

Structured Query Language:

Introduction:

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

SQL is a query language, not a database system. You are required to install DBMS software in your system to perform SQL Language operations 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 systems. SQL provides interaction between the user and the database via a set of standard commands.

In simple words, SQL is a language that helps users to communicate with databases. SQL is not a case-sensitive language means you can type your query in small or capital letters.



DIAGRAM: 7

HOW SQL PROVIDE INTERACTION BETWEEN USER AND DATABASE

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

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

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

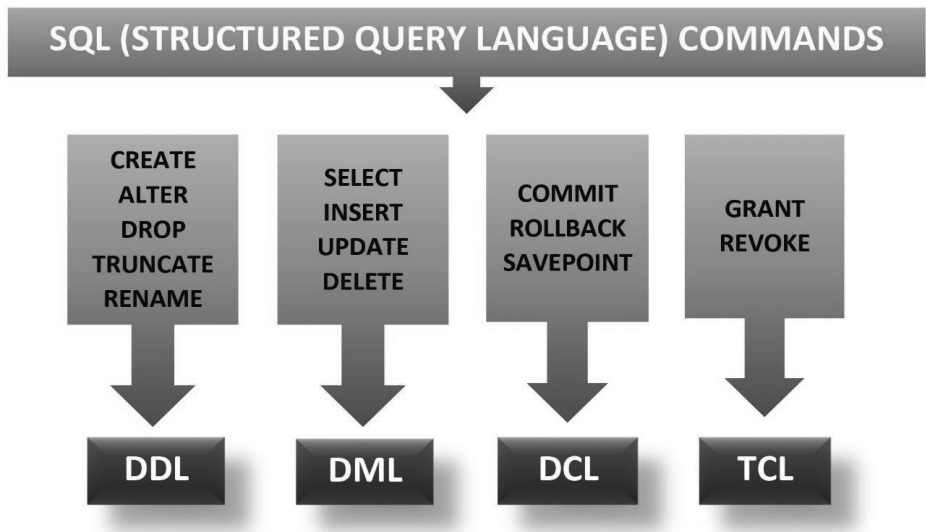


DIAGRAM: 8

SQL COMMANDS CAN BE CLASSIFIED INTO FOLLOWING CATEGORY

SQL perform the following operation:

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

SQL Commands:

SQL commands are a predefined set of commands which are already defined in SQL. Commands are combinations of keywords and statements you want to execute. Keywords are reserved words that have special meaning for SQL, you don't need to define them as they are already defined in SQL. All you need to use these keywords with your particular statements.

CLAUSE IN SQL:

Clause: Clause are built-in functions which are used to deal with data inside the table that help SQL to filter and analyse data quickly.

CLAUSE IN SQL

WHERE CLAUSE

Used to select or filter rows based on particular valid condition. It is mainly used with SELECT, UPDATE AND DELETE

FROM CLAUSE

Used to mention name of table or source table from where you want to fetch/retrieve data. It is mainly used with SELECT

ORDER BY CLAUSE

Used to sort the result set in ascending order (by default). You can use DESC keyword to sort in descending order

GROUP BY CLAUSE

Used to group rows from a table based on one or more columns. It's mainly used with aggregate functions.

HAVING CLAUSE

Used with GROUP BY clause to filter the result of a query based on aggregate function applied to grouped columns

LIKE CLAUSE

Used with WHERE clause to search for a specific pattern in a column. It uses wildcard (%,_) characters to match the strings

Any SQL statement is composed of two or more clauses.

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

NOTE: All SQL statements are terminated with (;) semicolon

SQL statements are not case sensitive, which means SQL treats both upper and lowercase commands as the same.

SQL (Structured Query Language)

Constraints:

Constraints in SQL are sets 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 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 the action can't be completed.

Types of constraints:

1. **Unique:** This constraint ensures that all values present or inserted in a column are different.
2. **Not Null:** This constraint ensures that no null values are present or inserted in a column.
3. **Primary key:** The primary key applies both unique and not null constraints to the column. It uniquely identifies a unique tuple/row in a relation/table.
4. **Foreign Key:** unique, not null and primary key constraint applies to a single table whereas the foreign key constraint applies to two tables.

