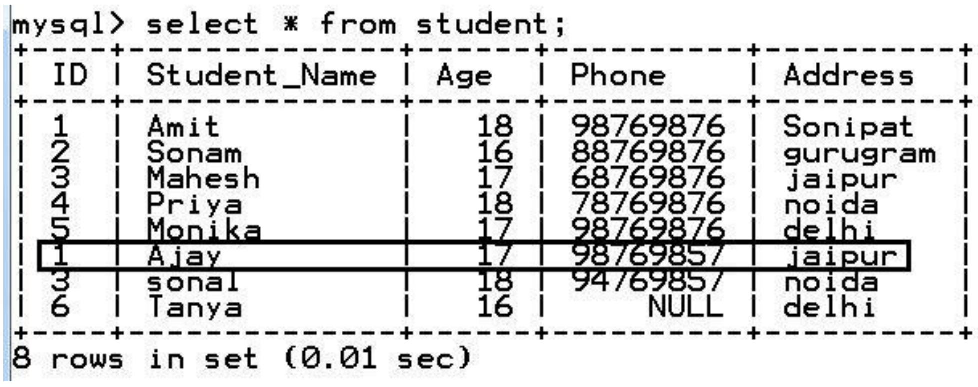


```
mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   17 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 4  | Priya        |   18 | 78769876 | noida    |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 1  | Ajay         |   17 | 98769857 | jaipur   |
| 3  | sonal        |   18 | 94769857 | noida    |
| 6  | Tanya        |   16 |     NULL | delhi    |
+----+--------------+------+----------+----------+
8 rows in set (0.00 sec)
```

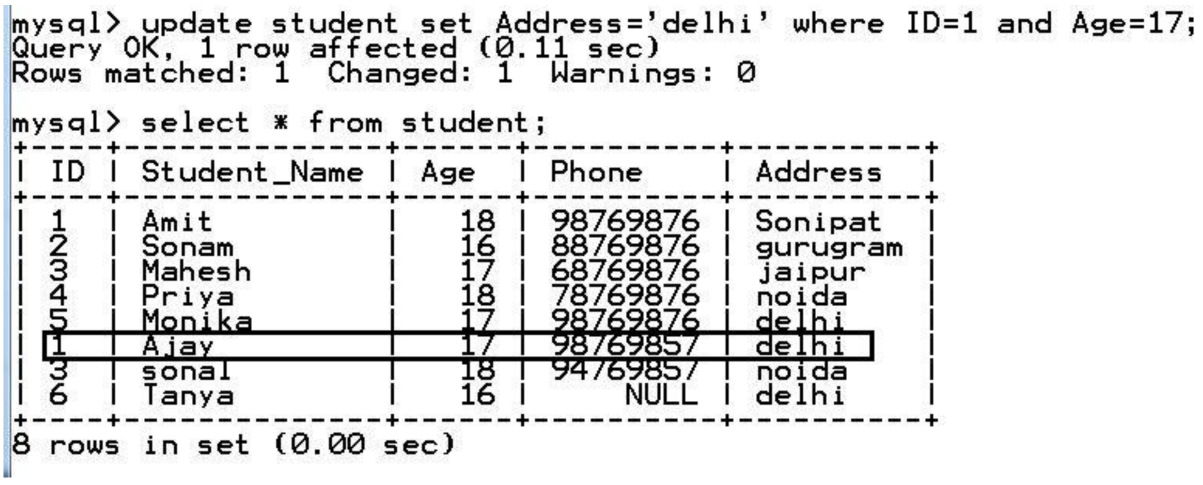


```
mysql> update student set Age=18 where Student_Name='Amit';
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from student;
+----+----------------+------+----------+----------+
| ID | Student_Name   | Age  | Phone    | Address  |
+----+----------------+------+----------+----------+
| 1  | Amit           |   18 | 98769876 | Sonipat  |
| 2  | Sonam          |   16 | 88769876 | gurugram |
| 3  | Mahesh         |   17 | 68769876 | jaipur   |
| 4  | Priya          |   18 | 78769876 | noida    |
| 5  | Monika         |   17 | 98769876 | delhi    |
| 1  | Ajay           |   17 | 98769857 | jaipur   |
| 3  | sonal          |   18 | 94769857 | noida    |
| 6  | Tanya          |   16 |     NULL | delhi    |
+----+----------------+------+----------+----------+
8 rows in set (0.01 sec)
```

Example 2: Let’s update the city to delhi of that student whose ID is 1 and Age is 17.



```
mysql> select * from student;
+----+----------------+------+----------+----------+
| ID | Student_Name   | Age  | Phone    | Address  |
+----+----------------+------+----------+----------+
| 1  | Amit           |   18 | 98769876 | Sonipat  |
| 2  | Sonam          |   16 | 88769876 | gurugram |
| 3  | Mahesh         |   17 | 68769876 | jaipur   |
| 4  | Priya          |   18 | 78769876 | noida    |
| 5  | Monika         |   17 | 98769876 | delhi    |
| 1  | Ajay           |   17 | 98769857 | jaipur   |
| 3  | sonal          |   18 | 94769857 | noida    |
| 6  | Tanya          |   16 |     NULL | delhi    |
+----+----------------+------+----------+----------+
8 rows in set (0.01 sec)
```



```
mysql> update student set Address='delhi' where ID=1 and Age=17;
Query OK, 1 row affected (0.11 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from student;
+----+----------------+------+----------+----------+
| ID | Student_Name   | Age  | Phone    | Address  |
+----+----------------+------+----------+----------+
| 1  | Amit           |   18 | 98769876 | Sonipat  |
| 2  | Sonam          |   16 | 88769876 | gurugram |
| 3  | Mahesh         |   17 | 68769876 | jaipur   |
| 4  | Priya          |   18 | 78769876 | noida    |
| 5  | Monika         |   17 | 98769876 | delhi    |
| 1  | Ajay           |   17 | 98769857 | delhi    |
| 3  | sonal          |   18 | 94769857 | noida    |
| 6  | Tanya          |   16 |     NULL | delhi    |
+----+----------------+------+----------+----------+
8 rows in set (0.00 sec)
```

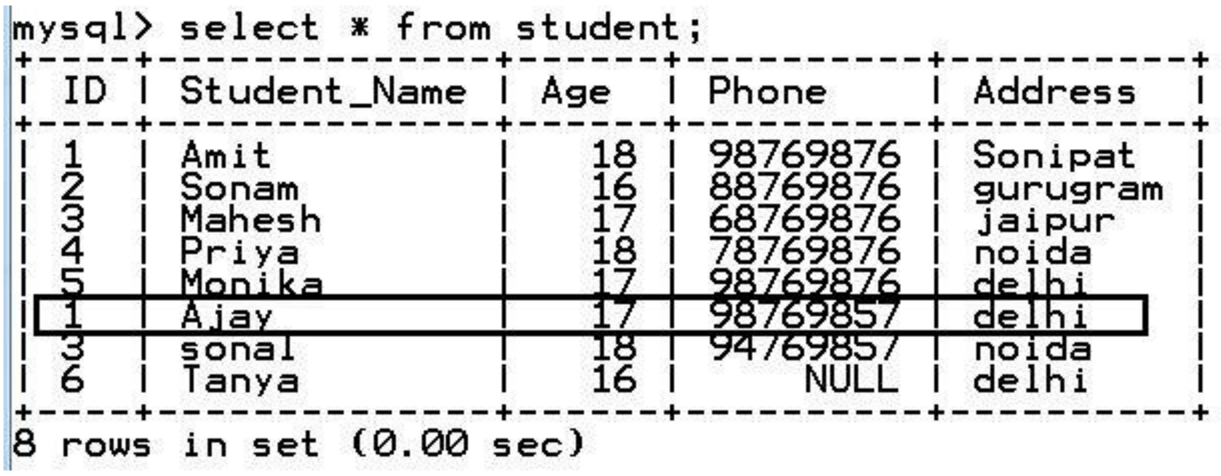
Delete Command:

It is used to delete the existing rows in a table that matches the condition.

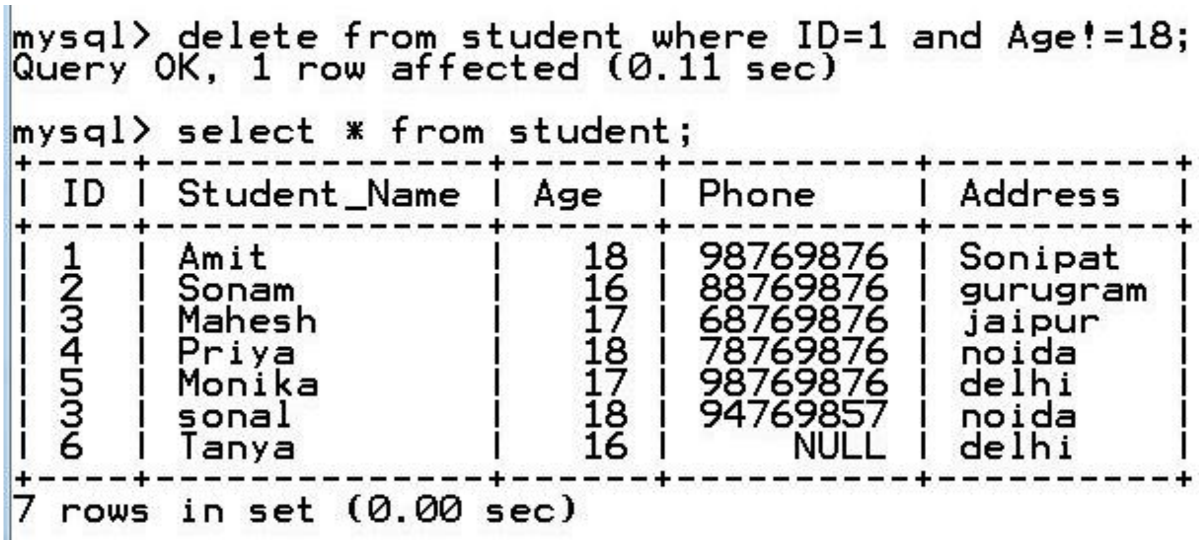
SQL Syntax:

delete from <table name> where <condition>;

Example 1: Let's delete data of those students whose ID is 1 but age is not 18.

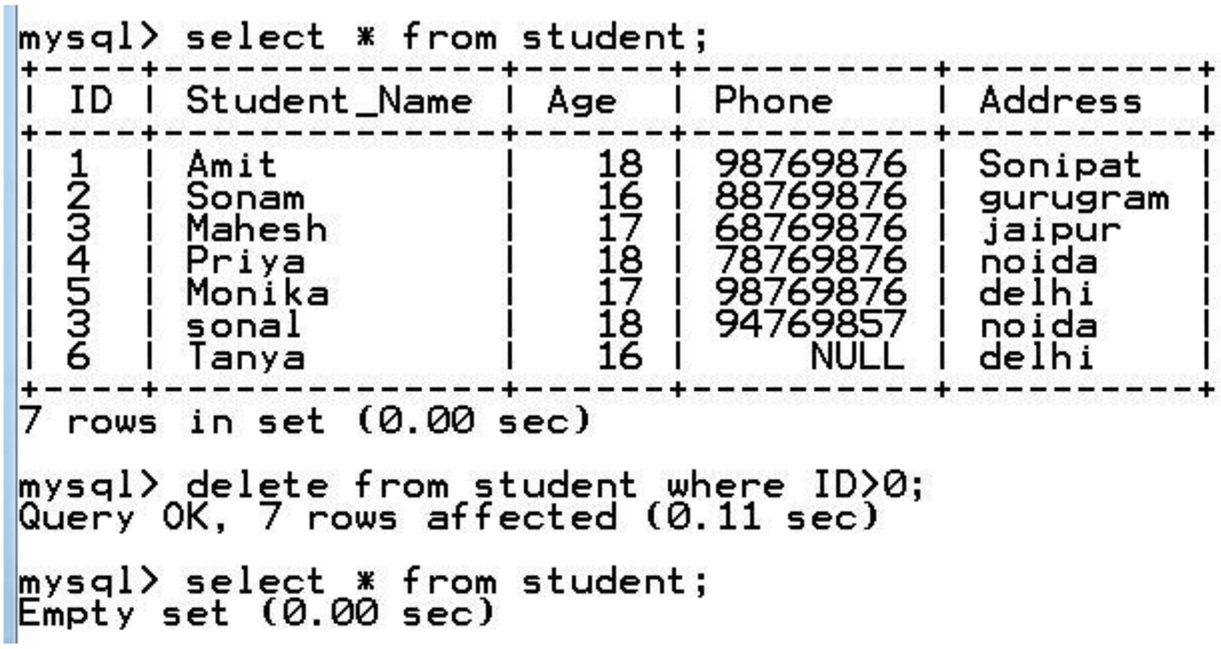


```
mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   18 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 4  | Priya        |   18 | 78769876 | noida    |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 1  | Ajay         |   17 | 98769857 | delhi    |
| 3  | sonal        |   18 | 94769857 | noida    |
| 6  | Tanya        |   16 |     NULL | delhi    |
+----+--------------+------+----------+----------+
8 rows in set (0.00 sec)
```



```
mysql> delete from student where ID=1 and Age!=18;
Query OK, 1 row affected (0.11 sec)

mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   18 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 4  | Priya        |   18 | 78769876 | noida    |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 3  | sonal        |   18 | 94769857 | noida    |
| 6  | Tanya        |   16 |     NULL | delhi    |
+----+--------------+------+----------+----------+
7 rows in set (0.00 sec)
```

Example 2: Let's delete all data of student table. For doing those, we needs to give a condition that matches with all the records. As ID's are greater than 0, so let's delete all those records where ID is greater than 0.

