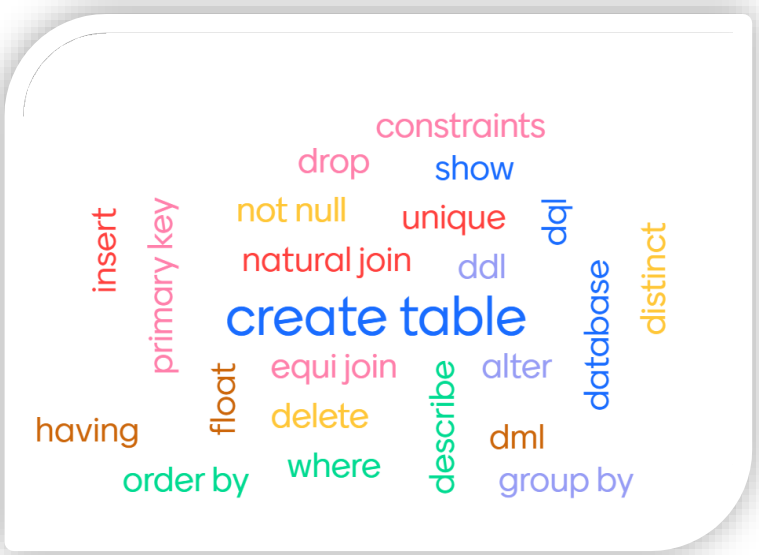
Database Management Systems



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 **column**s.

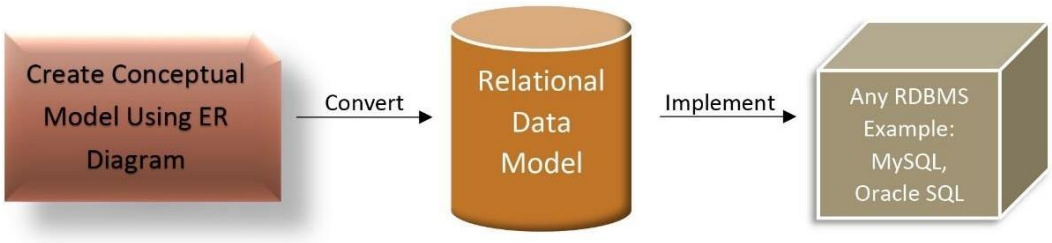


DIAGRAM: 2

ABOVE DIAGRAM SHOWS THAT HOW TO IMPLEMENT RELATIONAL DATA MODEL

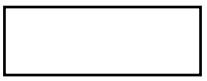
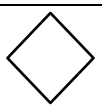
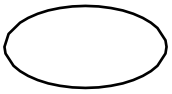
After creating conceptual data model using ER Diagram we need to convert it into Relational Data Model so that we can implement using any RDBMS language like MySQL, Oracle SQL.

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.

