SHOW DATABASES
This displays information of all the existing databases in the server.

USE database_name
database_name : name of the database
This sets the database as the current database in the MySQL server.
To display the current database name which is set, use syntax:
SELECT DATABASE();

DESCRIBE table_name
table_name : name of the table
This describes the columns of the *table_name* with respect to Field, Type, Null, Key, Default, Extra.

SHOW TABLES
This shows all the tables in the selected database as a information.

SHOW CREATE TABLE table_name
table_name : name of the table
This shows the complete CREATE TABLE statement used by MySQL for creating the table.

SELECT NOW()
MySQL queries mostly starts with SELECT statement.
This query shows the current date and time

Output: 2020-02-20 07:08:30

SELECT 2+4, CURDATE();
Output: 6 | 2020-02-20

Comments
Comments are of two types. Multi-line comments or single-line or end-of-line comment.

```
/*  These are multi-line comments.  */

#  This is single-line comment.

-- This is also single-line comment.
```

CREATE DATABASE database_name
database_name : name of the database
This statement creates a new database.

DROP DATABASE database_name
database_name : name of the database
This statement deletes the database.
Note : User has to be very careful before deleting a database as it will loose all the crucial information stored in the database.

CREATE TABLE table_name(column1, column2, column3..)
table_name : name of the table
column1 : name of first column
column2 : name of second column
column3 : name of third column
When the developer start building an application, he needs to create database tables.
This statement creates a new table with the given columns.
Example :

```
CREATE TABLE  employee(
  'id' INTEGER  NOT NULL AUTO_INCREMENT,
  'name' VARCHAR(30) NOT NULL,
  'profile' VARCHAR(40) DEFAULT 'engineer',
  PRIMARY KEY ('id')
)ENGINE = InnoDB;
```

Note : You have *'id'* column as AUTO_INCREMENT with a primary key constraint which ensures that each id is incremented value, avoiding duplication. Storage engine selected is *'InnoDB'* allowing foreign key constraint and related transactions.

AUTO_INCREMENT
It is used to generate a unique identification field for new row.

DROP TABLE table_name
table_name : name of the table
This statement deletes the mentioned table.

RENAME TABLE old_table_name TO new_table_name
old_table_name : name of the previous table.
new_table_name : name of the new table.
This statement renames the table to a new name.

ALTER TABLE table_name ADD(column1, column2, column3..)
table_name : name of the existing table.
column1 : name of first column.
column2 : name of second column.
column3 : name of third column.
This statement adds columns to the existing table.

ALTER TABLE table_name DROP(column1)
table_name : name of the existing table.
column1 : name of first column.
This statement deletes specified columns from the existing table.

INSERT INTO table_name (column1, column2, column3..) VALUES(value1, value2, value3 ..)
table_name : name of the existing table.
column1 : name of first column.
column2 : name of second column.
column3 : name of third column.
value1 : value for first column.
value2 : value for second column.
value3 : value for third column.
This statement inserts a new record into a table with specified values.

UPDATE table_name SET column1 = value1, column2 = value2, column3 = value3.. WHERE condition
table_name : name of the table.
column1 : name of first column.
column2 : name of second column.
column3 : name of third column.
value1 : value for first column.
value2 : value for second column.
value3 : value for third column.
condition : the condition statement.
This statement updates records in the table with the new given values for the columns.
Note : WHERE clause in MySQL queries is used to filter rows for a specific condition.

DELETE FROM table_name WHERE condition
table_name : name of the table.
condition : the condition statement.
This statement deletes records from the table.

SELECT column1, column2, column3.. FROM table_name WHERE condition
table_name : name of the table.
column1 : name of first column.
column2 : name of second column.
column3 : name of third column.
condition : the condition statement.
This statement executes and gives records from specific columns from the table which matches the condition after WHERE clause.

