Chapter- 07

STRUCTURED QUERY LANGUAGE

Introduction To SQL And MySQL:

- MySQL is a freely available open-source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL).
- MySQL can be downloaded from site www.mysql.org. MySQL is created and supported by MySQLAB, a company based in Sweden.
- In MySQL database, information is stored in Tables. A single MySQL database can contain many tables at once and store thousands of individual records.
- MySQL provides you with a rich set of features that support a secure environment for storing, maintaining, and accessing data.

Structured Query Language (SQL):

In order to access data within the MySQL database, all programmers and users must use, Structured Query Language (SQL). SQL is the set of commands that is recognized by all RDBMS. The Structured Query Language (SQL) is a language that enables you to create and operate on relational database, which are sets of related information stored in tables.

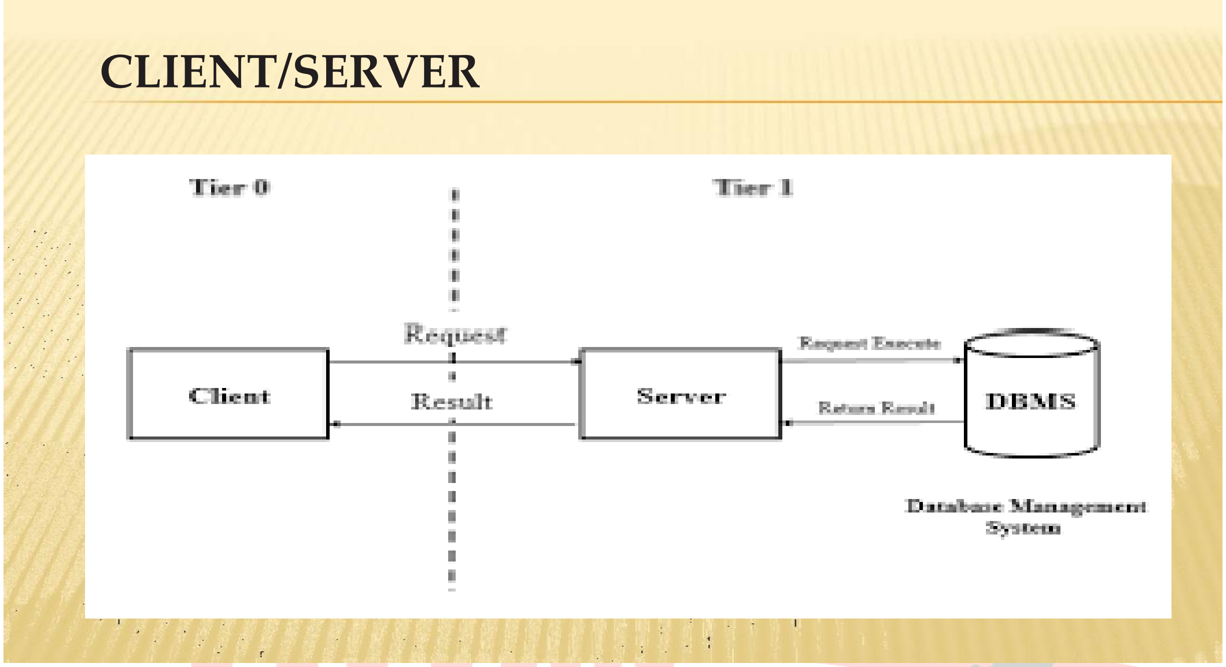
MySql Database System:

MySQL Database System is a combination of a MySQL server instance and a MySQL database. MySQL database system operates using client/server architecture, in which the server runs on the machine containing the databases and clients connect to the server over a network.

SQL Server And Clients:

MySQL Server:

It listens for client request coming in over the network, Accesses database contents according to those requests and Provides contents to the clients. MySQL is compatible with the standards-based SQL. The client program may contact the server programmatically or manually. MySQL clients are programs that connect to the MySQL server and issue queries in a pre-specified format.



