SQL(Structured Query Language)

02 August 2024 19:19

For database management system or DBMS There are special kind of languages called query language that can. be used to access and manipulate data from the database. The structured query language [SQL] Is the most popular query language used by major relational database management systems such as MySql, Oracle SQL-server, etc.

SQL is easy to learn as the statement comprise of descriptive English words and are not case sensitive. We can create and interact with the database using SQL easily. Benefit of using SQL is that we do not have to specify how to get the data from the database. Rather, we simply specify what is to be retrieved and SQL does the rest. Although called a query language, SQL can do much more besides Querying.

SQL provides statement for defining the structure of data, manipulating data in the database, declaring constraints and retrieving data from the database in various ways, depending on our requirement.

Data types and constraints in MySQL

We know that database consists of one or more relations and each relation(table) is made up of attributes(columns). Each attribute has a data type. We can also specify constraints for each attribute of a relation.

Data type of Attribute

Data type of an attribute indicates the type of data value that an attribute can have. It also decides the operations that can be performed on the data of that attribute. For example, arithmetic operations can be performed on numeric data, but not on character data. Commonly use data types in my SQL are numeric types, date and time types and string types(Varchar and char).

Data type	Description
CHAR(n)	Specifies character type data of length n where N could be any value from zero to 255. CHAR is of fixed length, means declaring CHAR(10) implies to reserve space for 10 characters. If data does not have 10 characters, MySQL fills the remaining six characters with spaces padded on the right.
VARCHAR(N)	Specifies character type data of length, where N could be any value from zero to 65,535. But unlike CHAR, VARCHAR(N) is a variable length data type. That is declaring varchar(30) Means a maximum of 30 characters can be stored, but the actual allocated bytes will depend on the length of entered string.
INT	Int specifies an integer value. Each int value occupies four bytes of storage. The range of unsigned values allowed in a 4 byte integer type, 0 to 4294967295. Four values larger than that we have to use BigInt which occupies eight bytes.
FLOAT	Holt's number with decimal points. Each float value occupies four bytes.
DATE	The DATE Type Is used for dates in "YYYY-MM-DD" format. YYYY is the 4 digit year, MM is the 2digit month and DD is 2 digit date. The support range is '**1000-01-01' to '9999-12-31'**

Constraints

Constraints are the certain types of restrictions on the data values that an attribute can have. They are used to ensure correctness of data. However, it is not mandatory to define constraints for each attribute of a table.

Constraints	Description
NOT NULL	Ensure that a column cannot have NULL value, where NULL means missing or unknown or not applicable value.

UNIQUE	Ensures that all the values of a column are distinct or unique.
DEFAULT	A default values specified for the column. If no value is provided.
PRIMARY KEY	The column, which can uniquely identify each row or record in a table.
FOREIGN KEY	The column, which refers to a value of an attribute defined as primary key in another table.

SQL for Data Definition

In order to be able to store data, we need to first define the relationship schema. Defining a schema includes creating a relation and giving name to a relation identifying the attributes in a relation, deciding upon the data type for each attribute, and also specify the constraints as per requirements. Sometimes we may require to make changes to the relation schema also. SQL allows us to write statements for defining, modifying and deleting relation schemas. These are parts of data definition language, or DDL.

Create Database

CREATE DATABASE databasename;

Add DBMS can manage multiple databases on one computer. Therefore, we need to select the database that we want to use. To know the names of existing database we use **the statement SHOW DATABASES.** From the listed databases, we can select the database to be used. Once the database is selected, we can proceed with creating tables or querying data.

```
mysql> show databases;
+-----------------------+
| Database              |
+-----------------------+
| carshowroom           |
| information_schema    |
| menagerie             |
| mydb                  |
| mysql                 |
| performance_schema    |
| sakila                |
| student_attendence    |
| studentattendancesbgs |
| sys                   |
| world                 |
+-----------------------+
11 rows in set (0.01 sec)

mysql> use student_attendence;
Database changed
mysql> show tables;
+-------------------------------+
| Tables_in_student_attendence  |
+-------------------------------+
| cost                          |
| dance                         |
| employee                      |
| guardian                      |
| music                         |
| student                       |
| t                             |
| t1                            |
| t2                            |
| uniform                       |
+-------------------------------+
10 rows in set (0.01 sec)
```

```
mysql> create database student_attendance2024;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+-----------------------+
| Database              |
+-----------------------+
| carshowroom           |
| information_schema    |
| menagerie             |
| mydb                  |
| mysql                 |
| performance_schema    |
| sakila                |
| student_attendance2024 |
| student_attendence    |
| studentattendancesbgs |
| sys                   |
| world                 |
+-----------------------+
12 rows in set (0.00 sec)

mysql> use student_attendance2024;
Database changed
```

← started using the database after this command.