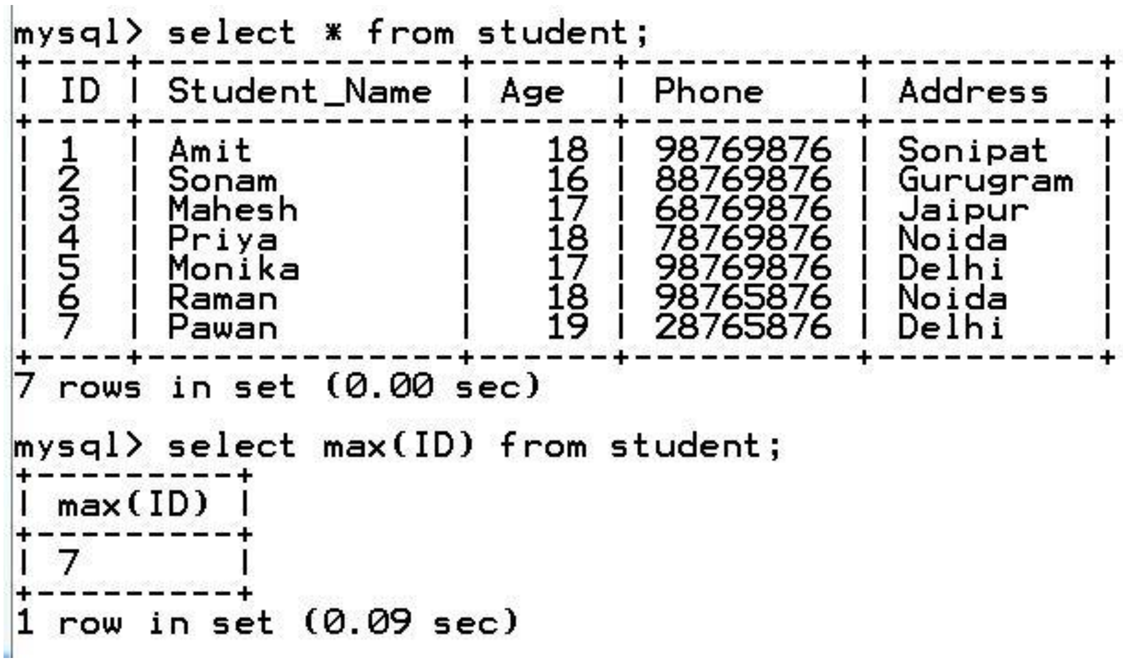


```
mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   18 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | gurugram |
| 3  | Mahesh       |   17 | 68769876 | jaipur   |
| 4  | Priya        |   18 | 78769876 | noida    |
| 5  | Monika       |   17 | 98769876 | delhi    |
| 3  | sonal        |   18 | 94769857 | noida    |
| 6  | Tanya        |   16 |     NULL | delhi    |
+----+--------------+------+----------+----------+
7 rows in set (0.00 sec)

mysql> delete from student where ID>0;
Query OK, 7 rows affected (0.11 sec)

mysql> select * from student;
Empty set (0.00 sec)
```

Aggregate Functions:

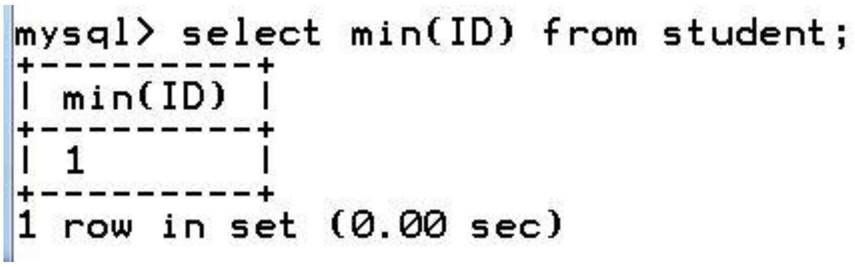
Aggregate functions are those functions that operates on a list of values and returns a single digit value or we can summarize the data using aggregate functions.

1. **Max()**:
   It is used to find out the maximum value from a column.



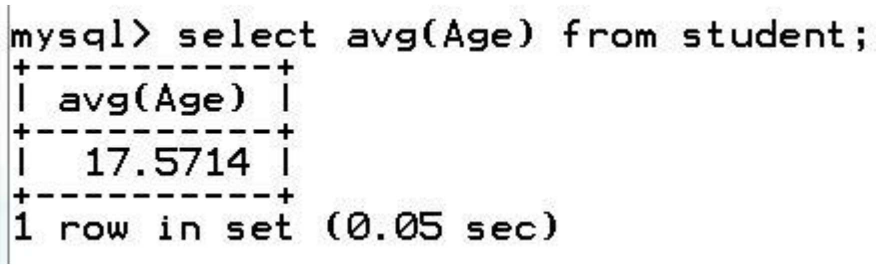
```
mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   18 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | Gurugram |
| 3  | Mahesh       |   17 | 68769876 | Jaipur   |
| 4  | Priya        |   18 | 78769876 | Noida    |
| 5  | Monika       |   17 | 98769876 | Delhi    |
| 6  | Raman        |   18 | 98765876 | Noida    |
| 7  | Pawan        |   19 | 28765876 | Delhi    |
+----+--------------+------+----------+----------+
7 rows in set (0.00 sec)

mysql> select max(ID) from student;
+---------+
| max(ID) |
+---------+
| 7       |
+---------+
1 row in set (0.09 sec)
```

2. **Min()**:
   It is used to find out the minimum value from a column.



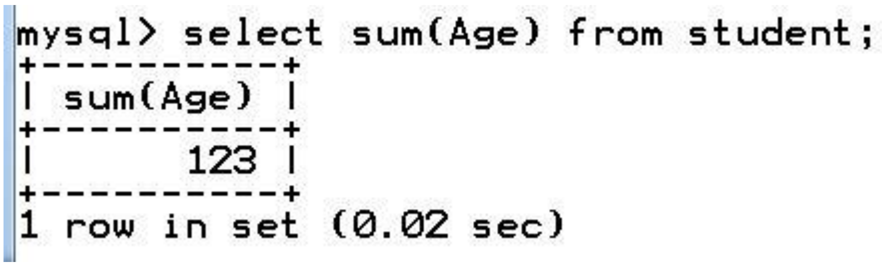
```
mysql> select min(ID) from student;
+---------+
| min(ID) |
+---------+
| 1       |
+---------+
1 row in set (0.00 sec)
```

3. **Avg()**:
   It is used to find out the average value from a column.



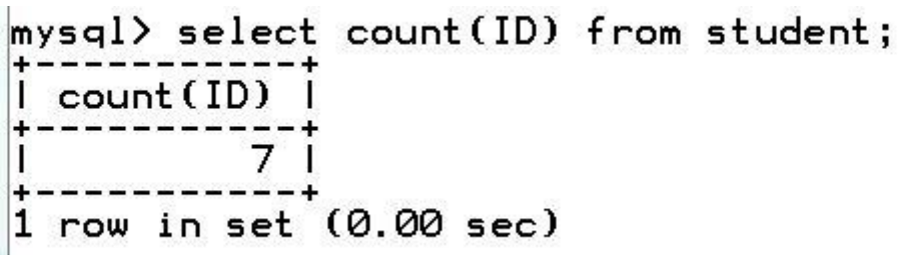
```
mysql> select avg(Age) from student;
+----------+
| avg(Age) |
+----------+
|  17.5714 |
+----------+
1 row in set (0.05 sec)
```

4. **Sum()**:
   It is used to find out the sum of all values of a column.



```
mysql> select sum(Age) from student;
+----------+
| sum(Age) |
+----------+
|      123 |
+----------+
1 row in set (0.02 sec)
```

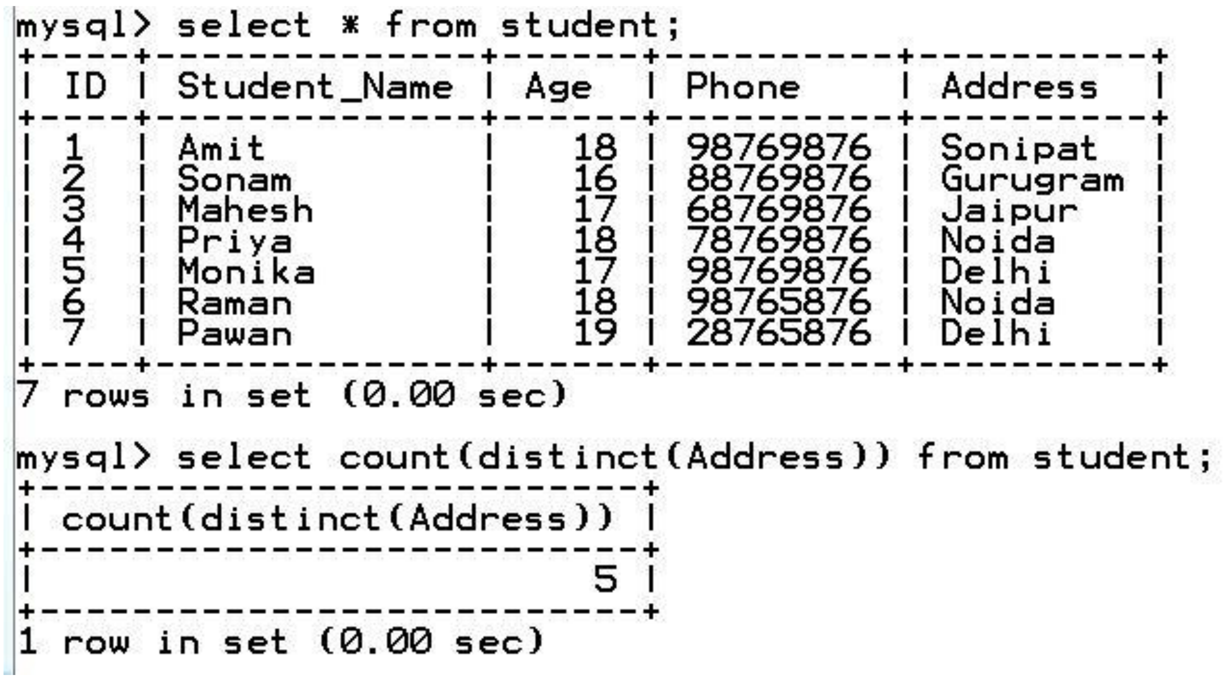
5. **Count**: it is used to count number of values in a column.



```
mysql> select count(ID) from student;
+-----------+
| count(ID) |
+-----------+
|         7 |
+-----------+
1 row in set (0.00 sec)
```

Note: Distinct keyword can be used with aggregate functions to find out max, min, sum, avg, count of only unique values.