CLIENT/SERVER

Features And Advantages Of MySQL:

Features Of MySQL:

- Speed: If the server hardware is optimal, MySQL runs very fast.
- Cost: Available free of cost.
- Portability: Provides portability as it has been tested with a broad range of different compiler and can work on many different platforms.
- Data Types: Provide many data types to support different types of data.
- Security: Offers a privilege and password system that is very flexible and secure.

Advantages Of MySQL:

1. Reliability and performance: MySQL is very reliable and high performance relational database management system.

2. Availability of source: MySQL source code is available that is why now we can recompile the source code.
3. Cross-Platform support: MySQL supports more than twenty different platforms including the major Linux.

Starting Of MySQL:

To start MySQL make sure that MySQL Server is installed on your machines. Once it is installed, you need to click at

Start

All Programs

MYSQL

MySQL Server

MySQL Command Line Client

It will start the MySQL client where you have to specify the password before start working. We can quit from MySQL by typing **Quit** at the **mysql> prompt**.

Processing Capabilities Of SQL:
The various processing capabilities of SQL are:

- Data Definition Language (DDL):
- Interactive Data Manipulation Language (DML)
- Transaction control language (TCL)
- Embedded Data Manipulation Language
- View Definition

- Authorization
- Integrity

Classification Of SQL Statements:

SQL provides many different types of commands used for different purposes. These commands can be divided into following categories:

1. Data Definition Language (DDL)commands
2. Data Manipulation Language (DML)commands
3. Transaction Control Language (TCL)commands
4. Session Control commands
5. System Control commands.

Data Definition Language (DDL) Commands:

DDL commands allow us to perform tasks related to data definition. One can perform the following tasks.

- It should identify the data item, segment, record, and data-base file.
- It should give a unique name to each data- item, record, file and database.
- It should specify the proper datatypes.
- Some DDL commands are: **Create, Alter, Drop**

Data Manipulation Language (DML) Commands:

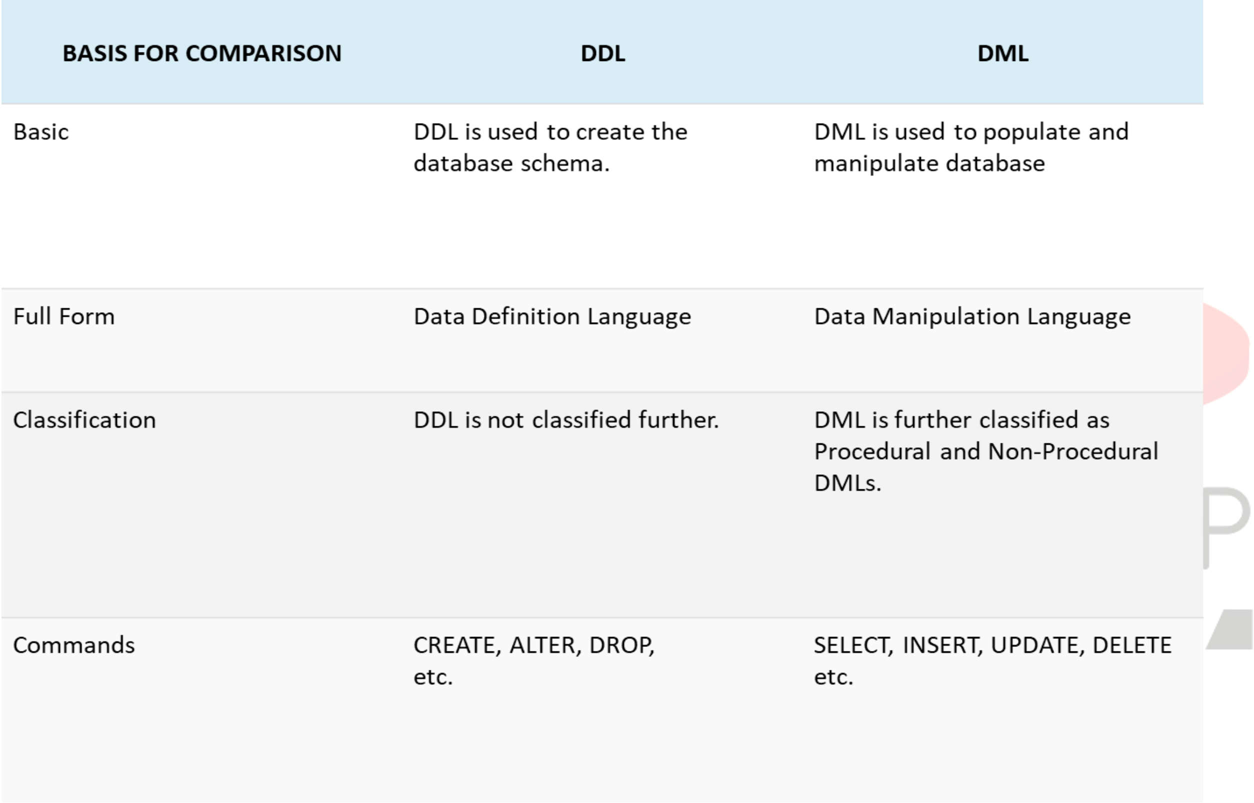
DML includes a set of commands that enables user to access or manipulate data. They do the following types of operations.