Create Table

Syntax:

```
Create table tablename(
Attributename1 datatype constraint,
Attributename2 datatype constraint,
…
…
AttributenameN datatype constraint,
```

It is important to observe the following points with respect to **create table** statement:

- The number of columns in a table defines a degree of that relation, which is denoted

by N.
- Attribute name specifies the name of the column in the table.
- Data type specifies the type of data that an attribute can hold.
- Constraint indicates the restriction imposed on the value of an attribute. By default, each attribute can take null value, except for the primary key.

For example:

```
mysql> CREATE TABLE STUDENT(
    -> RollNumber INT,
    -> SName VARCHAR(20),
    -> SDateOfBirth DATE,
    -> GUID CHAR(12),
    -> PRIMARY KEY (RollNumber));
Query OK, 0 rows affected (0.04 sec)

mysql> show tables;
+----------------------------------+
| Tables_in_student_attendance2024 |
+----------------------------------+
| student                          |
+----------------------------------+
1 row in set (0.00 sec)

mysql> desc student;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| RollNumber   | int         | NO   | PRI | NULL    |       |
| SName        | varchar(20) | YES  |     | NULL    |       |
| SDateOfBirth | date        | YES  |     | NULL    |       |
| GUID         | char(12)    | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

view structure of an already created table.

```
mysql> create table guardian(
    -> guid char(12),
    -> gName varchar(20),
    -> gPhone int(10),
    -> gAddress varchar(30),
    -> PRIMARY KEY (guid));
Query OK, 0 rows affected, 1 warning (0.03 sec)

mysql> show tables;
+----------------------------------+
| Tables_in_student_attendance2024 |
+----------------------------------+
| guardian                         |
| student                          |
+----------------------------------+
2 rows in set (0.01 sec)

mysql> desc guardian;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| guid     | char(12)    | NO   | PRI | NULL    |       |
| gName    | varchar(20) | YES  |     | NULL    |       |
| gPhone   | int         | YES  |     | NULL    |       |
| gAddress | varchar(30) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

```
mysql> create table attendance(
    -> attendanceDate DATE,
    -> RollNumber int,
    -> attendanceStatus char(1),
    -> PRIMARY KEY(attendanceDate, RollNumber),
    -> FOREIGN KEY(RollNumber) REFERENCES Student(RollNumber)
    -> ON DELETE CASCADE);
Query OK, 0 rows affected (0.06 sec)

mysql> DESC attendance;
+------------------+---------+------+-----+---------+-------+
| Field            | Type    | Null | Key | Default | Extra |
+------------------+---------+------+-----+---------+-------+
| attendanceDate   | date    | NO   | PRI | NULL    |       |
| RollNumber       | int     | NO   | PRI | NULL    |       |
| attendanceStatus | char(1) | YES  |     | NULL    |       |
+------------------+---------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

Alter table

After reading a table, we may realize that we need to add or remove an attribute or to modify the data type of an existing attribute, or to add constraints in attribute. In all such cases, we need to change or alter the structure or schema of the table by using the

alter statement.

A. Add primary key to a relation

```
mysql> alter table employee
    -> add primary key(eid);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc employee;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| eid      | char(5)     | NO   | PRI | NULL    |       |
| ename    | varchar(30) | YES  |     | NULL    |       |
| email    | varchar(40) | YES  |     | NULL    |       |
| mobilNum | int         | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

B. Add foreign key to a relation

Once primary keys are added, the next step is to add foreign key to the relation, if any Following points needed to be observed while adding a foreign key to a relation:

- The referenced relation must be already created.
- The referenced attributes must be the part of the primary key of the reference to relation.
- Data types and size of referenced and referencing attribute must be same.

```
mysql> desc department;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| depid    | char(3)     | YES  |     | NULL    |       |
| depName  | varchar(20) | YES  |     | NULL    |       |
| location | varchar(30) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> ALTER TABLE department
    -> ADD PRIMARY KEY(depid);
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc department;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| depid    | char(3)     | NO   | PRI | NULL    |       |
| depName  | varchar(20) | YES  |     | NULL    |       |
| location | varchar(30) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)

mysql> desc employee;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| eid      | char(5)     | NO   | PRI | NULL    |       |
| ename    | varchar(30) | YES  |     | NULL    |       |
| email    | varchar(40) | YES  |     | NULL    |       |
| mobilNum | int         | YES  |     | NULL    |       |
| depid    | char(3)     | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

```
mysql> ALTER TABLE employee
    -> ADD FOREIGN KEY(depid) REFERENCES department(depid);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> desc employee;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| eid      | char(5)     | NO   | PRI | NULL    |       |
| ename    | varchar(30) | YES  |     | NULL    |       |
| email    | varchar(40) | YES  |     | NULL    |       |
| mobilNum | int         | YES  |     | NULL    |       |
| depid    | char(3)     | YES  | MUL | NULL    |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