Example: Let's find out total number of cities from where student came for study. Here more than one student is from same city. So we needs to use distinct keyword along with count function.



```
mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   18 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | Gurugram |
| 3  | Mahesh       |   17 | 68769876 | Jaipur   |
| 4  | Priya        |   18 | 78769876 | Noida    |
| 5  | Monika       |   17 | 98769876 | Delhi    |
| 6  | Raman        |   18 | 98765876 | Noida    |
| 7  | Pawan        |   19 | 28765876 | Delhi    |
+----+--------------+------+----------+----------+
7 rows in set (0.00 sec)

mysql> select count(distinct(Address)) from student;
+--------------------------+
| count(distinct(Address)) |
+--------------------------+
|                        5 |
+--------------------------+
1 row in set (0.00 sec)
```

Group by clause:

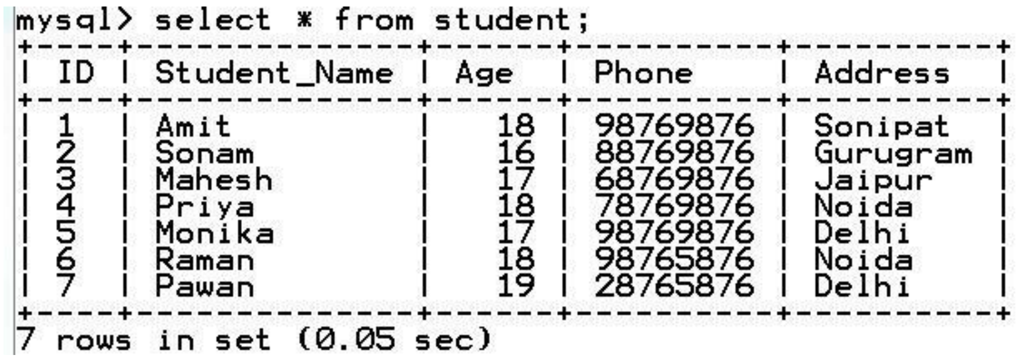
The GROUP BY clause is used to group rows that have the same values into summary rows. Group by clause is often used with aggregate functions like MAX(), MIN(), SUM(), AVG() and COUNT() to group the result by one or more columns.

- It can be used with or without where clause in select statement.
- It is applied only on numeric values.
- It can’t be applied with distinct keyword.

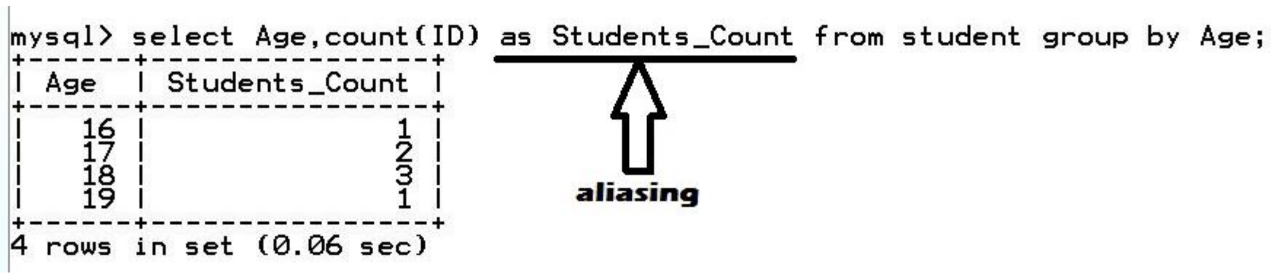
SQL Syntax:

group by <column name>

Example 1: Let’s count number of students having same age in student table.

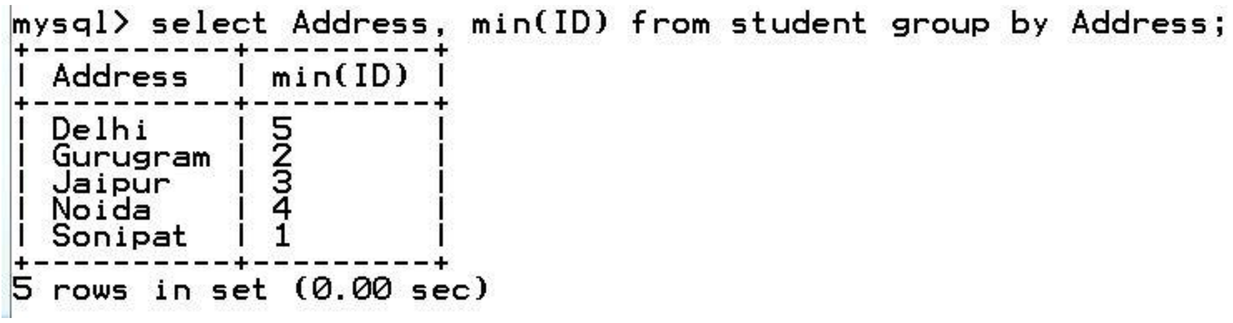


```
mysql> select * from student;
+----+--------------+------+----------+----------+
| ID | Student_Name | Age  | Phone    | Address  |
+----+--------------+------+----------+----------+
| 1  | Amit         |   18 | 98769876 | Sonipat  |
| 2  | Sonam        |   16 | 88769876 | Gurugram |
| 3  | Mahesh       |   17 | 68769876 | Jaipur   |
| 4  | Priya        |   18 | 78769876 | Noida    |
| 5  | Monika       |   17 | 98769876 | Delhi    |
| 6  | Raman        |   18 | 98765876 | Noida    |
| 7  | Pawan        |   19 | 28765876 | Delhi    |
+----+--------------+------+----------+----------+
7 rows in set (0.05 sec)
```



```
mysql> select Age,count(ID) as Students_Count from student group by Age;
+------+----------------+
| Age  | Students_Count |
+------+----------------+
|   16 |              1 |
|   17 |              2 |
|   18 |              3 |
|   19 |              1 |
+------+----------------+
4 rows in set (0.06 sec)
```

Example 2: Let's city wise find out the minimum value of ID.

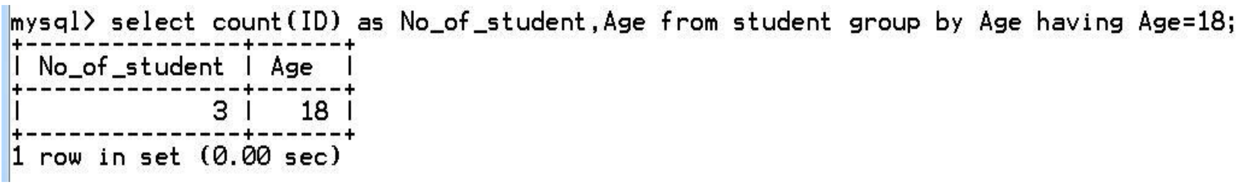


```
mysql> select Address, min(ID) from student group by Address;
+----------+---------+
| Address  | min(ID) |
+----------+---------+
| Delhi    | 5       |
| Gurugram | 2       |
| Jaipur   | 3       |
| Noida    | 4       |
| Sonipat  | 1       |
+----------+---------+
5 rows in set (0.00 sec)
```

Having clause:

It is used to filter the result set of group by clause in select statement.

Note: To filter the result set of group by clause, only having clause can be used whereas for all other queries where clause is used.



```
mysql> select count(ID) as No_of_student,Age from student group by Age having Age=18;
+---------------+------+
| No_of_student | Age  |
+---------------+------+
|             3 |   18 |
+---------------+------+
1 row in set (0.00 sec)
```

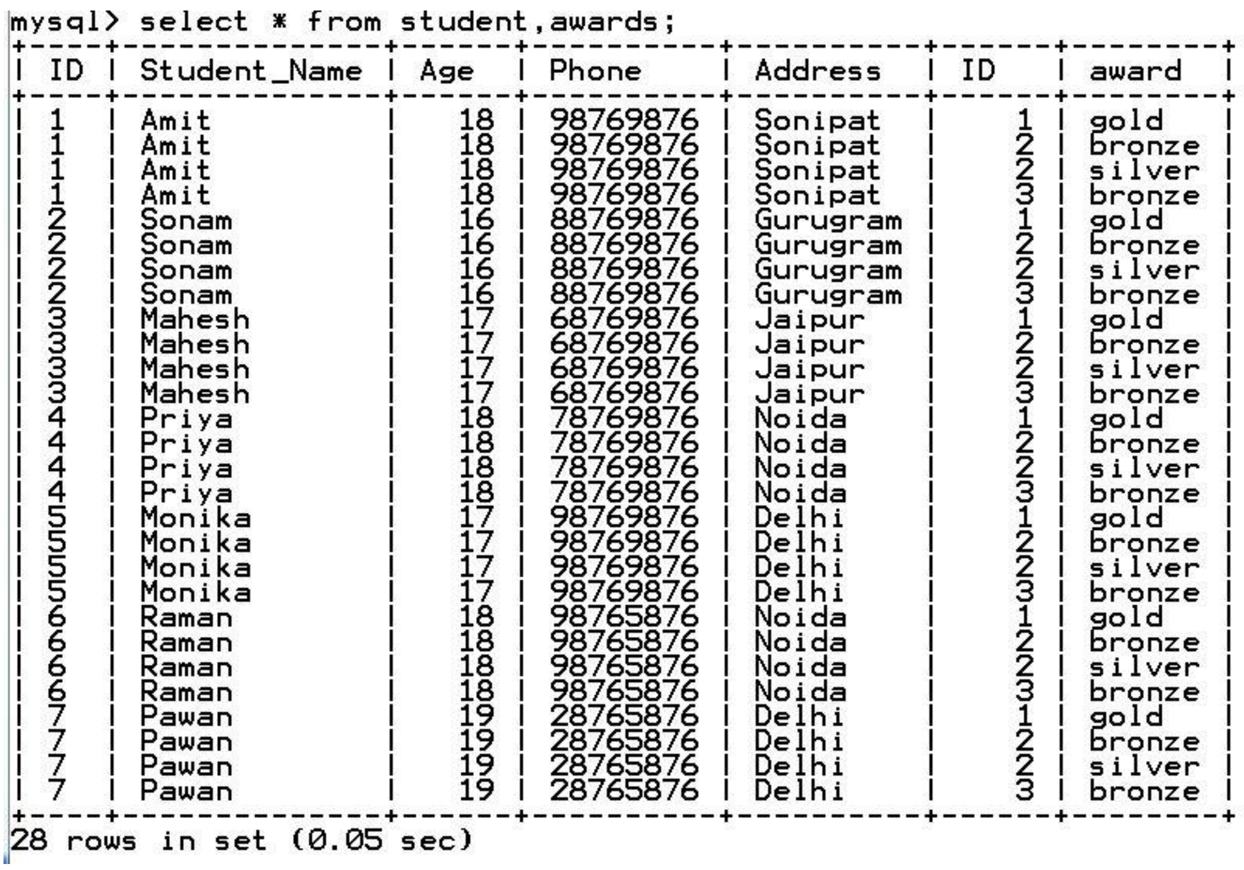
Joins:

Joins are used to combine rows from multiple tables.

Types of joins:

1. **Cartesian product (Cross Join):**
   It gives all possible combinations from more than one table. It combines every row from one table with every row from another table. Suppose we have 5 rows in first table and 4 rows in second table then the total number of rows in Cartesian product of these two tables will be 20 rows.
   Cardinality of final table of Cartesian product = cardinality of first table * cardinality of second table

   Example: we have two tables' student and awards. Let's apply Cartesian product on these two tables.