- Retrieval of information stored in database
- Insertion of new information into database

- Deletion of information from database
- Modification of data stored in database

Some DML commands are: **INSERT, UPDATE and DELETE**

Difference between DDL and DML commands



BASIS FOR COMPARISON	DDL	DML
Basic	DDL is used to create the database schema.	DML is used to populate and manipulate database
Full Form	Data Definition Language	Data Manipulation Language
Classification	DDL is not classified further.	DML is further classified as Procedural and Non-Procedural DMLs.
Commands	CREATE, ALTER, DROP, etc.	SELECT, INSERT, UPDATE, DELETE etc.

Types Of DMLs:

Procedural DML: These require a user to specify what data is needed and how to get it.

Non-Procedural DML: These require a user to specify what data is needed without specifying how to get it.

Transaction Control Language (TCL) Commands:

These commands are used to manage and control the transactions. These commands manage changes made by DML commands. Some TCL commands are as following:

- COMMIT
- ROLLBACK
- SAVEPOINT
- SET TRASACTIONS

Some Mysql SQL Elements:

Some basic elements that play an important role in defining/querying a database are:

- Literals
- Datatypes
- Nulls
- Comments

Literals:

- Literals are fixed data values.
- A fixed data value may be of character type or numeric literal.
- All character literals are enclosed in single quotation marks or double quotation marks e.g., 'Synthiya', 'Ronak Raj Singh‘,‘8’.
- Numbers that are not enclosed in quotation marks are numeric literals e.g., 22, 18,1997.
- Numeric literals can either be integer literals or be real literals e.g., 17 is an integer literal but 17.0 and 17 .5 are real literals.

Data Types:

MySQL uses many different data types, divided into three categories:

(A) Numeric
(B) Date and time
(C) String types

Various SQL Commands and Functions:

Database commands:

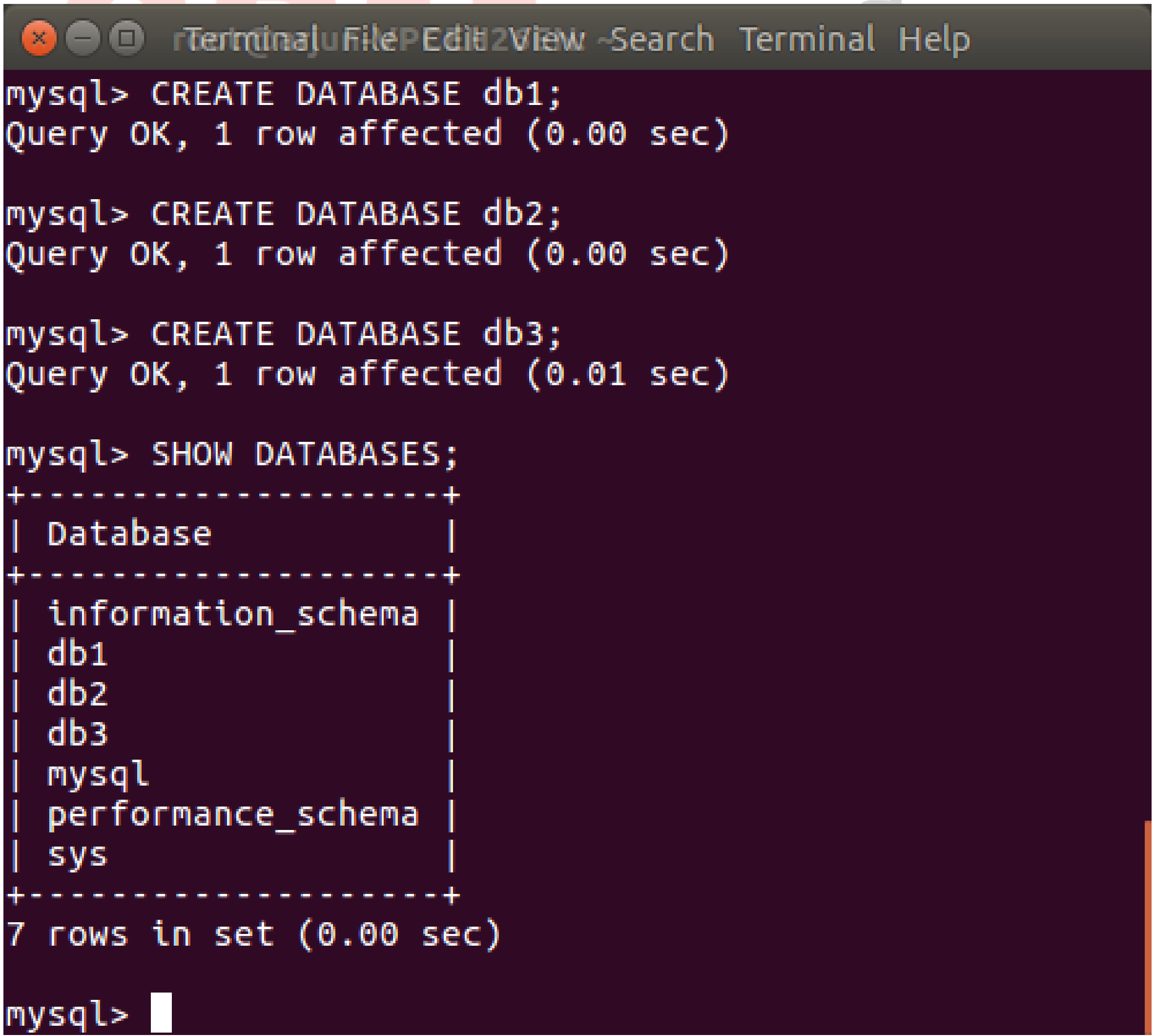
- CREATE DATABASE Command
- Use command
- Show databases command
- Drop database command

Show Databases:

- **SHOW DATABASES** lists the **databases** on the MySQL server.