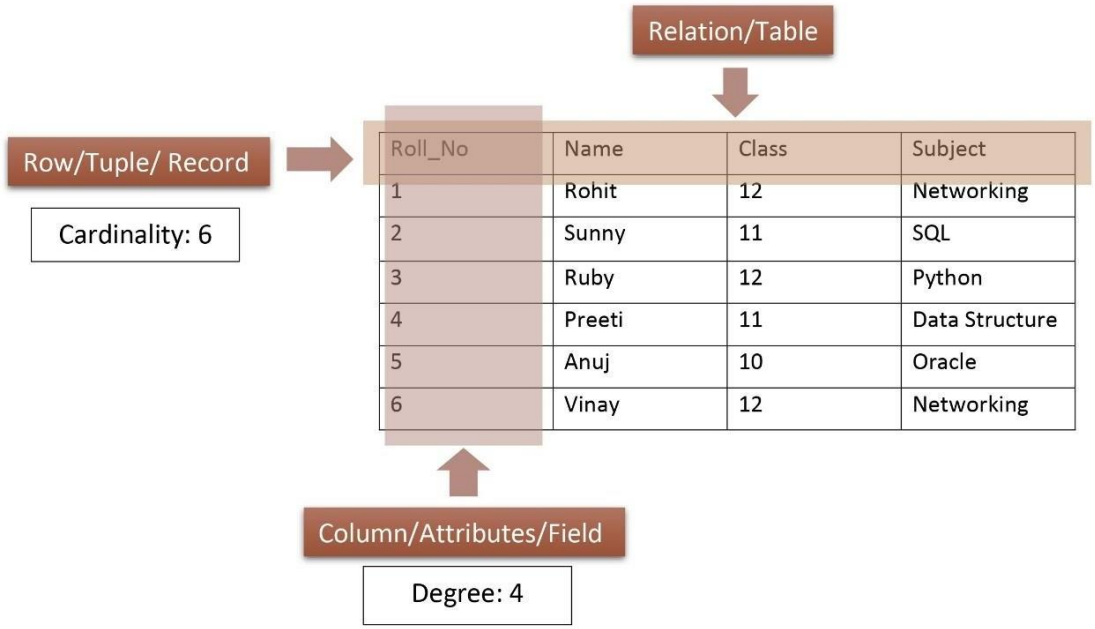


DIAGRAM: 3

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

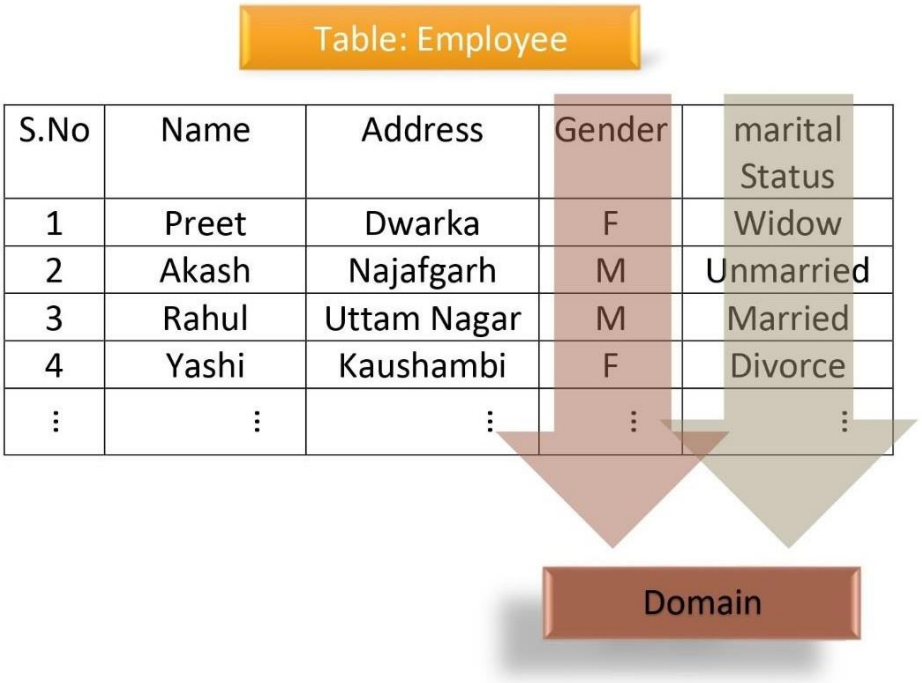


DIAGRAM: 4

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.