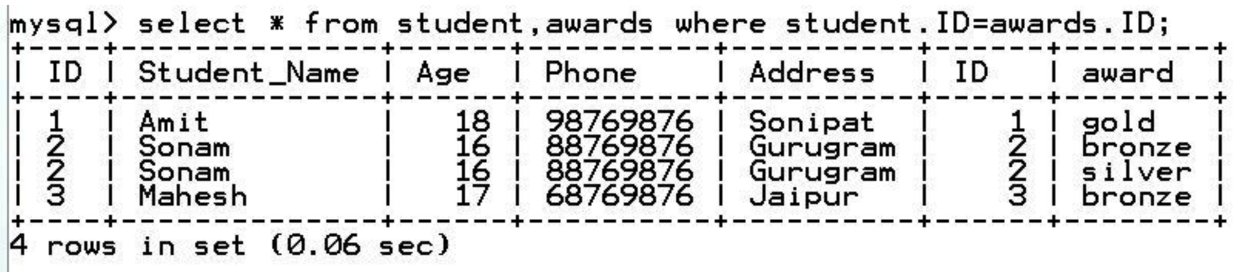


```
mysql> select * from student,awards;
+----+--------------+------+----------+----------+------+--------+
| ID | Student_Name | Age  | Phone    | Address  | ID   | award  |
+----+--------------+------+----------+----------+------+--------+
| 1  | Amit         |   18 | 98769876 | Sonipat  |    1 | gold   |
| 1  | Amit         |   18 | 98769876 | Sonipat  |    2 | bronze |
| 1  | Amit         |   18 | 98769876 | Sonipat  |    2 | silver |
| 1  | Amit         |   18 | 98769876 | Sonipat  |    3 | bronze |
| 2  | Sonam        |   16 | 88769876 | Gurugram |    1 | gold   |
| 2  | Sonam        |   16 | 88769876 | Gurugram |    2 | bronze |
| 2  | Sonam        |   16 | 88769876 | Gurugram |    2 | silver |
| 2  | Sonam        |   16 | 88769876 | Gurugram |    3 | bronze |
| 3  | Mahesh       |   17 | 68769876 | Jaipur   |    1 | gold   |
| 3  | Mahesh       |   17 | 68769876 | Jaipur   |    2 | bronze |
| 3  | Mahesh       |   17 | 68769876 | Jaipur   |    2 | silver |
| 3  | Mahesh       |   17 | 68769876 | Jaipur   |    3 | bronze |
| 4  | Priya        |   18 | 78769876 | Noida    |    1 | gold   |
| 4  | Priya        |   18 | 78769876 | Noida    |    2 | bronze |
| 4  | Priya        |   18 | 78769876 | Noida    |    2 | silver |
| 4  | Priya        |   18 | 78769876 | Noida    |    3 | bronze |
| 5  | Monika       |   17 | 98769876 | Delhi    |    1 | gold   |
| 5  | Monika       |   17 | 98769876 | Delhi    |    2 | bronze |
| 5  | Monika       |   17 | 98769876 | Delhi    |    2 | silver |
| 5  | Monika       |   17 | 98769876 | Delhi    |    3 | bronze |
| 6  | Raman        |   18 | 98765876 | Noida    |    1 | gold   |
| 6  | Raman        |   18 | 98765876 | Noida    |    2 | bronze |
| 6  | Raman        |   18 | 98765876 | Noida    |    2 | silver |
| 6  | Raman        |   18 | 98765876 | Noida    |    3 | bronze |
| 7  | Pawan        |   19 | 28765876 | Delhi    |    1 | gold   |
| 7  | Pawan        |   19 | 28765876 | Delhi    |    2 | bronze |
| 7  | Pawan        |   19 | 28765876 | Delhi    |    2 | silver |
| 7  | Pawan        |   19 | 28765876 | Delhi    |    3 | bronze |
+----+--------------+------+----------+----------+------+--------+
28 rows in set (0.05 sec)
```

2. Equi join:

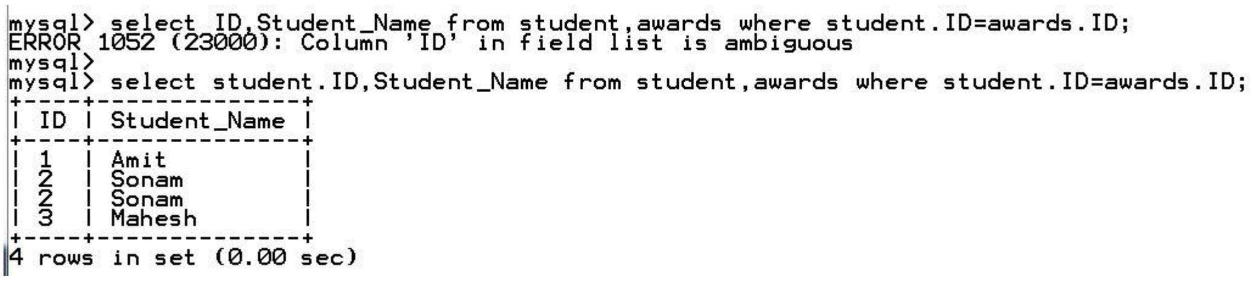
It joins the tables based on one common column. However, final result will consists of common column from both the tables.

Example: we have two tables' student and awards. Let's apply equi join on these two tables.



```
mysql> select * from student,awards where student.ID=awards.ID;
+----+--------------+------+----------+----------+------+--------+
| ID | Student_Name | Age  | Phone    | Address  | ID   | award  |
+----+--------------+------+----------+----------+------+--------+
| 1  | Amit         |   18 | 98769876 | Sonipat  |    1 | gold   |
| 2  | Sonam        |   16 | 88769876 | Gurugram |    2 | bronze |
| 2  | Sonam        |   16 | 88769876 | Gurugram |    2 | silver |
| 3  | Mahesh       |   17 | 68769876 | Jaipur   |    3 | bronze |
+----+--------------+------+----------+----------+------+--------+
4 rows in set (0.06 sec)
```

Here both the tables has common column ID. So, to avoid ambiguity (confusion), we needs to mention table name before column name.

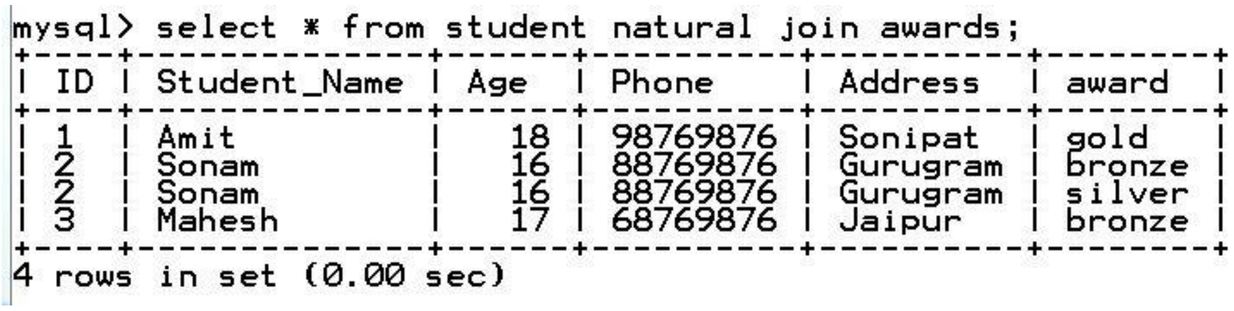


```
mysql> select ID,Student_Name from student,awards where student.ID=awards.ID;
ERROR 1052 (23000): Column 'ID' in field list is ambiguous
mysql>
mysql> select student.ID,Student_Name from student,awards where student.ID=awards.ID;
+----+--------------+
| ID | Student_Name |
+----+--------------+
| 1  | Amit         |
| 2  | Sonam        |
| 2  | Sonam        |
| 3  | Mahesh       |
+----+--------------+
4 rows in set (0.00 sec)
```

3. Natural Join:

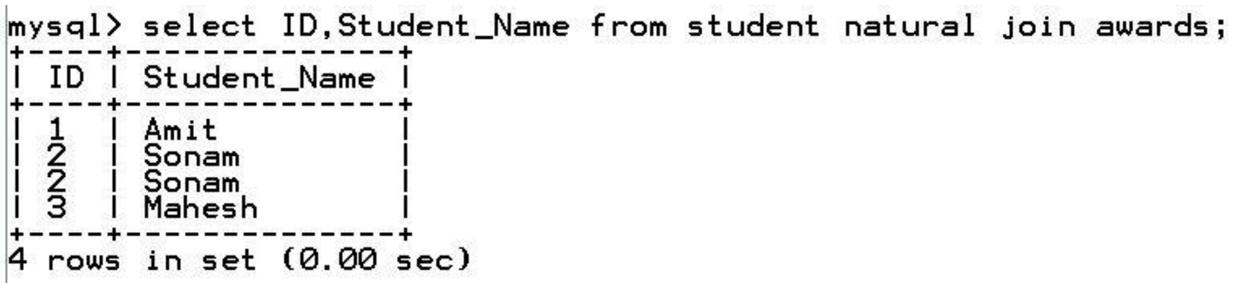
It joins the tables based on one common column. However, final result will consists of common column only once.

Example: we have two tables' student and awards. Let's apply natural join on these two tables.



```
mysql> select * from student natural join awards;
+----+--------------+------+----------+----------+--------+
| ID | Student_Name | Age  | Phone    | Address  | award  |
+----+--------------+------+----------+----------+--------+
| 1  | Amit         |   18 | 98769876 | Sonipat  | gold   |
| 2  | Sonam        |   16 | 88769876 | Gurugram | bronze |
| 2  | Sonam        |   16 | 88769876 | Gurugram | silver |
| 3  | Mahesh       |   17 | 68769876 | Jaipur   | bronze |
+----+--------------+------+----------+----------+--------+
4 rows in set (0.00 sec)
```

Here both the tables has common column ID. But there is no ambiguity arises on name of common column.



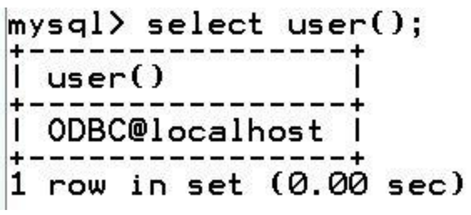
```
mysql> select ID,Student_Name from student natural join awards;
+----+--------------+
| ID | Student_Name |
+----+--------------+
| 1  | Amit         |
| 2  | Sonam        |
| 2  | Sonam        |
| 3  | Mahesh       |
+----+--------------+
4 rows in set (0.00 sec)
```

Python and MySQL Interface

In real life scenario, user interact with applications whereas data is stored in database. We have already developed python applications and database tables. Now, we will connect python application and MySQL database.

Steps for creating python and MySQL connectivity application:

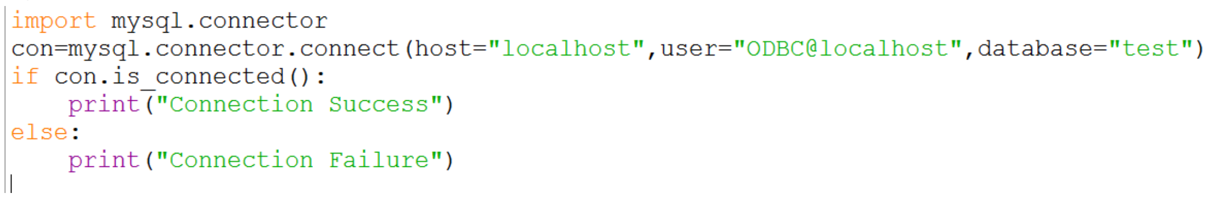
1. Import mysql.connector
2. Create a connection
   <connection object> =mysql.connector.connect (host="localhost", user=<username>, passwd= <password>, database =<database name>)
   Username = To know the user name, run select user(); command on your MySQL.



```
mysql> select user();
+-----------------+
| user()          |
+-----------------+
| ODBC@localhost  |
+-----------------+
1 row in set (0.00 sec)
```

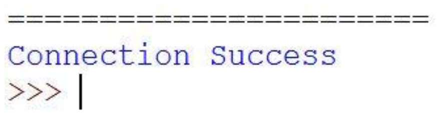
Password: It should be same as we set during MySQL installation.
Example: Let's create a connection of python with MySQL and test it.

Python Code:



```python
import mysql.connector
con=mysql.connector.connect(host="localhost",user="ODBC@localhost",database="test")
if con.is_connected():
    print("Connection Success")
else:
    print("Connection Failure")
```

Output:

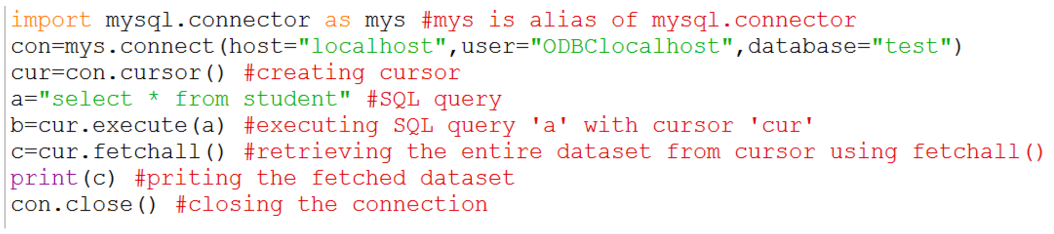


```
========================
Connection Success
>>> 
```

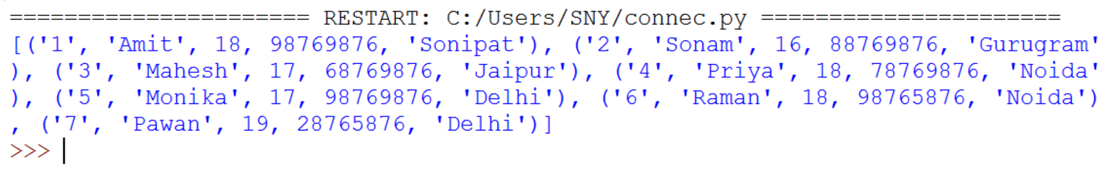
Here in above example, I had not set any database password, so, passwd option is not mentioned in connection string. Is_connected() function checks whether connection string is able to connect python with MySQL. If connection string is able to make connect python and MySQL, is_connected() returns True otherwise False.