For example: we have two tables' - student and awards, as follows:

Table: Student				Table: Awards		
ID	Name	Age	City	ID	Award	Sport
1	Amit	15	Delhi	1	Gold	Badminton
2	Madhu	14	Gurugram	2	Silver	Tennis
3	Manoj	15	Noida	1	Silver	Hockey
4	Asif	15	Faridabad	4	Bronze	Badminton

Here we will establish foreign key constraints on the column **ID** of the **Student** table and the column **ID** of the **Awards** table. Here we will consider **the Student table** as **the parent table** and the **Awards table** as the **child table**. The following rules must be followed:

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

Let's assume that our parent table **Student** has already been created. Now we will create child table **awards** and apply **foreign key** constraints on it. (Note: Foreign key relation can only be applied on child table)

SQL Query

create table <child table name> (<column name 1> <data type>, <column name 2> <data type>,..., foreign key(<column name>) references <parent table name>(<column name>));

```
mysql> create table awards(ID int, Award char(50),Sport char(50),foreign key(ID) references student(ID));
Query OK, 0 rows affected (0.06 sec)
```

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

```
mysql> select * from student;
+----+-----+-----+-----+
| ID | Name  | Age  | City      |
+----+-----+-----+-----+
| 1  | Amit  | 15   | Delhi     |
| 2  | Madhu | 14   | Gurugram  |
| 3  | Manoj | 15   | Noida     |
| 4  | Asif  | 15   | Faridabad |
+----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> insert into awards values(1,'Gold','Badminton');
Query OK, 1 row affected (0.01 sec)
```

Here, As ID '1' is already present in the parent table 'student', So, ID '1' can be inserted in the child table 'awards'.

Now, we will try to insert ID '6' in the child table 'awards' which is not present in the parent table 'student'.

```
mysql> insert into awards values(6,'Gold','Football');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails ('demo'.`awards`, CONSTRAINT `awards_ibfk_1` FOREIGN KEY (`ID`) REFERENCES `student` (`ID`))
```

Here we can see that when we tried to insert value 6 in column ID of child table awards, it showed an error. This is how a foreign key works.

A few basic SQL Queries:

<p>Create database: To create a new database, the create database command is used.</p> <p>Query: create database <database name>;</p>	<pre>mysql> create database test; Query OK, 1 row affected (0.00 sec)</pre>
<p>Show databases: To view the list of available/created databases in SQL, the <i>show databases</i> command is used.</p> <p>Query: show databases;</p>	<pre>mysql> show databases; +-----+ Database +-----+ information_schema test +-----+ 2 rows in set (0.00 sec)</pre>
<p>Use database: To select a database among available databases, <i>use</i> command is used.</p> <p>Query: use <database name>;</p>	<pre>mysql> use test; Database changed mysql></pre>

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

Query: 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 the following attributes and data types:

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

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

```
mysql> create table student(Student_ID int,Student_Name char(30),Age int,Phone int,Address varchar(50));
Query OK, 0 rows affected (0.16 sec)
```

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

Query: desc <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 a table, an insert command is used (one row at a time). Here in this example, data of 4 students are inserted in the table student.

Query: insert into <table name> values (<value1>, <value2>, <value3> ...);

```
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 the student table that was

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

<p>inserted in the previous command.</p> <p>Query: select * from <table name>;</p> <p>Here * means all columns</p>	
<p>Drop table command: To delete data as well as the structure of a table, drop command is used.</p> <p>Query: drop table <table name></p>	<pre>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)</pre>
<p>Drop database command: To delete a database along with all the tables present in the database, drop command is used.</p> <p>Query: drop database <database name></p>	<pre>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)</pre>

DDL (Data Definition Language) Commands:

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

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

We have already covered a few DDL Commands like create database, create table, drop database, and drop table. A 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.

Query: alter table <table name> [alter option]

(a) Add a column to the table: We have a table student which was created in the 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)
```

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

Example: Let's 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)
```

(b) Drop a column from the table: Let's delete the column 'class' from the table 'student' which we added in the previous section.

Command: 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)
```

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

(i) Changing column name: We can change the column name of a table using the alter command.

Example: In table student, Let's change column name Student_ID to ID.

Query: 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)
```

- (ii) **Changing column data type:** We can change the column data type from varchar to char or int to varchar etc. of a table using the alter command.

Example: in table student, Let's change the datatype of column 'ID' from int to varchar.

Query: alter table <table name> modify column <column name> <new data type> <size>

```
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> 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)
```

- (iii) **Changing the maximum size of the data in a column:** We can change the maximum size of the data in a column of a table using the alter command.

Example: In table student, Let's change the size of column ID from varchar(50) to varchar(40)

Query: alter table <table name> modify 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)
```

(iv) Changing the order of the column: We can change the order of the column of a table using the alter command. For example, in table student, we are going to place column ID after column Age.