  To see all the databases created write the command

  Show databases;



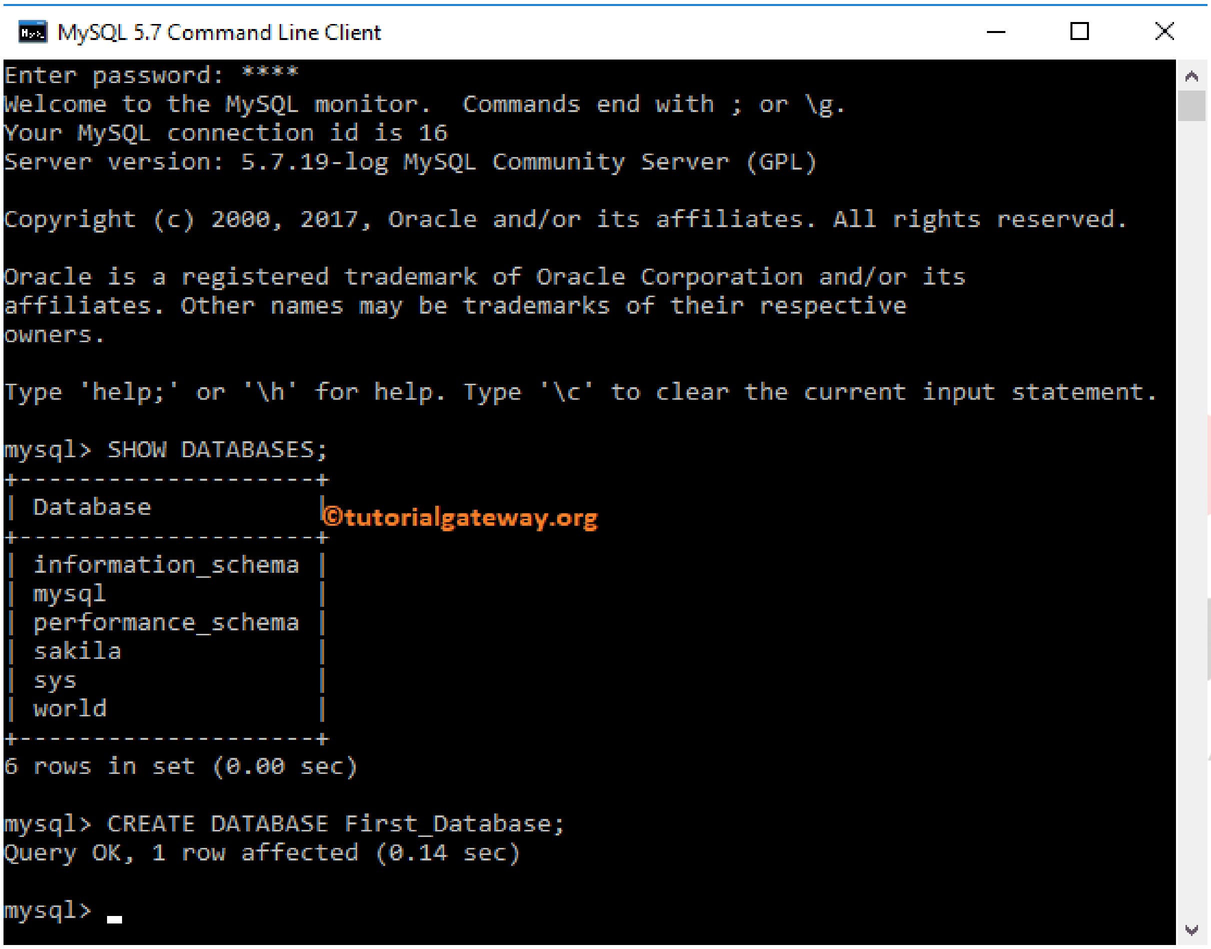
```
mysql> CREATE DATABASE db1;
Query OK, 1 row affected (0.00 sec)

mysql> CREATE DATABASE db2;
Query OK, 1 row affected (0.00 sec)

mysql> CREATE DATABASE db3;
Query OK, 1 row affected (0.01 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| db1                |
| db2                |
| db3                |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
7 rows in set (0.00 sec)

mysql> 
```

CREATE DATABASE Command:

This command is used to create a database.