3. Create a cursor
   Cursor: It is a pointer or iterator which points towards the resultset of the SQL query. Whenever a SQL query runs, It give the entire result set in one go. We may not require the entire resultset at once. So, a cursor is created and data from the entire resultset will be fetched row by row as per our requirement.
   Syntax: <cursor object> = <connection object>.cursor()

4. Execute query
   Syntax: <cursor object> . execute(<SQL query string>)
5. Extract data from result set
   As we know that, Data from database is retrieved using select query. After running the select query, we get the resultset. Now to fetch the data from resultset, following functions are used:
   a) fetchall(): It returns all the records from resultset. Each individual record will be in the form of a tuple whereas the entire resultset will be in the form of a list.
      Syntax: <variable name>=<cursor>.fetchall()



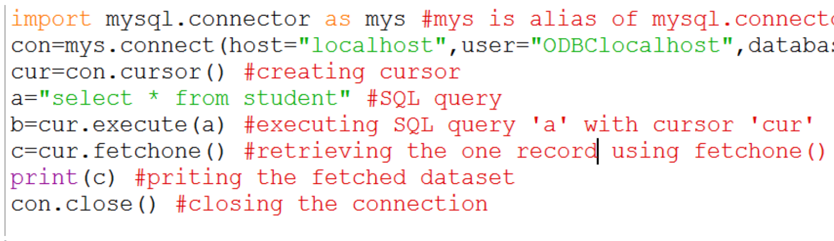
```
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
a="select * from student" #SQL query
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
c=cur.fetchall() #retrieving the entire dataset from cursor using fetchall()
print(c) #priting the fetched dataset
con.close() #closing the connection
```



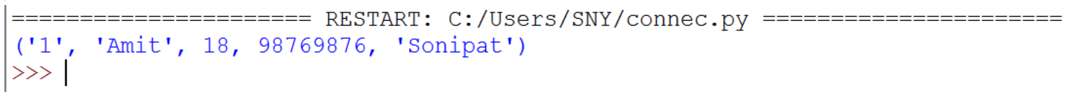
```
====================== RESTART: C:/Users/SNY/connec.py ======================
[('1', 'Amit', 18, 98769876, 'Sonipat'), ('2', 'Sonam', 16, 88769876, 'Gurugram'
), ('3', 'Mahesh', 17, 68769876, 'Jaipur'), ('4', 'Priya', 18, 78769876, 'Noida'
), ('5', 'Monika', 17, 98769876, 'Delhi'), ('6', 'Raman', 18, 98765876, 'Noida')
, ('7', 'Pawan', 19, 28765876, 'Delhi')]
>>> 
```

b) fetchone(): It returns one row from resultset in the form of a tuple. It returns None, if no more records are there. To get multiple rows, we needs to run fetchone() multiple times.

Syntax: <variable name>=<cursor>.fetchone()



```
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
a="select * from student" #SQL query
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
c=cur.fetchone() #retrieving the one record using fetchone()
print(c) #priting the fetched dataset
con.close() #closing the connection
```

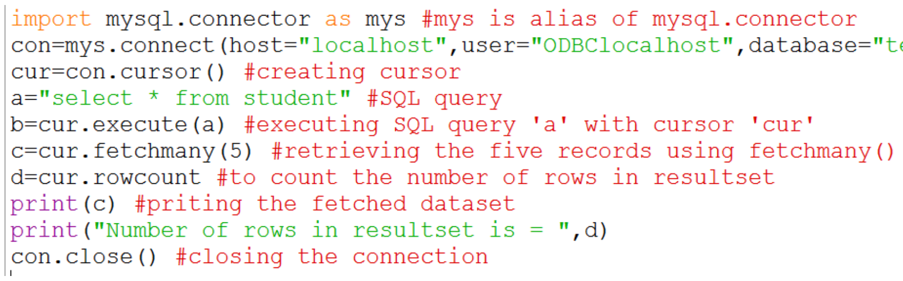


```
====================== RESTART: C:/Users/SNY/connec.py ======================
('1', 'Amit', 18, 98769876, 'Sonipat')
>>> 
```

c) fetchmany(): It returns n number of records from resultset in the form of a list where each individual record is in the form of a tuple. It returns empty tuple, if no more records are there.

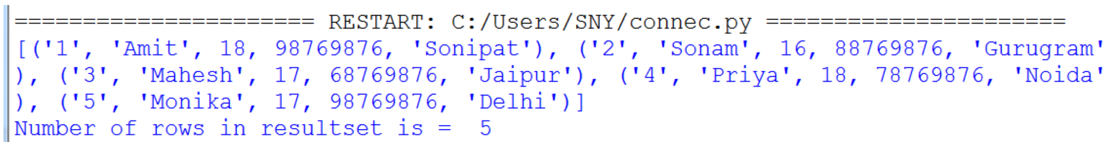
Syntax: <variable name>=<cursor>.fetchmany(n)

Python Code:



```
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
a="select * from student" #SQL query
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
c=cur.fetchmany(5) #retrieving the five records using fetchmany()
d=cur.rowcount #to count the number of rows in resultset
print(c) #priting the fetched dataset
print("Number of rows in resultset is = ",d)
con.close() #closing the connection
```

Output :



```
====================== RESTART: C:/Users/SNY/connec.py ======================
[('1', 'Amit', 18, 98769876, 'Sonipat'), ('2', 'Sonam', 16, 88769876, 'Gurugram'
), ('3', 'Mahesh', 17, 68769876, 'Jaipur'), ('4', 'Priya', 18, 78769876, 'Noida'
), ('5', 'Monika', 17, 98769876, 'Delhi')]
Number of rows in resultset is =  5
```

d) **rowcount:** It is cursor's property to count the number of rows in resultset.
Syntax: <variable name>=<cursor>.rowcount
Example: Refer to the example of fetchmany()

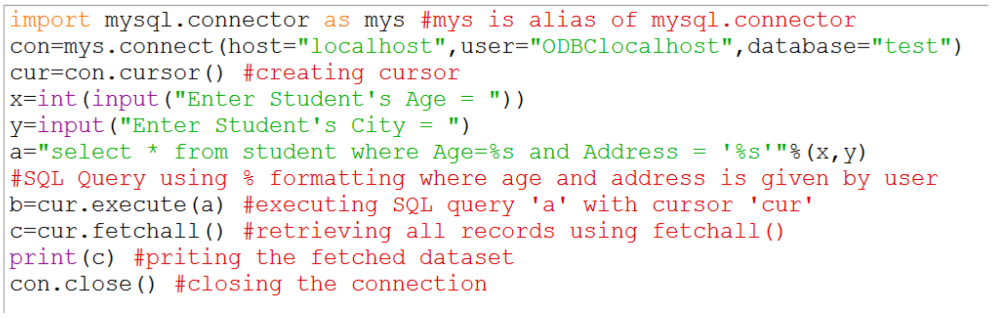
6. Close the connection
After doing all the processing, connection should be closed.
Syntax: <connection object>.close()
Example: Refer to the example of fetchmany()

Format specifier:

We need format specifier to write SQL query based on user input. For doing this we have two ways:

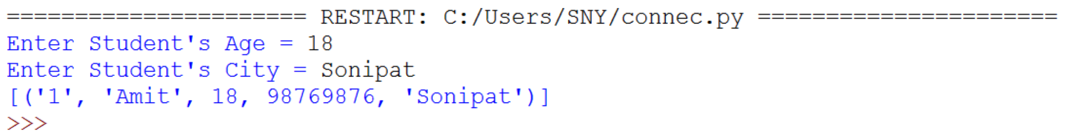
1. **Using % formatting:**
Whenever we needs to complete SQL query based on user input, we write a placeholder %s on that place.
Example: Let's fetch all the data from student table where age is 'x' and city is 'y'. (Here, value of x and y will be given by the user during run time).

Python Code:



```python
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
x=int(input("Enter Student's Age = "))
y=input("Enter Student's City = ")
a="select * from student where Age=%s and Address = '%s'"%(x,y)
#SQL Query using % formatting where age and address is given by user
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
c=cur.fetchall() #retrieving all records using fetchall()
print(c) #priting the fetched dataset
con.close() #closing the connection
```

Output:



```
====================== RESTART: C:/Users/SNY/connec.py ======================
Enter Student's Age = 18
Enter Student's City = Sonipat
[('1', 'Amit', 18, 98769876, 'Sonipat')]
>>>
```

2. **Using format() function:**
Whenever we needs to complete SQL query based on user input, we write a placeholder {} on that place. If we needs to complete the SQL query based on multiple user input, we

write placeholder {}, {}, {} and so on at those places and pass user defined values in format function in sequence. Here 1st values passed in format function will be passed to 1st {}, 2nd values passed to 2nd {}, 3rd value passed to 3rd {} and so on.
Example: Let's fetch all the data from student table where age is 'x' and city is 'y'. (Here, value of x and y will be given by the user during run time).

Python Code:

```
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
x=int(input("Enter Student's Age = "))
y=input("Enter Student's City = ")
a="select * from student where Age={} and Address='{}'".format(x,y)
#SQL Query using format() function where age and address is given by user
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
c=cur.fetchall() #retrieving all records using fetchall()
print(c) #priting the fetched dataset
con.close() #closing the connection
```

Output:

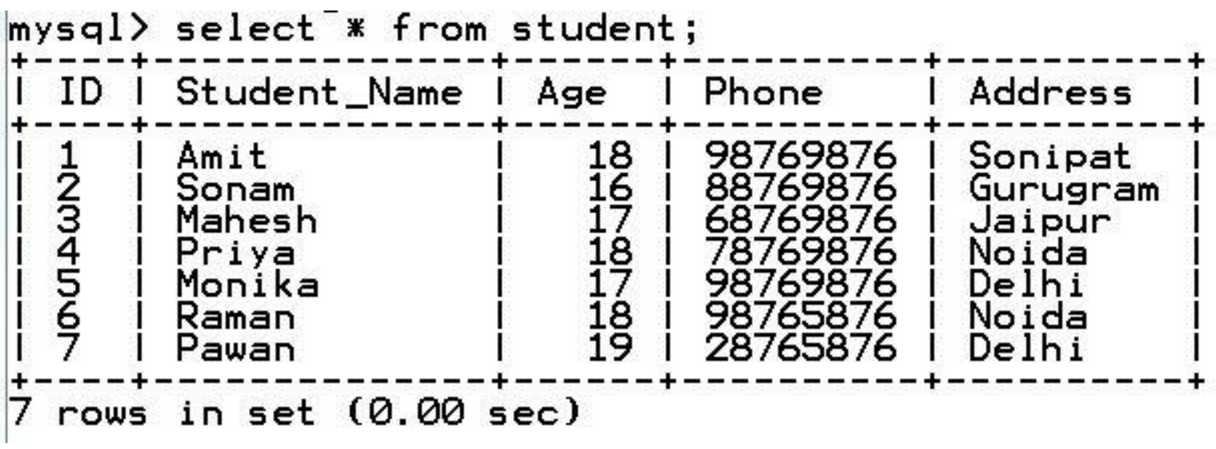
```
Enter Student's Age = 18
Enter Student's City = Sonipat
[('1', 'Amit', 18, 98769876, 'Sonipat')]
>>>
```

Commit() : Whenever we perform update, delete or insert query, commit() function must be run before closing the connection.

Syntax: <connection object>.commit()

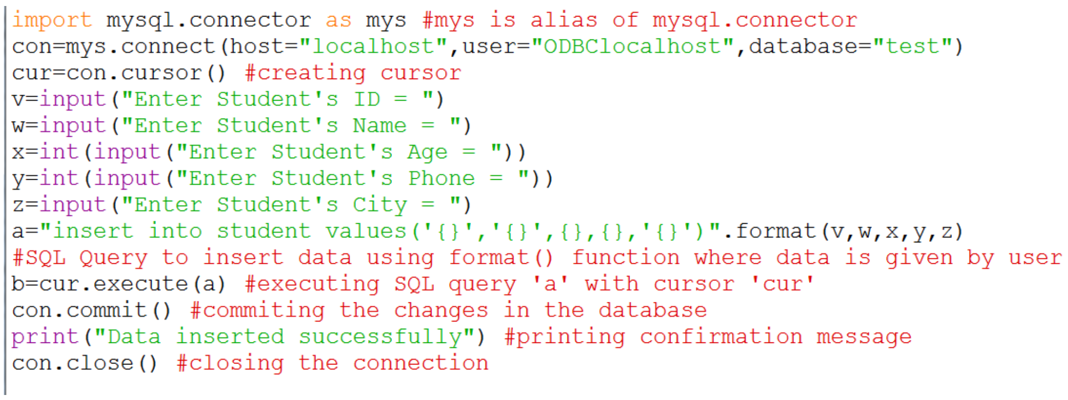
Example 1: Insert data in student table which will be given by the user during run time.