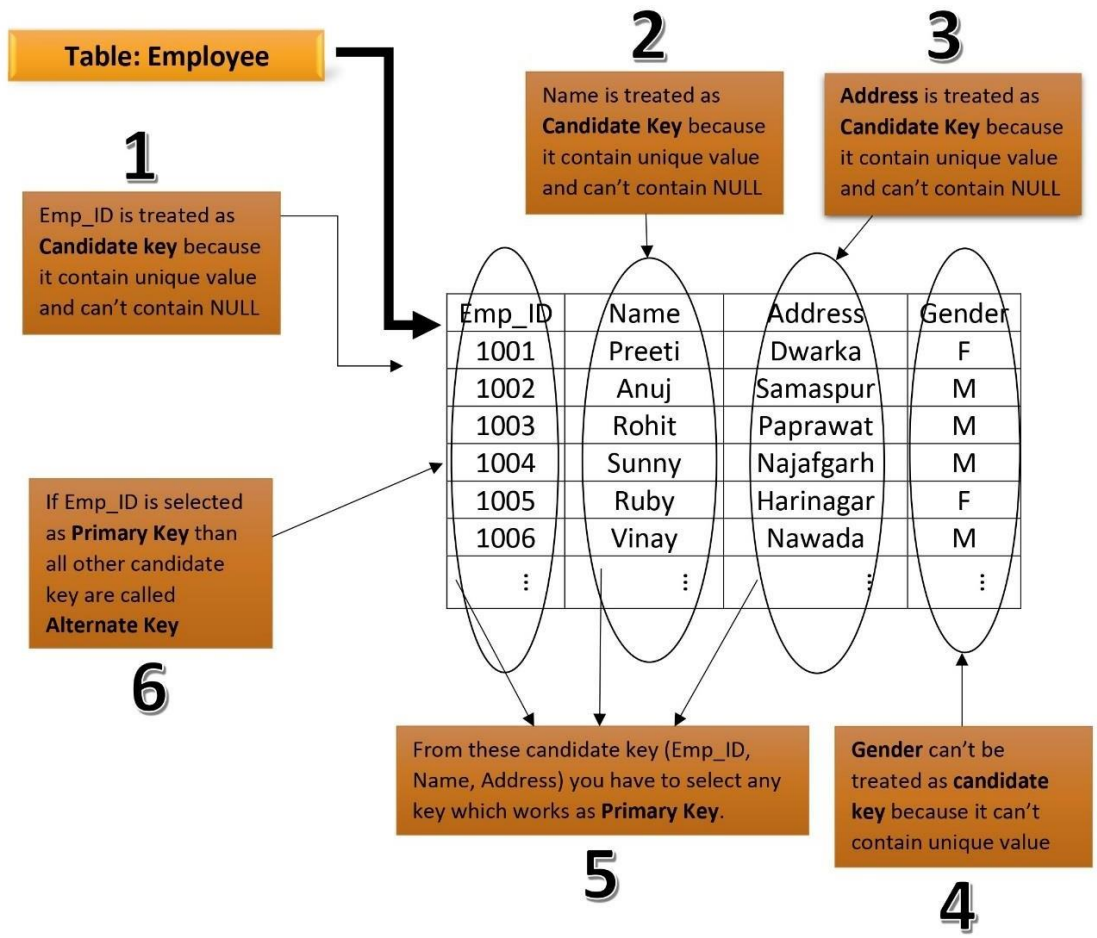


DIAGRAM: 5

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.

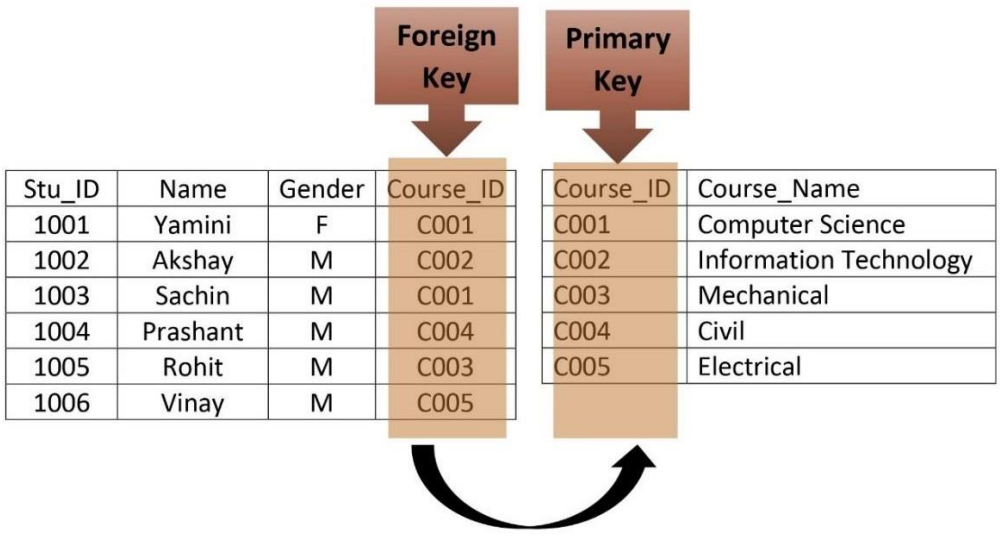


DIAGRAM: 6

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.