C. Add constraint UNIQUE to an existing attribute.

D. Add an attribute to an existing table.

E. Modify datatype of an attribute

```
mysql> desc guardian;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| guid     | char(12)    | NO   | PRI | NULL    |       |
| gName    | varchar(20) | YES  |     | NULL    |       |
| gPhone   | int         | YES  |     | NULL    |       |
| gAddress | varchar(30) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> ALTER TABLE guardian
    -> MODIFY gAddress varchar(40);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc guardian;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| guid     | char(12)    | NO   | PRI | NULL    |       |
| gName    | varchar(20) | YES  |     | NULL    |       |
| gPhone   | int         | YES  |     | NULL    |       |
| gAddress | varchar(40) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

F. Modify constraint of an attribute
G. Add default value to an attribute
H. Remove an attribute from a table by using DROP keyword.
I. Remove primary key from a table by using DROP keyword.
Note: the primary key attribute should not be referred to other table as foreign key.

DROP Statement

Sometimes a table in a database or a database itself needs to be removed. We can use a drop statement to remove a database or a table permanently from the system. However, one should be very cautious while using this statement as it cannot be undone.

Syntax:

```
DROP TABLE table_name;
```

Syntax:

```
DROP DATABASE database_name;
```

For example:

```
mysql> create table t1(
    -> x int);
Query OK, 0 rows affected (0.02 sec)

mysql> show tables;
+-----------------------------------+
| Tables_in_student_attendance2024 |
+-----------------------------------+
| attendance                        |
| department                        |
| employee                          |
| guardian                          |
| student                           |
| t1                                |
+-----------------------------------+
6 rows in set (0.00 sec)

mysql> desc t1
    -> ;
+-------+------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-------+------+------+-----+---------+-------+
| x     | int  | YES  |     | NULL    |       |
+-------+------+------+-----+---------+-------+
1 row in set (0.00 sec)

mysql> DROP TABLE t1;
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> show tables;
+---------------------------------+
| Tables_in_student_attendance2024 |
+---------------------------------+
| attendance                      |
| department                      |
| employee                        |
| guardian                        |
| student                         |
+---------------------------------+
5 rows in set (0.00 sec)
```

Insertion of Records

INSERT INTO statement is used to insert new records in a table.
Syntax:

```
INSERT INTO tableName
VALUES(value1, value2, … valueN);
```

Here value one corresponds to attribute one value 2 corresponds to attribute two, and so on. Note that we need not to specify attribute name in the insert statement. If there are exactly the same number of values in the insert statement, as the total number of attributes in the table.

Caution: While populating records in a table with foreign key, ensure that records in referenced table are already populated.

```
mysql> INSERT INTO guardian
    -> VALUES ('G0000000001', 'Amit Kumar', 987654101, 'Garia Garden');
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO guardian
    -> VALUES ('G0000000002', 'Sumit Kumar', 987655101, 'Garia Garden');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO guardian
    -> VALUES ('G0000000003', 'Ajay Parakh', 887653102, 'Garia Station Road');
Query OK, 1 row affected (0.01 sec)
```

Another Syntax:

```
INSERT INTO tableName(column1, column2,…columnN)
VALUES (value1, value2, … valueN);
```

For example:

```
mysql> INSERT INTO guardian(guid, gName, gPhone, gAddress)
    -> VALUES('G00000000004', 'Harshit Agarwal', 888999111, 'New Garia');
Query OK, 1 row affected (0.01 sec)

mysql> select * from guardian;
+--------------+-----------------+-----------+--------------------+
| guid         | gName           | gPhone    | gAddress           |
+--------------+-----------------+-----------+--------------------+
| G00000000004 | Harshit Agarwal | 888999111 | New Garia          |
| G0000000001  | Amit Kumar      | 987654101 | Garia Garden       |
| G0000000002  | Sumit Kumar     | 987655101 | Garia Garden       |
| G0000000003  | Ajay Parakh     | 887653102 | Garia Station Road |
+--------------+-----------------+-----------+--------------------+
4 rows in set (0.00 sec)
```

```
mysql> INSERT INTO guardian(guid, gAddress, gName, gPhone)
    -> VALUES('G00000000005','Maheshtalla', 'Nupur Verma', 999888777);
Query OK, 1 row affected (0.01 sec)

mysql> select * from guardian;
+--------------+-----------------+-----------+--------------------+
| guid         | gName           | gPhone    | gAddress           |
+--------------+-----------------+-----------+--------------------+
| G00000000004 | Harshit Agarwal | 888999111 | New Garia          |
| G00000000005 | Nupur Verma     | 999888777 | Maheshtalla        |
| G0000000001  | Amit Kumar      | 987654101 | Garia Garden       |
| G0000000002  | Sumit Kumar     | 987655101 | Garia Garden       |
| G0000000003  | Ajay Parakh     | 887653102 | Garia Station Road |
+--------------+-----------------+-----------+--------------------+
5 rows in set (0.01 sec)

mysql>
```

ordering of attributes does not matter.
Depending upon your ordering you can supply values.