Data in student table before insertion



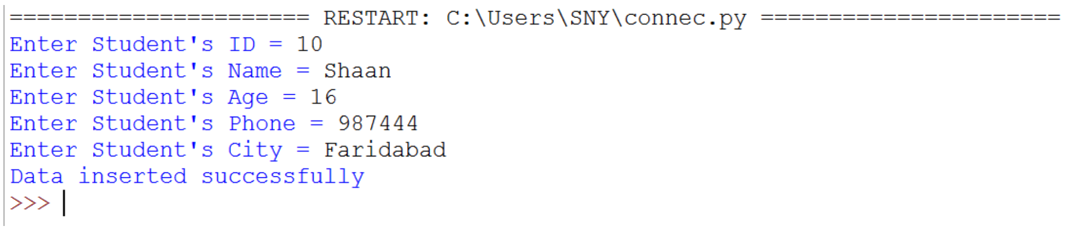
```
mysql> select * from student;
+----+----------------+-------+------------+-----------+
| ID | Student_Name   | Age   | Phone      | Address   |
+----+----------------+-------+------------+-----------+
| 1  | Amit           |    18 | 98769876   | Sonipat   |
| 2  | Sonam          |    16 | 88769876   | Gurugram  |
| 3  | Mahesh         |    17 | 68769876   | Jaipur    |
| 4  | Priya          |    18 | 78769876   | Noida     |
| 5  | Monika         |    17 | 98769876   | Delhi     |
| 6  | Raman          |    18 | 98765876   | Noida     |
| 7  | Pawan          |    19 | 28765876   | Delhi     |
+----+----------------+-------+------------+-----------+
7 rows in set (0.00 sec)
```

Python code:



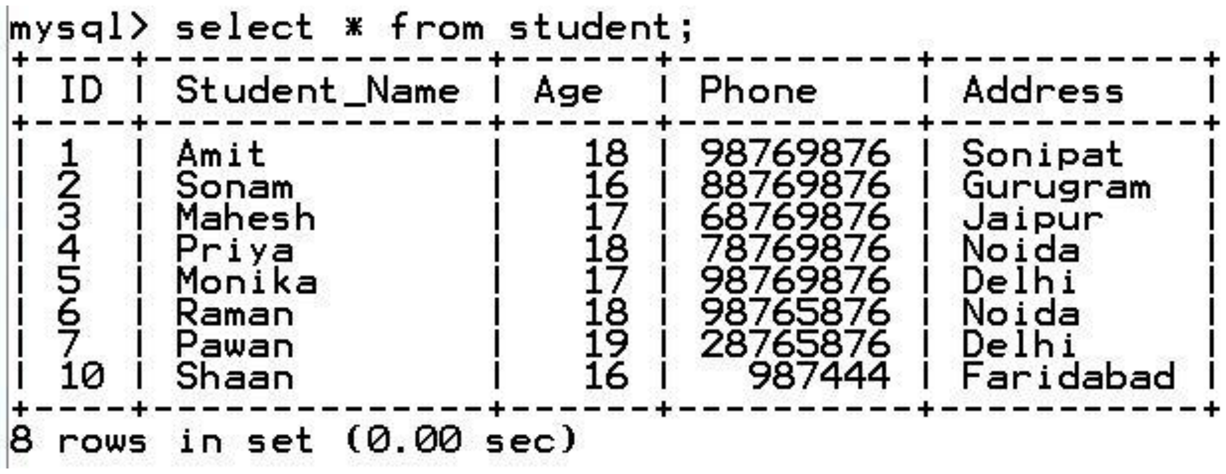
```
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
v=input("Enter Student's ID = ")
w=input("Enter Student's Name = ")
x=int(input("Enter Student's Age = "))
y=int(input("Enter Student's Phone = "))
z=input("Enter Student's City = ")
a="insert into student values('{}','{}',{},{},'{}')".format(v,w,x,y,z)
#SQL Query to insert data using format() function where data is given by user
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
con.commit() #commiting the changes in the database
print("Data inserted successfully") #printing confirmation message
con.close() #closing the connection
```

Output:



```
====================== RESTART: C:\Users\SNY\connec.py ======================
Enter Student's ID = 10
Enter Student's Name = Shaan
Enter Student's Age = 16
Enter Student's Phone = 987444
Enter Student's City = Faridabad
Data inserted successfully
>>> |
```

Data in student table after insertion

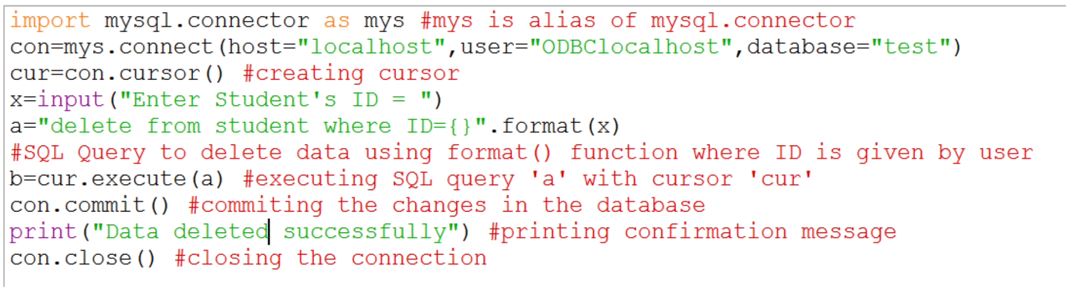


```
mysql> select * from student;
+----+----------------+------+-----------+-----------+
| ID | Student_Name   | Age  | Phone     | Address   |
+----+----------------+------+-----------+-----------+
| 1  | Amit           |   18 | 98769876  | Sonipat   |
| 2  | Sonam          |   16 | 88769876  | Gurugram  |
| 3  | Mahesh         |   17 | 68769876  | Jaipur    |
| 4  | Priya          |   18 | 78769876  | Noida     |
| 5  | Monika         |   17 | 98769876  | Delhi     |
| 6  | Raman          |   18 | 98765876  | Noida     |
| 7  | Pawan          |   19 | 28765876  | Delhi     |
| 10 | Shaan          |   16 |   987444  | Faridabad |
+----+----------------+------+-----------+-----------+
8 rows in set (0.00 sec)
```

Example 2: Delete data of those students whose ID is given by the user during run time in student table.

In the previous example, after inserting a record in student table, we have 8 records in the student table. (Kindly refer previous example for current state of student table).

Python code:

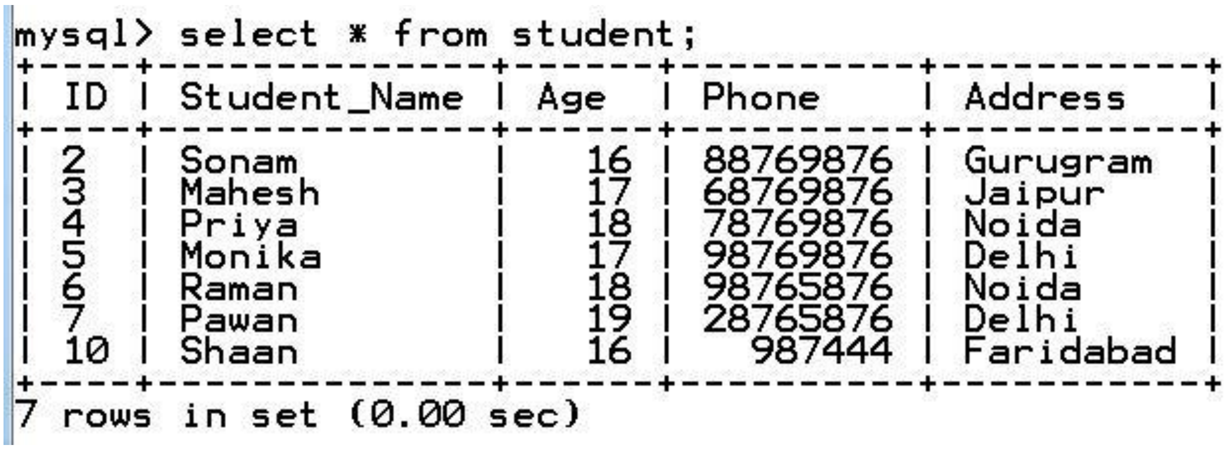


```
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
x=input("Enter Student's ID = ")
a="delete from student where ID={}".format(x)
#SQL Query to delete data using format() function where ID is given by user
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
con.commit() #commiting the changes in the database
print("Data deleted successfully") #printing confirmation message
con.close() #closing the connection
```

Output:

```
>>>
===================== RESTART: C:/Users/SNY/delete.py =====================
Enter Student's ID = 1
Data deleted successfully
>>>
```

Data in student table after deletion



```
mysql> select * from student;
+----+--------------+------+----------+-----------+
| ID | Student_Name | Age  | Phone    | Address   |
+----+--------------+------+----------+-----------+
| 2  | Sonam        |   16 | 88769876 | Gurugram  |
| 3  | Mahesh       |   17 | 68769876 | Jaipur    |
| 4  | Priya        |   18 | 78769876 | Noida     |
| 5  | Monika       |   17 | 98769876 | Delhi     |
| 6  | Raman        |   18 | 98765876 | Noida     |
| 7  | Pawan        |   19 | 28765876 | Delhi     |
| 10 | Shaan        |   16 |   987444 | Faridabad |
+----+--------------+------+----------+-----------+
7 rows in set (0.00 sec)
```

Example 3: Update name of those students whose name and ID is given by the user during run time in student table.

In the previous example, after deleting a record in student table, we have 7 records in the student table. (Kindly refer previous example for current state of student table).

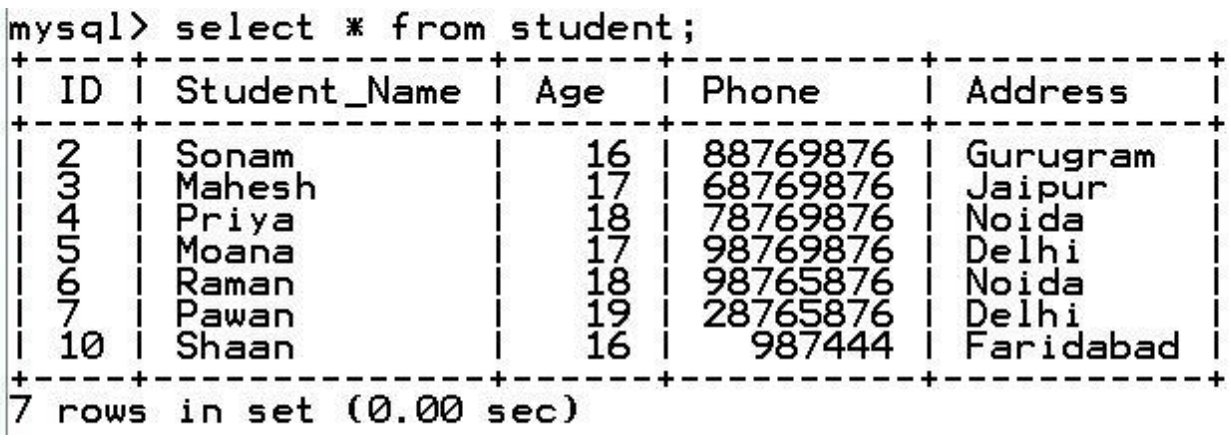
Python code:

```python
import mysql.connector as mys #mys is alias of mysql.connector
con=mys.connect(host="localhost",user="ODBClocalhost",database="test")
cur=con.cursor() #creating cursor
x=input("Enter Student's ID = ")
y=input("Enter Student's Name = ")
a="update student set Student_Name='{}' where ID='{}'".format(y,x)
#SQL Query to update student's name using format() function
#where ID is given by user
b=cur.execute(a) #executing SQL query 'a' with cursor 'cur'
con.commit() #commiting the changes in the database
print("Data updated successfully") #printing confirmation message
con.close() #closing the connection
```

Output:

```
===================== RESTART: C:/Users/SNY/update.py =====================
Enter Student's ID = 5
Enter Student's Name = Moana
Data updated successfully
>>> |
```