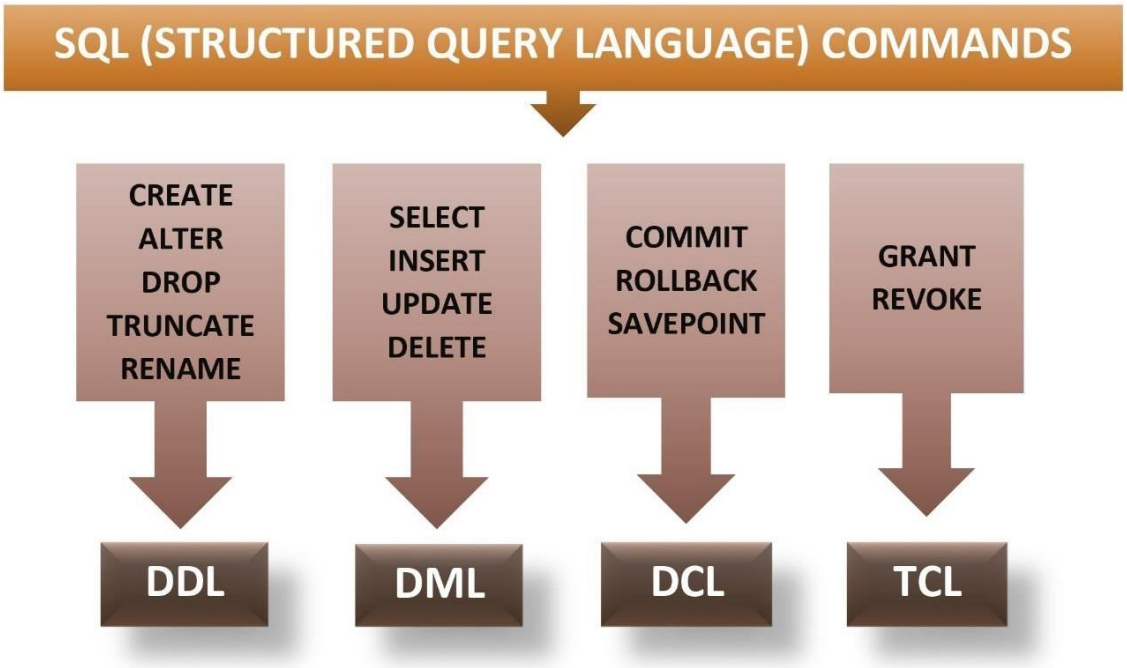


DIAGRAM: 8

SQL COMMANDS CAN BE CLASSIFIED INTO FOLLOWING CATEGORY

SQL perform following operation:

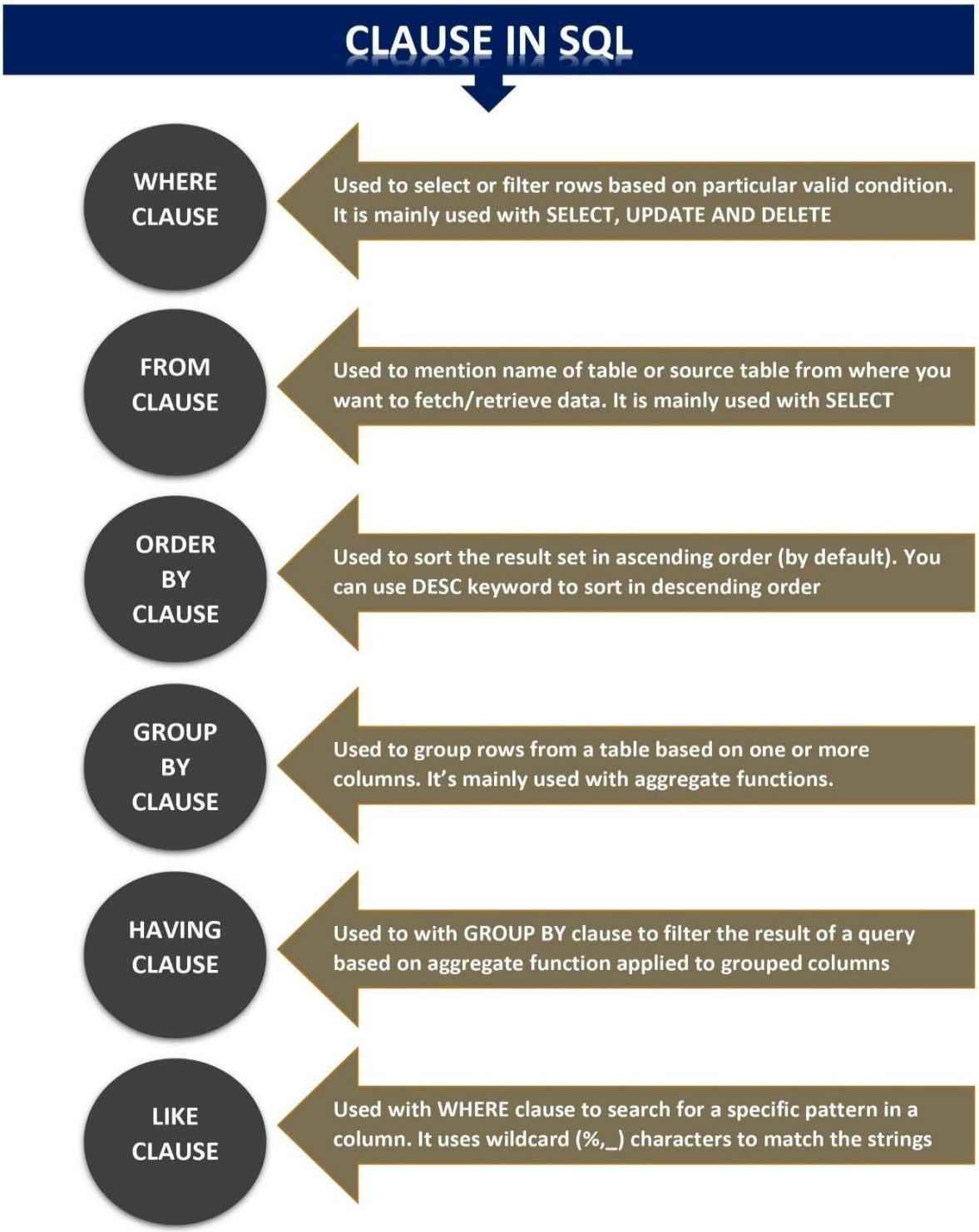
- Create a database
- Create a table
- Create view
- Insert data
- Update data
- Delete data
- Execute Query
- Set Permission or Constraints in table

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.



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. **Changing maximum size of the data in a column:** 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. **Changing order of the column:** 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. **Add/drop constraints/column:** We can add/drop constraints in a table using alter command.
   - **Adding primary key**: We are going to add primary key at column ID using alter command.

     Command: alter table <table name> add primary key(<column name>);

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

mysql> 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)
```

- **Dropping primary key:** We are going to remove primary key at column ID which we added in the previous section.
  Command: 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)
```

- **Adding a new column:** We are going to add a column country with data type char of size 50 to the table student using alter command.
  Command: 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)
```

- **Dropping a column:** We are going to remove a column ‘country’ which we added in the last section using alter command.
  Command: 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)
```

Update command: 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 | 9476985/ | 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.

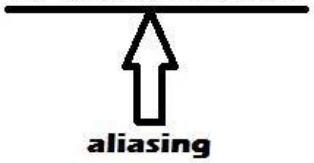
SQL Syntax:

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)
```