```
mysql> ALTER TABLE student
    -> ADD FOREIGN KEY(GUID) REFERENCES guardian(GUID);
Query OK, 1 row affected (0.09 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> desc student;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| RollNumber   | int         | NO   | PRI | NULL    |       |
| SName        | varchar(20) | YES  |     | NULL    |       |
| SDateOfBirth | date        | YES  |     | NULL    |       |
| GUID         | char(12)    | YES  | MUL | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> INSERT INTO student
    -> VALUES(2, 'James Parakh', '2007-05-17', 'G00000000003');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`student_attendance2024`.`student`
, CONSTRAINT `student_ibfk_1` FOREIGN KEY (`GUID`) REFERENCES `guardian` (`guid`))
mysql>
```

We can only provide values from parent table to child table:

```
mysql> INSERT INTO student
    -> VALUES(2, 'James Parakh', '2007-05-17', 'G0000000003');
Query OK, 1 row affected (0.01 sec)

mysql> select * from student;
+------------+--------------+--------------+-------------+
| RollNumber | SName        | SDateOfBirth | GUID        |
+------------+--------------+--------------+-------------+
|          1 | Jack Parakh  | 2007-05-17   | G0000000003 |
|          2 | James Parakh | 2007-05-17   | G0000000003 |
+------------+--------------+--------------+-------------+
2 rows in set (0.00 sec)
```

SQL for Data Query

The SQL statement **SELECT** is used to retrieve data from the tables in a database and is also called a query statement.

SELECT Statement

The SQL statement **SELECT** is used to retrieve data from the tables in a database, and the output is also displayed in tabular form.

Syntax:

```
SELECT attribute1, attribute2, attribute3,…
FROM table_name
WHERE condition;
```

Here attribute1, attribute2,… Are the column names of table table_name from which we want to retrieve data. The **FROM** clause is always written with the select clause as it specifies the name of the table from which data is to be retrieved. The **WHERE** clause is optional and is used to retrieve data that meet the specified condition(s).

To select all data available in a table, we use the following select statement:

```
select * from table_name;
```

For example:

```
select * from student;
```

```
mysql> select * from student;
+------------+--------------+--------------+-------------+
| RollNumber | SName        | SDateOfBirth | GUID        |
+------------+--------------+--------------+-------------+
|          1 | Jack Parakh  | 2007-05-17   | G0000000003 |
|          2 | James Parakh | 2007-05-17   | G0000000003 |
+------------+--------------+--------------+-------------+
2 rows in set (0.00 sec)
```

NSERT INTO `student_attendance2024`.`department` (`depid`, `depName`, `location`) VALUES ('D03', 'Accounts', 'Garia');
INSERT INTO `student_attendance2024`.`department` (`depid`, `depName`, `location`) VALUES ('D04', 'Marketing', 'Garia');
INSERT INTO `student_attendance2024`.`department` (`depid`, `depName`, `location`) VALUES ('D05', 'Finance', 'Patuli');
INSERT INTO `student_attendance2024`.`department` (`depid`, `depName`, `location`) VALUES

```
('D01', 'Supply', 'Tollygung');
INSERT INTO `student_attendance2024`.`department` (`depid`, `depName`, `location`) VALUES
('D02', 'Maintenance', 'Tollygung');
```

Retrieve selected columns

Consider the table employee:

```
mysql> select * from employee;
+-------+----------+--------+-------+--------+
| EmpNo | Ename    | Salary | Bonus | DeptId |
+-------+----------+--------+-------+--------+
|   101 | Aaliya   |  10000 |   234 | D02    |
|   102 | Kritika  |  60000 |   123 | D01    |
|   103 | Shabbir  |  45000 |   566 | D01    |
|   104 | Gurpreet |  19000 |   565 | D04    |
|   105 | Joseph   |  34000 |   875 | D03    |
|   106 | Sanya    |  48000 |   695 | D02    |
|   107 | Vergese  |  15000 |  NULL | D01    |
|   108 | Nachaobi |  29000 |  NULL | D05    |
|   109 | Daribha  |  42000 |  NULL | D04    |
|   101 | Aaliya   |  10000 |   234 | D02    |
|   110 | Tanya    |  50000 |   467 | D05    |
|   111 | Kapil    |  15000 |   763 | D02    |
+-------+----------+--------+-------+--------+
12 rows in set (0.00 sec)
```

A. Retrieve selected columns