- To create a database named school the command is

  Create database school;



```
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 16
Server version: 5.7.19-log MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| world              |
+--------------------+
6 rows in set (0.00 sec)

mysql> CREATE DATABASE First_Database;
Query OK, 1 row affected (0.14 sec)

mysql> _
```

Accessing Database:

Use command:

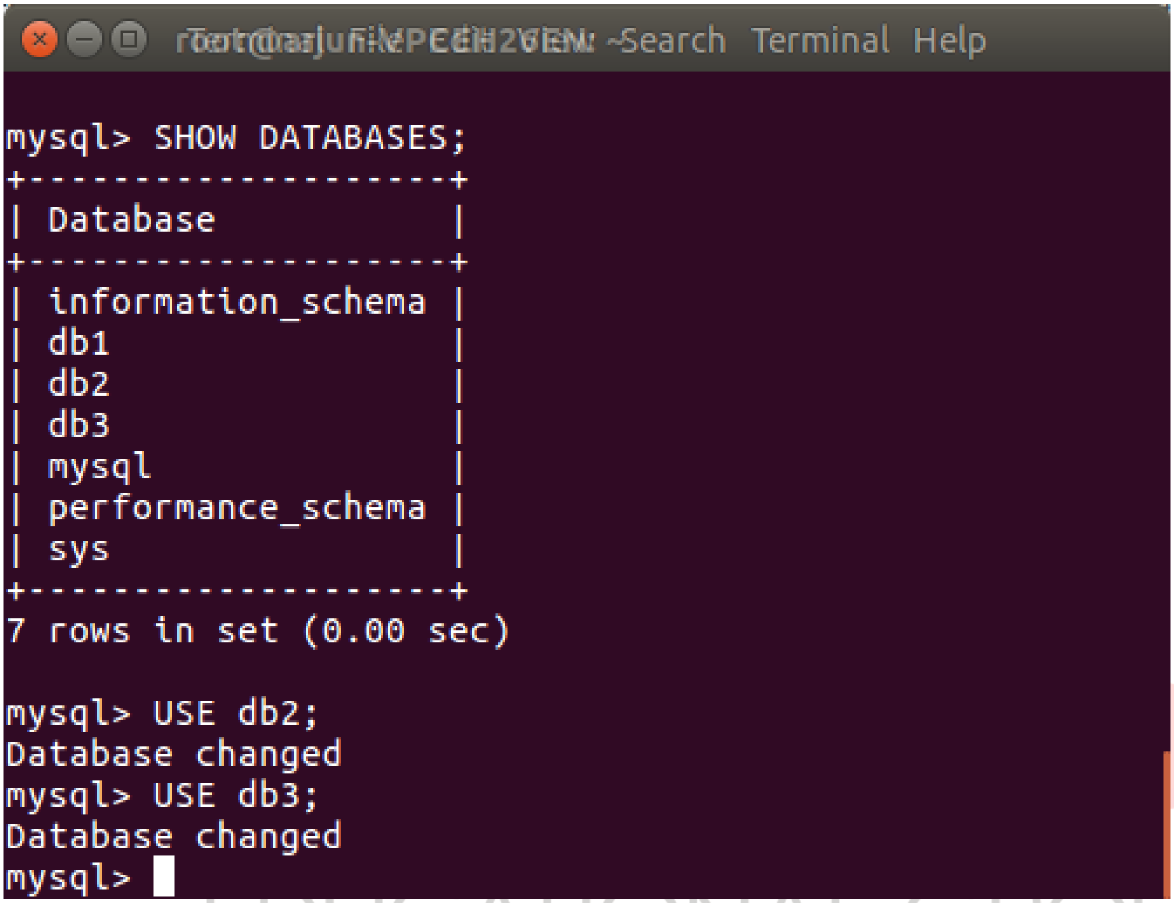
Before doing any operation upon the table in the database, we need to open the database.