Query: 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 the 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)
```

(v) **Add/drop constraints/column:** We can add/drop constraints in a table using the alter command.

➤ **Adding primary key:** Let's add the primary key at column 'ID' using the alter command.

Query: 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:** Let's remove the primary key at column ID which was added in the previous section.

Query: 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:** Let's add a column 'country' with data type char of size 50 to the table 'student' using the alter command.

Query: 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:** Let's remove a column 'country' which was added in the last section using the 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 Definition 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 a few DML Commands like insert and select. Now we will discuss delete and update commands.

- Delete command:** The 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 satisfy the condition will be deleted from the table.

Command: delete from <table name> where <condition>;

Now, let's delete the data of all those students from the 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)
```

2. **Update command:** The 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 satisfy the condition will be updated in the table.

Command: 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. The alias can be created using the 'as' keyword. Creating aliases doesn't change the 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:

The distinct clause is used to display unique values by neglecting all the duplicate values. 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 the 'student' table, two duplicate IDs 1 and 3 are present. Now using distinct clauses we can get 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 all duplicate IDs are removed but it is temporary. Duplicate values are not removed and are 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 meet 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 the select statement to filter the result.
- Where clause can be used with an update statement to update the data of the table that matches with the condition.

- Where clause can be used with a delete statement to delete the rows of the table that match with the condition.

Query: where <condition>;

Examples:

- 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)
```

- To filter the result based on multiple conditions:

```
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 clause and **not in clause** in SQL is used to filter the rows in output based on a list of values.

Query for in clause: where <column name> in (item1, item2,...);

Query 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 live in either 'Delhi' or 'Jaipur' or 'Gurugram'. Now to solve this problem, we have two ways. Either we write multiple comparisons 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
5	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
5	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 live 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.

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

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

```
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.

Query: 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 mention the order of sorting as it is the default mode of sorting.

```
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;
```

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 the absence of a value or a field doesn't have 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, 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)

Example: To find out the names of students whose phone numbers are 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.

Example: To find out the names of students whose phone numbers are 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 the **where** 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 '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 a letter 'a' at the second position, its pattern will be '_a%'. As we know there must be exactly one character before 'a' and we don't know how many characters are thereafter 'a', so the '_' sign is used before 'a' and the '%' 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 characters, its pattern will be '_____'. As we know there must be exactly 5 characters, so the '_' 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 characters and ends with 't', its pattern will be '____t'. As we know there must be exactly 7 characters, so the '_' 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.

Command: 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.

Query: delete from <table name> where <condition>;

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

```
mysql> select * from student;
```

ID	Student_Name	Age	Phone	Address
1	Amit	18	98769876	Sonipat
2	Sonam	16	88769876	gurugram
3	Mahesh	17	68769876	jaipur
4	Priya	18	78769876	noida
5	Monika	17	98769876	delhi
1	Ajay	17	98769857	delhi
3	sonal	18	94769857	noida
6	Tanya	16	NULL	delhi

8 rows in set (0.00 sec)

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

```
mysql> select * from student;
```

ID	Student_Name	Age	Phone	Address
1	Amit	18	98769876	Sonipat
2	Sonam	16	88769876	gurugram
3	Mahesh	17	68769876	jaipur
4	Priya	18	78769876	noida
5	Monika	17	98769876	delhi
3	sonal	18	94769857	noida
6	Tanya	16	NULL	delhi

7 rows in set (0.00 sec)

Example 2: Let's delete all data from the student table. To do this, we need to give a condition that matches all the records. As IDs are greater than 0, so let's delete all those records where ID is greater than 0. Note: The same can be done using the truncate command. ***truncate student;***

```
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 operate on a list of values and return 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 the 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 the max, min, sum, avg, and count of unique values.

Example: Let's find out the total number of cities from where students came to study. Here more than one student is from the same city. So, we need to use distinct keyword along with the 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 the select statement.
- It is applied only to numeric values.

- It can't be applied with distinct keyword.

Query: group by <column name>

Example 1: Let's count the number of students of having same age in the 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)



aliasing

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 the select statement.

Note: To filter the result set of the 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 the first table and 4 rows in the second table then the total number of rows in the Cartesian product of these two tables will be 20 rows.

Cardinality of the final table of Cartesian product = cardinality of the first table * cardinality of the second table.

Example: we have two tables 'student' and 'awards'. Let's apply the Cartesian product to these two tables. (Refer page No. 180 for student & awards table)

```
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	3	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, the final result will consist of a common column from both 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
3	Mahesh	17	68769876	Jaipur	3	silver
						bronze

4 rows in set (0.06 sec)

Here both the tables have common column IDs. So, to avoid ambiguity (confusion), we need to mention the table name before the 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, the final result will consist of a common column only once.

Example: we have two tables' - students 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 have common column IDs. But there is no ambiguity arises on the name of the 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)
```