```
mysql> select Ename from employee;
+----------+
| Ename    |
+----------+
| Aaliya   |
| Kritika  |
| Shabbir  |
| Gurpreet |
| Joseph   |
| Sanya    |
| Vergese  |
| Nachaobi |
| Daribha  |
| Aaliya   |
| Tanya    |
| Kapil    |
+----------+
12 rows in set (0.00 sec)
```

```
mysql> select Ename, salary from employee;
+----------+--------+
| Ename    | salary |
+----------+--------+
| Aaliya   |  10000 |
| Kritika  |  60000 |
| Shabbir  |  45000 |
| Gurpreet |  19000 |
| Joseph   |  34000 |
| Sanya    |  48000 |
| Vergese  |  15000 |
| Nachaobi |  29000 |
| Daribha  |  42000 |
| Aaliya   |  10000 |
| Tanya    |  50000 |
| Kapil    |  15000 |
+----------+--------+
12 rows in set (0.00 sec)
```

B. Renaming of columns

```
mysql> select Ename as Name, salary as Salary from employee;
+----------+--------+
| Name     | Salary |
+----------+--------+
| Aaliya   |  10000 |
| Kritika  |  60000 |
| Shabbir  |  45000 |
| Gurpreet |  19000 |
| Joseph   |  34000 |
| Sanya    |  48000 |
| Vergese  |  15000 |
| Nachaobi |  29000 |
| Daribha  |  42000 |
| Aaliya   |  10000 |
| Tanya    |  50000 |
| Kapil    |  15000 |
+----------+--------+
12 rows in set (0.00 sec)
```

```
mysql> select Ename as Name, (salary * 12) as 'Yearly Salary' from employee;
+----------+---------------+
| Name     | Yearly Salary |
+----------+---------------+
| Aaliya   |        120000 |
| Kritika  |        720000 |
| Shabbir  |        540000 |
| Gurpreet |        228000 |
| Joseph   |        408000 |
| Sanya    |        576000 |
| Vergese  |        180000 |
| Nachaobi |        348000 |
| Daribha  |        504000 |
| Aaliya   |        120000 |
| Tanya    |        600000 |
| Kapil    |        180000 |
+----------+---------------+
12 rows in set (0.00 sec)
```

C. Distinct clause:

By default SQL shows all the data retrieved through query output. However, there can be duplicate values. The select statement, when combined with distinct clause, returns a record without repetition. For example, while. retrieving a department number from the employee relation, there can be a duplicate values, as many employees are assigned to the same department. To select unique department number for all the employees we use **'distinct'** as shown below:

```
mysql> select * from employee;
+-------+----------+--------+-------+--------+
| EmpNo | Ename    | Salary | Bonus | DeptId |
+-------+----------+--------+-------+--------+
|   101 | Aaliya   |  10000 |   234 | D02    |
|   102 | Kritika  |  60000 |   123 | D01    |
|   103 | Shabbir  |  45000 |   566 | D01    |
|   104 | Gurpreet |  19000 |   565 | D04    |
|   105 | Joseph   |  34000 |   875 | D03    |
|   106 | Sanya    |  48000 |   695 | D02    |
|   107 | Vergese  |  15000 |  NULL | D01    |
|   108 | Nachaobi |  29000 |  NULL | D05    |
|   109 | Daribha  |  42000 |  NULL | D04    |
|   101 | Aaliya   |  10000 |   234 | D02    |
|   110 | Tanya    |  50000 |   467 | D05    |
|   111 | Kapil    |  15000 |   763 | D02    |
+-------+----------+--------+-------+--------+
12 rows in set (0.00 sec)

mysql> select distinct DeptId from employee;
+--------+
| DeptId |
+--------+
| D02    |
| D01    |
| D04    |
| D03    |
| D05    |
+--------+
5 rows in set (0.00 sec)
```

D. WHERE Clause:

The wear clauses used to retrieve data that meet some specified condition. In the employee table, more than one employee can be working in one department.

```
mysql> select distinct salary from employee where deptid='D01';
+--------+
| salary |
+--------+
|  60000 |
|  45000 |
|  15000 |
+--------+
3 rows in set (0.00 sec)
```

```
mysql> select * from employee
    -> where salary > 20000 and deptid='D04';
+-------+---------+--------+-------+--------+
| EmpNo | Ename   | Salary | Bonus | DeptId |
+-------+---------+--------+-------+--------+
|   109 | Daribha |  42000 |  NULL | D04    |
+-------+---------+--------+-------+--------+
1 row in set (0.00 sec)

mysql> select * from employee
    -> where not ename='Aaliya';
+-------+----------+--------+-------+--------+
| EmpNo | Ename    | Salary | Bonus | DeptId |
+-------+----------+--------+-------+--------+
|   102 | Kritika  |  60000 |   123 | D01    |
|   103 | Shabbir  |  45000 |   566 | D01    |
|   104 | Gurpreet |  19000 |   565 | D04    |
|   105 | Joseph   |  34000 |   875 | D03    |
|   106 | Sanya    |  48000 |   695 | D02    |
|   107 | Vergese  |  15000 |  NULL | D01    |
|   108 | Nachaobi |  29000 |  NULL | D05    |
|   109 | Daribha  |  42000 |  NULL | D04    |
|   110 | Tanya    |  50000 |   467 | D05    |
|   111 | Kapil    |  15000 |   763 | D02    |
+-------+----------+--------+-------+--------+
10 rows in set (0.00 sec)
```