Data in student table after updating



```
mysql> select * from student;
+----+--------------+------+----------+-----------+
| ID | Student_Name | Age  | Phone    | Address   |
+----+--------------+------+----------+-----------+
| 2  | Sonam        |   16 | 88769876 | Gurugram  |
| 3  | Mahesh       |   17 | 68769876 | Jaipur    |
| 4  | Priya        |   18 | 78769876 | Noida     |
| 5  | Moana        |   17 | 98769876 | Delhi     |
| 6  | Raman        |   18 | 98765876 | Noida     |
| 7  | Pawan        |   19 | 28765876 | Delhi     |
| 10 | Shaan        |   16 |   987444 | Faridabad |
+----+--------------+------+----------+-----------+
7 rows in set (0.00 sec)
```

MULTIPLE CHOICE QUESTIONS (MCQ)

1. What is the use of DBMS?
  a. Data Retrieval
  b. Data Storage
  c. Data Manipulation
  d. All of the above

2. What is relational data model?
  a. Where data is organized in tree like structure with parent child relationship
  b. Where pointer is used to navigate through data
  c. Where data is represented as objects.
  d. Where data is organized into tables with rows and columns

3. What is tuple is relation?
  a. row
  b. column
  c. field
  d. attribute

4. What is attribute in relation?
  a. row
  b. column
  c. tuple
  d. key

5. What is degree in database?
  a. Foreign key of the table
  b. Primary key of the table
  c. Number of columns in the table
  d. Number of rows in the table

6. What is domain in database?
  a. The alternate key of a table
  b. The primary key of a table
  c. The set of all possible values in column
  d. The number of rows in table

7. What is cardinality in database?
  a. Number of columns
  b. Number of rows
  c. Number of primary key
  d. Number of attribute key

8. What is key in SQL?
   a. Which is used to uniquely identify record
   b. To maintain data integrity
   c. To enable efficient data retrieval
   d. All of the above

9. Which key is used to uniquely identify record in table and doesn’t allow NULL values?
   a. Alternate key
   b. Primary key
   c. Super key
   d. Foreign key

10. A column that could potentially be used as the primary key and it should also contain unique values?
   a. Alternate key
   b. Primary key
   c. Candidate key
   d. Surrogate key

11. A candidate key that is not selected as primary key which can serve as a unique identifier.
   a. Alternate key
   b. Primary key
   c. Candidate key
   d. Surrogate key

12. What is the use of foreign key?
   a. To uniquely identify record
   b. To enforce data integrity
   c. To define the data type of column
   d. To establish a link between two tables

13. A column or set of column in table that refers to the primary key of another table.
   a. Alternate key
   b. Primary key
   c. Candidate key
   d. Foreign key

14. Which of the following SQL commands is used to view list of all database?
   a. select databases
   b. show databases
   c. view databases
   d. project databases

15. Which of the following SQL commands is used to use/select a particular database?

a. use
b. select
c. view
d. project

16. Which SQL command is used to define and maintain physical structure or schema of table in database like creating, altering and deleting database object such as table and constraints?
a. DDL
b. DML
c. DCL
d. TCL

17. Which SQL command used to change or modify any attribute/column like addition and deletion of column in database?
a. create
b. alter
c. delete
d. update

18. A user request to retrieve data or information from database/table is called?
a. key
b. data
c. query
d. view

19. Which of the following is not a valid datatype in SQL?
a. Date
b. String
c. Decimal
d. Char

20. Which command is used to delete database permanently?
a. create database
b. delete database
c. drop database
d. use database

21. Which commands is used to show all table in current using database?
a. display tables;
b. show tables;
c. view tables;
d. select all tables;

22. Which command is used in where clause to search NULL values in a particular column?

a. IS NULL
b. IN NULL
c. NOT NULL
d. IS NOT NULL

23. Wild card operator (%,_) are used with?

a. count
b. max
c. like
d. min

24. Which SQL function is used to determine the no. of row or non-null values?

a. min
b. max
c. count
d. sum

25. Which SQL function is used to count the entire number of row in database table?

a. count
b. count(*)
c. max
d. min

26. In relational database, a key that established a link between two tables?

a. Primary key
b. Alternate key
c. Foreign key
d. Candidate key

27. A table containing data organized in the form of row and column in relational data model?

a. Relation
b. View
c. Tuple
d. Database

28. Which SQL clause is used to filter the result of SELECT statement in Relational Database?

a. from
b. where
c. join
d. having

29. INSERT command is used in database to perform

a. To retrieve a set of statement
b. To add new row of data to a table
c. To change already existing data in table
d. To delete any existing row in table

30. How many candidate key can a relation/table have in relational database
a. One
b. Two
c. Multiple
d. None

31. What is the main difference between candidate key and primary key?
a. A candidate key can contain NULL value but primary key can't contain NULL value
b. A primary key can contain unique value but it is not necessary for the candidate key to have unique value
c. Candidate key is chosen by database system while primary key is chosen by the designer
d. No difference

32. Which DDL command is used to modify the structure of an existing table?
a. create table
b. modify table
c. alter table
d. update table

33. Which DDL command is used to remove a table along with its all content from a database?
a. DELETE TABLE
b. DROP TABLE
c. REMOVE TABLE
d. ERASE TABLE

34. Which constraint enforce data integrity by ensuring that a column always contain a values?
a. NULL
b. NOT NULL
c. CHECK
d. DEFAULT

35. What is primary difference between candidate key and primary key in database?
a. Primary key is chosen by the end user/designer while candidate key is generated by database system
b. A candidate key uniquely identifies each row in a table, while primary key is not unique
c. A primary key can have NULL values while a candidate key can't

d. A candidate key can become a primary key, but a primary key can’t become a candidate key

36. Which key is used to enforce referential integrity between tables in database system

a. Primary key
b. Candidate key
c. Foreign key
d. Alternate key

37. What is the main difference between CHAR and VARCHAR datatype in SQL?

a. CHAR is case-sensitive while VARCHAR is non case sensitive
b. CHAR store variable length strings while VARCHAR stores fixed length string
c. CHAR store fixed length strings while VARCHAR stores variable length string
d. CHAR is used storing numeric data while VARCHAR is used for text data

38. What is the primary purpose of unique key constraints in a relational databases?

a. To ensure that values in a column are NULL
b. To ensure that values in a column are unique
c. To define relationship between tables
d. To specify a condition that must be met for data to be valid in a column

39. Which constraint key is used when you want to enforce uniqueness but allows NULL values in database

a. PRIMARY KEY
b. UNIQUE KEY
c. NOT NULL
d. CHECK

40. Which of the following is not a valid DML command in SQL?

a. INSERT
b. UPDATE
c. ALTER
d. DELETE

41. Which keyword is used for table aliasing that involves giving a table short and alternative name to simplify query syntax?

a. from
b. as
c. where
d. on

42. Which SQL clause is used in database table to eliminate duplicate rows from the query result?

a. group by

b. distinct
c. describe
d. duplicate

43. Which of the following clauses in SQL is most appropriate to use to select matching tuples in a specific range of values?
a. IN
b. LIKE
c. BETWEEN
d. IS

44. Which of the following SQL datatype allows NULL values by default?
a. INT
b. CHAR
c. VARCHAR
d. FLOAT

45. Which of the following is NOT a required argument to the connect() function?
a. Hostname
b. Username
c. Password
d. Database name

46. Which method is used to execute a SQL query on a MySQL database?
a. execute()
b. query()
c. run()
d. None of the above

47. Which method is used to fetch n number of results from a SQL query?
a. fetchall()
b. fetchone()
c. fetchmany()
d. All of the above

48. _______ it is a pointer or iterator which points towards the resultset of the SQL query.
a. cursor
b. rset
c. temp
d. None of these

49. Which of the following is not a valid method to fetch records from database in python.
a. fetchmany()
b. fetchone()

c. fetchmulti()
d. fetchall()

50. To get all the records from result set, you may use ___________.
a. cursor.fetchmany()
b. cursor.fetchall()
c. cursor.fetchone()
d. cursor.execute()

Short answer type Questions:

1. What is SQL?
2. Differentiate between DML and DDL. Explain with the help of examples.
3. What are the different type of SQL data type?
4. What is the difference between where statement and having statement in SQL?
5. What is primary key?
6. Differentiate between a database table and a database record.
7. What is foreign Key? And how does it relate in a database.
8. What is Domain in database?
9. Explain the role of clause in query of SQL commands.
10. What is Aliasing?
11. Explain char, varchar and int datatype in SQL with example.
12. Differentiate between candidate key and alternate key?
13. Differentiate between degree and cardinality of a table with the help of example.
14. What are database keys? Explain all the database keys.
15. What are constraints in SQL?
16. Explain the wildcards used with like operator.
17. What are joins in SQL?
18. Differentiate between Natural Join and Equi Join.
19. Explain any two aggregate function in SQL.
20. What is a cursor?
21. What is the role of commit() function in SQL?
22. What is Cartesian product? What will be the number of rows in output table if we apply Cartesian product on two tables T1 and T2 with 9 and 10 rows respectively?
23. Explain alter table command with the help of example.
24. Differentiate between fetchall() function and fetchmany() function in SQL.
25. In SQL, Define aggregate function and write the name of the aggregate function which will display the cardinality of a table.

Assertion-and-Reason Type

In the following questions, Assertion (A) and Reason (R).
Choose the correct choice as:

(a) Both Assertion (A) and Reason (R) are the true and Reason (R) is a correct explanation of Assertion (A).
(b) Both Assertion (A) and Reason (R) are the true but Reason (R) is not a correct explanation of Assertion (A).
(c) Assertion (A) is true and Reason (R) is false.
(d) Assertion (A) is false and Reason (R) is true.

1. **Assertion (A):** COUNT function can be work with distinct keyword.
   Reason (R): DISTINCT keyword can only be used with COUNT function.

2. **Assertion (A):** HAVING clause can only be used with GROUP BY statement.
   Reason (R): WHERE clause can be used in place of HAVING clause in GROUP BY statement.

3. **Assertion (A):** LIKE operator is used for pattern matching in WHERE clause.
   Reason (R): % and _ wildcard is used in LIKE operator for making a pattern.

4. **Assertion (A):** The primary key is applied on a column of a table.
   Reason (R): NOT NULL constraint restricts NULL values in a column.

5. **Assertion (A):** SUM and COUNT are aggregate functions.
   Reason (R): Aggregate functions works on multiple tuples.

6. **Assertion (A):** Primary key constraints allows NULL values.
   Reason (R): The primary key constraints ensures that a column can contain unique Values for each row.

7. **Assertion (A):** Unique key constraint allows NULL values.
   Reason (R): The unique key constraints ensures that a column can contain unique Values.

8. **Assertion (A):** The HAVING clause is used to filter aggregated data in SQL queries.
   Reason (R): The HAVING clause is used to group rows with similar values in one or more column into result sets.

9. **Assertion (A):** Inner Join retrieves rows that have matching values in both tables being joined.
   Reason (R): Inner join excludes row with no matching values

10. **Assertion (A):** Between operator is used to filter data within a specified range.
    Reason (R): Where clause works exactly same as between operator

Long Answer Type Questions:

1. Write MySQL command to create the table 'Employee' with the following structure and constraint. Table: Employee

Column_Name	**DataType(size)**	**Constraint**
Emp_ID	Int(20)	Primary key
Emp_Name	char(100)	Not Null
Salary	int(20)	Not Null
Department	char(30)	
Age	int(15)	Not Null
Address	Varchar(200)	Unique

2. Write MySQL command to create the table 'Student' and 'Activities' with the following structure and constraint.

Table: Student

Column_Name	**DataType(size)**	**Constraint**
Student_ID	varchar(20)	Primary key
Student_Name	char(80)	Not Null
Gender	char(20)	Not Null
Class	varchar(30)	
Age	int(20)	Not Null
Address	Varchar(150)	Unique
Phone	Int(15)	Not Null, unique

Table: Activities

Column_Name	**DataType(size)**	**Constraint**
Student_ID	varchar(20)	Foreign key references to Student_ID of Employee table
Activity_Name	char(80)	Not Null
Position	char(30)	Not Null

3. Anmol maintain that database of medicines for his pharmacy using SQL to store the data. The structure of the table PHARMA for the purpose is as follows:

- Name of the table-PHARMA
- The attributes of PHARMA are as follows:

MID - numeric
MNAME - character of size 20
PRICE - numeric
UNITS - numeric
EXPIRY – date

Table: PHARMA

MID	**MNAME**	**PRICE**	**UNITS**	**EXPIRY**
M1	PARACETAMOL	12	120	2022-12-25
M2	CETRIZINE	6	125	2022-10-12
M3	METFORMIN	14	150	2022-05-23
M4	VITAMIN B-6	12	120	2022-07-01
M5	VITAMIN D3	25	150	2022-06-30
M6	TELMISARTAN	22	115	2022-02-25

(a) Write the degree and cardinality of the table PHARMA.