SELECT * FROM table_name
table_name : name of the table.
Instead of specifying one column or many columns, you can use an asterisk (*) which represents all columns of table. This query retrieves all records from the table
---

COUNT
The COUNT function is used to return total number of records matching a condition from any table.
It is one of the known AGGREGATE function.
Example :

```
SELECT COUNT(*) from student;
```

Note : AGGREGATE functions allow you to run calculations on data and provide information by using a SELECT query.

MAX
It is used to get the maximum numeric value of a particular column of table.
Example :

```
SELECT MAX(marks) FROM student_report;
```

Note : The MIN and MAX functions work correctly on numeric as well as alphabetic values.

MIN
It is used to get the minimum numeric value of a particular column of table.
Example :

```
SELECT MIN(marks) FROM student_report;
```

Note : The above given example queries can also be nested with each other depending on the requirement.
Example :

```
SELECT MIN(marks)

FROM student_report

WHERE marks > ( SELECT MIN(marks) from student_report);
```

LIMIT
It is used to set the limit of number of records in result set.
Example :

```
SELECT *

FROM student limit 4, 10;
```

This gives 10 records starting from the 5th record.

BETWEEN
It is used to get records from the specified lower limit to upper limit.
This verifies if a value lies within that given range.
Example :

```
SELECT * FROM employee

WHERE age BETWEEN 25 to 45.
```

DISTINCT
This is used to fetch all distinct records avoiding all duplicate ones.
Example :

```
SELECT DISTINCT profile

FROM employee;
```

IN clause
This verfies if a row is contained in a set of given values.
It is used instead of using so many OR clause in a query.
Example :

```
SELECT *
FROM employee
WHERE age IN(40, 50, 55);
```

AND
This condition in MySQL queries are used to filter the result data based on AND conditions.
Example :

```
SELECT NAME, AGE
FROM student
WHERE marks > 95 AND grade = 7;
```

OR
This condition in MySQL queries are used to filter the result data based on OR conditions.
Example :

```
SELECT *
FROM student
WHERE address = 'Hyderabad' OR address = 'Bangalore';
```

IS NULL
This keyword is used for boolean comparison or to check if the data value of a column is null.
Example :

```
1. SELECT *
2. FROM employee
WHERE contact_number IS NULL;
```

FOREIGN KEY
It is used for pointing a PRIMARY KEY of another table.
Example :

```
CREATE TABLE Customers
(
id INT AUTO_INCREMENT PRIMARY KEY,
name VARCHAR(30) NOT NULL,
)
CREATE TABLE Orders
(
order_id INT AUTO_INCREMENT PRIMARY KEY,
FOREIGN KEY (id) REFERENCES Customers(id)
);
```

Note : This is not used in the MYISAM storage engine of MySQL server.
InnoDB storage engines supports foreign key constraints.

LIKE
This is used to fetch records matching for specified string pattern.

Example :

```
SELECT *
FROM employee
WHERE name LIKE 'Sh%';


SELECT *
FROM employee
WHERE name LIKE '%Sh%';
```

Note : Percentage signs (%) in the query represent zero or more characters.

JOINS
Joins are the joining of two or more database tables to fetch data based on a common field.
There are various types of joins with different names in different databases.
Commonly known joins are self join, outer join, inner join and many more.

Regular Join :
It is the join which gets all the records from both the tables which exactly match the given condition.
Example :

```
SELECT student.name, department.name

FROM student JOIN department ON student.department = department.name
```

Left Join :
It is the join which gets all the records that match the given condition, and also fetch all the records from
the left table.
Example :

```
SELECT student.name, department.name

FROM student LEFT JOIN department ON student.deptartment = department.name
```

Right Join :
It is the join which gets all the records that match the given condition, and also fetch all the records from
the right table.
Example :

```
SELECT student.name, department.name

FROM student RIGHT JOIN department on student.department = department.name
```

ADD or DROP a column
A new column can be added on a database table, if required later on.
Example :

```
ALTER TABLE employee ADD COLUMN salary VARCHAR(25);
```

Similarly, any column can be deleted from a database table.

Example :

```
ALTER TABLE employee DROP COLUMN salary;
```