```
mysql> select ename as Name, deptid as 'Department ID' from employee
    -> where salary >= 20000 and salary <= 50000;
+----------+---------------+
| Name     | Department ID |
+----------+---------------+
| Shabbir  | D01           |
| Joseph   | D03           |
| Sanya    | D02           |
| Nachaobi | D05           |
| Daribha  | D04           |
| Tanya    | D05           |
+----------+---------------+
6 rows in set (0.00 sec)
```

~~BETWEEN Clause~~

The **'between operator'** defines the range of values in which the column value must fall into to make the condition true.

```
mysql> select ename as Name, deptid as 'Department ID' from employee
    -> where salary between 20000 and 50000
    -> ;
+----------+---------------+
| Name     | Department ID |
+----------+---------------+
| Shabbir  | D01           |
| Joseph   | D03           |
| Sanya    | D02           |
| Nachaobi | D05           |
| Daribha  | D04           |
| Tanya    | D05           |
+----------+---------------+
6 rows in set (0.00 sec)
```

Printing records for selected departments using 'OR' operator.

```
mysql> select * from employee
    -> where
    -> DeptId = 'D01' OR DeptId = 'D02' OR DeptID = 'D04';
+-------+----------+--------+-------+--------+
| EmpNo | Ename    | Salary | Bonus | DeptId |
+-------+----------+--------+-------+--------+
|   101 | Aaliya   |  10000 |   234 | D02    |
|   102 | Kritika  |  60000 |   123 | D01    |
|   103 | Shabbir  |  45000 |   566 | D01    |
|   104 | Gurpreet |  19000 |   565 | D04    |
|   106 | Sanya    |  48000 |   695 | D02    |
|   107 | Vergese  |  15000 |  NULL | D01    |
|   109 | Daribha  |  42000 |  NULL | D04    |
|   101 | Aaliya   |  10000 |   234 | D02    |
|   111 | Kapil    |  15000 |   763 | D02    |
+-------+----------+--------+-------+--------+
9 rows in set (0.00 sec)
```

```
mysql> select * from employee
    -> where
    -> NOT (DeptId = 'D01' OR DeptId = 'D02' OR DeptID = 'D04');
+-------+----------+--------+-------+--------+
| EmpNo | Ename    | Salary | Bonus | DeptId |
+-------+----------+--------+-------+--------+
|   105 | Joseph   |  34000 |   875 | D03    |
|   108 | Nachaobi |  29000 |  NULL | D05    |
|   110 | Tanya    |  50000 |   467 | D05    |
+-------+----------+--------+-------+--------+
3 rows in set (0.00 sec)
```

MEMBERSHIP operator IN

The 'IN' operator compares a value with a set of values and returns true if the value belongs to that set.

```
mysql> select * from employee
    -> where
    -> DeptId in ('D01','D02','D04');
+-------+----------+--------+-------+--------+
| EmpNo | Ename    | Salary | Bonus | DeptId |
+-------+----------+--------+-------+--------+
|   101 | Aaliya   |  10000 |   234 | D02    |
|   102 | Kritika  |  60000 |   123 | D01    |
|   103 | Shabbir  |  45000 |   566 | D01    |
|   104 | Gurpreet |  19000 |   565 | D04    |
|   106 | Sanya    |  48000 |   695 | D02    |
|   107 | Vergese  |  15000 |  NULL | D01    |
|   109 | Daribha  |  42000 |  NULL | D04    |
|   101 | Aaliya   |  10000 |   234 | D02    |
|   111 | Kapil    |  15000 |   763 | D02    |
+-------+----------+--------+-------+--------+
9 rows in set (0.00 sec)
```

The members of 'D01', 'D02', 'D04' is selected for the output.

```
mysql> select * from employee
    -> where
    -> DeptId not in ('D01','D02','D04');
+-------+----------+--------+-------+--------+
| EmpNo | Ename    | Salary | Bonus | DeptId |
+-------+----------+--------+-------+--------+
|   105 | Joseph   |  34000 |   875 | D03    |
|   108 | Nachaobi |  29000 |  NULL | D05    |
|   110 | Tanya    |  50000 |   467 | D05    |
+-------+----------+--------+-------+--------+
3 rows in set (0.00 sec)
```

Not is applied in the above query to select the rows which are not the member of deptid('D01', 'D02', and 'D04').

ORDER BY Clause

Order by clause is used to display data in an ordered form with respect to a specified column. By default, **'order by'** displays records in ascending order of the specified columns values. To display the record in descending order. The DESC (means descending) keyword needs to be written with that column.