(b) Identify the attribute best suitable to be declared as a primary key.

(c) Anmol has received a new medicine to be added into his stock, but for which he does not know the number of UNITS. So he decides to add the medicine without its value for UNITS. The rest of the values are as follows:

MID	MNAME	PRICE	EXPIRY
M7	SUCRALFATE	17	2022-03-20

Write the SQL command which Anmol should execute to perform the required task.

(d) Anmol wants to change the name of the attribute UNITS to QUANTITY in the table PHARMA. Which of the following commands will he use for the purpose?

I. UPDATE
II. DROP TABLE
III. CREATE TABLE
IV. ALTER TABLE

(e) Now Anmol wants to increase the PRICE of all medicines following commands will he use for the purpose?

I. UPDATE SET
II. INCREASE BY
III. ALTER TABLE
IV. INSERT INTO

4. For the following SQL Table named PASSENGERS in a database TRAVEL:

TNO	NAME	START	END
T1	RAVI KUMAR	DELHI	MUMBAI
T2	NISHANT JAIN	DELHI	KOLKATA
T3	DEEPAK PRAKASH	MUMBAI	PUNE

A cursor named Cur is created in Python for a connection of a host which contains the database TRAVEL. Write the output for the execution of the following Python statements

```
for the above SQL Table PASSENGERS:
Cur.execute('USE TRAVEL')
Cur.execute('SELECT * FROM PASSENGERS')
Recs=Cur.fetchall()
For R in Recs:
    Print(R[1])
```

5. Write SQL statements for the following queries (i) to (v) based on the relations CUSTOMER and TRANSACTION given below:

Table: CUSTOMER

ACNO	NAME	GENDER	BALANCE
C1	RISHABH	M	15000
C2	AAKASH	M	12500
C3	INDIRA	F	9750
C4	TUSHAR	M	14600
C5	ANKITA	F	22000

Table: TRANSACTION

ACNO	TDATE	AMOUNT	TYPE
C1	2020-07-21	1000	DEBIT
C5	2019-12-31	1500	CREDIT
C3	2020-01-01	2000	CREDIT

1. To display all information about the CUSTOMERS whose NAME starts with 'A'.
2. To display the NAME and BALANCE of Female CUSTOMERS (with GENDER as 'F') whose TRANSACTION Date (TDATE) is in the year 2019.
3. To display the total number of CUSTOMERS for each GENDER.
4. To display the CUSTOMER NAME and BALANCE in ascending order of GENDER.
5. To display CUSTOMER NAME and their respective INTEREST for all CUSTOMERS where INTEREST is calculated as 8% of BALANCE.

6. The IT Company XYZ has asked their IT manager Ms. Preeti to maintain the data of all the employees in two tables EMPLOYEE and DEPT. Ms. Preeti has created two tables EMPLOYEE and DEPT. She entered 6 rows in EMPLOYEE table and 5 rows in DEPT table.

Table: DEPT

D_CODE	**D_NAME**	**CITY**
D001	INFRASTRUCTURE	DELHI
D002	MARKETING	DELHI
D003	MEDIA	MUMBAI
D005	FINANCE	KOLKATA
D004	HUMAN RESOURCE	MUMBAI

Table: EMPLOYEE

E_NO	NAME	DOJ	DOB	GENDER	D_CODE	Salary
1001	Vinay	2013-09-02	1991-09-01	MALE	D001	250000
1002	Ruby	2012-12-11	1990-12-15	FEMALE	D003	270000
1003	Anuj	2013-02-03	1987-09-04	MALE	D005	240000
1007	Sunny	2014-01-17	1988-10-19	MALE	D004	250000
1004	Rohit	2012-12-09	1986-11-14	MALE	D001	270000
1005	Preeti	2013-11-18	1989-03-31	FEMALE	D002	NULL

Note: DOJ refers to date of joining and DOB refers to date of Birth of employees.
Based on the above data, answer the following questions:

1. Identify the column which can be consider as primary key in EMPLOYEE table.
2. Identify the column which can be consider as primary key in DEPT table
3. What is the degree and cardinality of EMPLOYEE table?
4. What is the degree and cardinality of DEPT table?
5. **Write SQL queries for the following:**
   1) Insert two new row in Employee table with following data:

1006	Rahul	2019-11-06	1992-01-04	MALE	D003	156000
1008	Sonam	2022-01-06	1991-04-06	FEMALE	D005	167000

   2) To display E_NO, NAME, GENDER from the table EMPLOYEE in descending order of E_NO.
   3) To display the NAME of all the 'FEMALE' employees from the table EMPLOYEE.
   4) To display the E_NO and NAME of those employees from the table EMPLOYEE who are born between '1987-01-01' and '1991-12-01'.
   5) To display NAME and CITY of those employees whose DEPARTMENT is either 'MEDIA' or 'FINANCE'.
   6) To display the NAME of those employees whose name starts with letter 'R'.
   7) To display NAME of those employees whose name contains letter 'n'.
   8) To display NAME of those employees whose name has exact 5 letters.
   9) To display D_NAME and CITY from table DEPT where D_NAME ends with letter 'G' and CITY is 'DELHI'.
   10) To display the maximum SALARY of EMPLOYEE table.
   11) To delete data of all those employees whose age is less than 25.
   12) To update SALARY to 230000 of those employee whose E_NO is 1004.
   13) To change the sequence of DOB column in employee table and move it before DOJ column.
   14) To add a new column MOBILE int(20) before column SALARY in employee table.
   15) To set SALARY to 300000 of all those employees whose age is NULL.
   16) To Increase the salary of all employees by 30000 in EMPLOYEE table.
   17) To display the average SALARY of EMPLOYEE table.
   18) To display name of employees who have SALARY more than 200000 in ascending order of NAME.
   19) To display department wise average salary of employees.
   20) To display total number of departments in XYZ company.
   21) To delete data of all the employees whose D_CODE is not 'D001'.
   22) To display E_NO, NAME and SALARY of all those employees who don't live in 'DELHI'.

23) To change column name CITY to D_CITY in DEPT table.
24) To delete EMPLOYEE table.
25) To delete D_NAME column from DEPT table.

7. A garment store is considering to maintain their inventory using SQL to store the data. as a database administrator, Mr.Rohit has decided that:

- Name of the database – **STORE**
- Name of the table – **GARMENT**
- The attributes of GARMENT table are as follows:
  - **GCODE** – numeric
  - **DESCRIPTION** – character of size 50
  - **PRICE** – numeric
  - **FCODE** – varchar of size 10

Table: GARMENT

GCODE	DESCRIPTION	PRICE	FCODE
10023	JEANS	1150	F01
10001	SHIRT	750	F02
10044	SHORTS	600	F05
10005	TIE	400	F04
10002	JACKET	5000	F01
10022	SOCKS	150	NULL

1. Categorize the following commands as DDL or DML:
   ALTER, INSERT, UPDATE, CREATE, DROP, DELETE
2. Identify the attribute of Garment table to be declared as primary key.
3. Write the degree and cardinality of the table GARMENT
4. Write SQL query to create database STORE.
5. Write SQL query to display list of available databases.
6. Write SQL query to use database STORE.
7. Write SQL query to display list of available tables in database STORE.
8. Write SQL query to create table GARMENT with aforementioned attributes.
9. Obtain the output of following SQL queries based on the data given in table GARMENT:
   (i) SELECT MAX(PRICE), MIN(PRICE) FROM GARMENT;
   (ii) SELECT GCODE, DESCRIPTION FROM GARMENT;
   (iii) SELECT FCODE,GCODE FROM GARMENT WHERE PRICE BETWEEN 500 AND 800;
   (iv) SELECT * FROM GARMENT WHERE DESCRIPTION NOT IN ('JEANS','TIE');
   (v) SELECT GCODE FROM GARMENT WHERE DESCRIPTION LIKE '%S%';
   (vi) SELECT GCODE,PRICE FROM GARMENT WHERE DESCRIPTION LIKE '_ _ _';
   (vii) SELECT DISTINCT FCODE FROM GARMENT;

(viii) SELECT SUM(PRICE) FROM GARMENT;
(ix) SELECT * FROM GARMENT WHERE DESCRIPTION LIKE '%T%' AND FCODE!='F02';
(x) SELECT * FROM GARMENT ORDER BY PRICE DESC;
(xi) SELECT PRICE*10 FROM GARMENT;
(xii) SELECT COUNT(DISTINCT FCODE) FROM GARMENT;
(xiii) SELECT * FROM GARMENT WHERE FCODE NOT IN ('F01','F02') AND PRICE<500;
(xiv) SELECT GCODE, PRICE FROM GARMENT WHERE FCODE IS NULL;
(xv) SELECT * FROM GARMENT WHERE PRICE >500 AND PRICE <1000;

8. Write the output of the following SQL queries based on table TRANSACTION given below:
Table: TRANSACTION

T_NO	M_NO	AMOUNT	CARD_TYPE	DATE	STATUS
1	11	5000	CREDIT	2019-10-11	SUCCESS
2	11	170	CREDIT	2019-10-14	FAILURE
3	13	800	DEBIT	201-10-24	FAILURE
4	12	90	CREDIT	2019-11-10	SUCCESS
5	13	1400	DEBIT	2019-11-11	SUCCESS
6	11	500	DEBIT	2019-11-18	SUCCESS
7	13	1600	DEBIT	2019-11-27	FAILURE

Table: COMPANY

T_NO	QTY_ISSUED	COMPANY
1	15	SBI
3	50	ICICI
4	34	HDFC

(i) SELECT M_NO, MIN(AMOUNT) FROM TRANSACTION GROUP BY M_NO HAVING COUNT(*)>2;
(ii) SELECT T_NO,AMOUNT FROM TRANSACTION WHERE CARD_TYPE='CREDIT';
(iii) SELECT * FROM TRANSACTION WHERE M_NO>12 AND STATUS='FAILURE';
(iv) SELECT CARD_TYPE, SUM(AMOUNT) FROM TRANSACTION GROUP BY CARD_TYPE;
(v) SELECT T_NO, AMOUNT*10 AS PAYMENT FROM TRANSACTION WHERE STATUS IN ('SUCCESS');
(vi) SELECT COUNT(*) FROM TRANSACTION WHERE STATUS='SUCCESS';
(vii) SELECT T_NO, AMOUNT, CARD_TYPE FROM TRANSACTION WHERE T_NO>5;
(viii) SELECT DISTINCT M_NO FROM TRANSACTION;
(ix) SELECT T_NO, M_NO,CARD_TYPE, COMPANY FROM TRANSACTION T, COMPANY C WHERE T.T_NO=C.T_NO AND AMOUNT>1000;
(x) SELECT T.T_NO AS TRANS_NO FROM TRANSACTION T, COMPANY C WHERE T.T_NO=C.T_NO;

9. The code given below reads the records from the table employee and displays only those records of employee table who don't lives in city 'Delhi':

E_ID – varchar(50)

E_Name – char(50)

Salary – int(15)

City – char(20)

Note the following to establish connectivity between Python and MySQL:

- **Username** is root
- **Password** is 12345
- The table exists in a MySQL database named *company*.
- The table has four attributes (E_ID, E_Name, Salary, City).

Write the following statements to complete the code:

Statement 1- to import the desired library.
Statement 2- to create a cursor
Statement 3- to write a sql query that fetches records of all those employees who don't lives in delhi.
Statement 4- to execute the query
Statement 5- to print all the records fetched in statement 4

```
Import _________ as mysql                                #Statement 1
def print():
    a=mysql.connect (host="localhost", user "root",passwd=12345, database=
    "company")
    b=____________                                        #Statement 2
    query = _________________                             #Statement 3
    c = ___________________                               #Statement 4
    for z in c:
        _________                                         #Statement 5
```

10. The code given below deletes the records from the table employee which contains following record structure:

E_ID – varchar(50)

E_Name – char(50)

Salary – int(15)

City – char(20)

Note the following to establish connectivity between Python and MySQL:

- **Username** is root
- **Password** is 12345
- The table exists in a MySQL database named *company*.
- The table has four attributes (E_ID, E_Name, Salary, City).

Write the following statements to complete the code:

Statement 1- to import the desired library.
Statement 2- to connect to the database