```
mysql> select * from employee
    -> order by salary;
+-------+----------+--------+-------+--------+
| EmpNo | Ename    | Salary | Bonus | DeptId |
+-------+----------+--------+-------+--------+
|   101 | Aaliya   |  10000 |   234 | D02    |
|   107 | Vergese  |  15000 |  NULL | D01    |
|   111 | Kapil    |  15000 |   763 | D02    |
|   104 | Gurpreet |  19000 |   565 | D04    |
|   108 | Nachaobi |  29000 |  NULL | D05    |
|   105 | Joseph   |  34000 |   875 | D03    |
|   109 | Daribha  |  42000 |  NULL | D04    |
|   103 | Shabbir  |  45000 |   566 | D01    |
|   106 | Sanya    |  48000 |   695 | D02    |
|   110 | Tanya    |  50000 |   467 | D05    |
|   102 | Kritika  |  60000 |   123 | D01    |
+-------+----------+--------+-------+--------+
11 rows in set (0.00 sec)

mysql> select * from employee
    -> order by salary desc;
+-------+----------+--------+-------+--------+
| EmpNo | Ename    | Salary | Bonus | DeptId |
+-------+----------+--------+-------+--------+
|   102 | Kritika  |  60000 |   123 | D01    |
|   110 | Tanya    |  50000 |   467 | D05    |
|   106 | Sanya    |  48000 |   695 | D02    |
|   103 | Shabbir  |  45000 |   566 | D01    |
|   109 | Daribha  |  42000 |  NULL | D04    |
|   105 | Joseph   |  34000 |   875 | D03    |
|   108 | Nachaobi |  29000 |  NULL | D05    |
|   104 | Gurpreet |  19000 |   565 | D04    |
|   107 | Vergese  |  15000 |  NULL | D01    |
|   111 | Kapil    |  15000 |   763 | D02    |
|   101 | Aaliya   |  10000 |   234 | D02    |
+-------+----------+--------+-------+--------+
11 rows in set (0.00 sec)
```

```
mysql> select * from employee
    -> order by ename desc;
+-------+----------+--------+-------+--------+
| EmpNo | Ename    | Salary | Bonus | DeptId |
+-------+----------+--------+-------+--------+
|   107 | Vergese  |  15000 |  NULL | D01    |
|   110 | Tanya    |  50000 |   467 | D05    |
|   103 | Shabbir  |  45000 |   566 | D01    |
|   106 | Sanya    |  48000 |   695 | D02    |
|   108 | Nachaobi |  29000 |  NULL | D05    |
|   102 | Kritika  |  60000 |   123 | D01    |
|   111 | Kapil    |  15000 |   763 | D02    |
|   105 | Joseph   |  34000 |   875 | D03    |
|   104 | Gurpreet |  19000 |   565 | D04    |
|   109 | Daribha  |  42000 |  NULL | D04    |
|   101 | Aaliya   |  10000 |   234 | D02    |
+-------+----------+--------+-------+--------+
11 rows in set (0.00 sec)
```

~~Handling NULL Values~~

SQL supports a special value called NULL to represent a missing or unknown value. For example, the bonus column in the employee table can have missing values for certain records. Hence, null is used to represent such unknown values. It is important to note that the. null is different from zero. Also any arithmetic operation performed with null values gives null.
~~For example, 5+ NULL = NULL, because NULL is unknown hence the result is also unknown in order to check for NULL~~ value in a column we use **'is null'** operator.

```
mysql> select * from employee
    -> where bonus is null;
+-------+----------+--------+-------+--------+
| EmpNo | Ename    | Salary | Bonus | DeptId |
+-------+----------+--------+-------+--------+
|   107 | Vergese  |  15000 |  NULL | D01    |
|   108 | Nachaobi |  29000 |  NULL | D05    |
|   109 | Daribha  |  42000 |  NULL | D04    |
+-------+----------+--------+-------+--------+
3 rows in set (0.00 sec)
```

```
mysql> select * from employee
    -> where bonus is not null and deptid = 'D01';
+-------+---------+--------+-------+--------+
| EmpNo | Ename   | Salary | Bonus | DeptId |
+-------+---------+--------+-------+--------+
|   102 | Kritika |  60000 |   123 | D01    |
|   103 | Shabbir |  45000 |   566 | D01    |
+-------+---------+--------+-------+--------+
2 rows in set (0.00 sec)
```

Substring pattern matching

Many a times we come across a situation where we do not want to query by matching exact text or value. Rather, we are interested to find matching of only a few characters or values in column values. For example, to find out the name, starting with "T". or to find out a pin code, starting with "60". This is called substring pattern matching. We cannot match such pattern using equal to operator as we are not looking for exact match. SQL provides a **"like"** operator that can be used with. the wear clause to search for a specified pattern in a column.

The "LIKE" operator makes use of the following two wild card characters:

- %(percent) - used to represent zero 1 or multiple characters.
- _(underscore) - He used to represent exactly a single character.

Examples:

```
mysql> select * from employee
    -> where ename like 'K%';
+-------+---------+--------+-------+--------+
| EmpNo | Ename   | Salary | Bonus | DeptId |
+-------+---------+--------+-------+--------+
|   102 | Kritika |  60000 |   123 | D01    |
|   111 | Kapil   |  15000 |   763 | D02    |
+-------+---------+--------+-------+--------+
2 rows in set (0.00 sec)

mysql> select * from employee
    -> where ename like '_A%';
+-------+----------+--------+-------+--------+
| EmpNo | Ename    | Salary | Bonus | DeptId |
+-------+----------+--------+-------+--------+
|   106 | Sanya    |  48000 |   695 | D02    |
|   108 | Nachaobi |  29000 |  NULL | D05    |
|   109 | Daribha  |  42000 |  NULL | D04    |
|   110 | Tanya    |  50000 |   467 | D05    |
|   111 | Kapil    |  15000 |   763 | D02    |
|   101 | Aaliya   |  10000 |   234 | D02    |
+-------+----------+--------+-------+--------+
6 rows in set (0.00 sec)
```

```
mysql> select * from employee
    -> where ename like '%A%';
+-------+----------+--------+-------+--------+
| EmpNo | Ename    | Salary | Bonus | DeptId |
+-------+----------+--------+-------+--------+
|   102 | Kritika  |  60000 |   123 | D01    |
|   103 | Shabbir  |  45000 |   566 | D01    |
|   106 | Sanya    |  48000 |   695 | D02    |
|   108 | Nachaobi |  29000 |  NULL | D05    |
|   109 | Daribha  |  42000 |  NULL | D04    |
|   110 | Tanya    |  50000 |   467 | D05    |
|   111 | Kapil    |  15000 |   763 | D02    |
|   101 | Aaliya   |  10000 |   234 | D02    |
+-------+----------+--------+-------+--------+
8 rows in set (0.00 sec)
```

```
mysql> select * from employee
    -> where ename like '%e%';
+-------+----------+--------+-------+--------+
| EmpNo | Ename    | Salary | Bonus | DeptId |
+-------+----------+--------+-------+--------+
|   104 | Gurpreet |  19000 |   565 | D04    |
|   105 | Joseph   |  34000 |   875 | D03    |
|   107 | Vergese  |  15000 |  NULL | D01    |
+-------+----------+--------+-------+--------+
3 rows in set (0.00 sec)
```

```
mysql> select * from employee
    -> where ename like '%i%';
+-------+----------+--------+-------+--------+
| EmpNo | Ename    | Salary | Bonus | DeptId |
+-------+----------+--------+-------+--------+
|   102 | Kritika  |  60000 |   123 | D01    |
|   103 | Shabbir  |  45000 |   566 | D01    |
|   108 | Nachaobi |  29000 |  NULL | D05    |
|   109 | Daribha  |  42000 |  NULL | D04    |
|   111 | Kapil    |  15000 |   763 | D02    |
|   101 | Aaliya   |  10000 |   234 | D02    |
+-------+----------+--------+-------+--------+
6 rows in set (0.00 sec)
```

```
mysql> select * from employee
    -> where ename like '%e%'
    -> and salary > 40000;
Empty set (0.00 sec)

mysql> select * from employee
    -> where ename like '%i%'
    -> and salary > 40000;
+-------+---------+--------+-------+--------+
| EmpNo | Ename   | Salary | Bonus | DeptId |
+-------+---------+--------+-------+--------+
|   102 | Kritika |  60000 |   123 | D01    |
|   103 | Shabbir |  45000 |   566 | D01    |
|   109 | Daribha |  42000 |  NULL | D04    |
+-------+---------+--------+-------+--------+
3 rows in set (0.00 sec)
```

```
mysql> select * from employee
    -> where ename like '%se%';
+-------+---------+--------+-------+--------+
| EmpNo | Ename   | Salary | Bonus | DeptId |
+-------+---------+--------+-------+--------+
|   105 | Joseph  |  34000 |   875 | D03    |
|   107 | Vergese |  15000 |  NULL | D01    |
+-------+---------+--------+-------+--------+
2 rows in set (0.00 sec)
```