Syntax:

Use database name;

Ex: To work with the database school the command is

Use school;



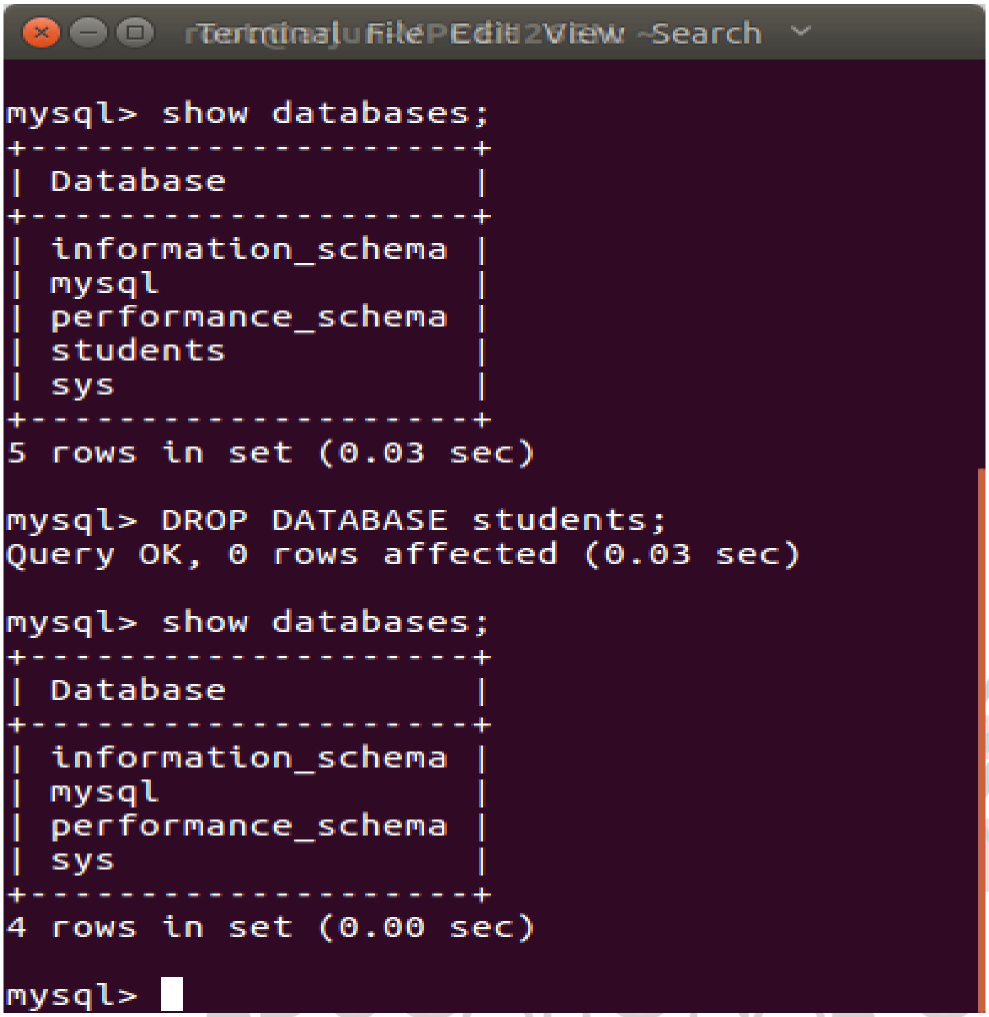
```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| db1                |
| db2                |
| db3                |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
7 rows in set (0.00 sec)

mysql> USE db2;
Database changed
mysql> USE db3;
Database changed
mysql> 
```

Drop database command:

- The **DROP DATABASE** Statement is used to **drop** or **delete** a **database**.
- **Dropping** the **database** will **drop** all **database** objects (tables, views, procedures etc.) inside it.

Ex: To delete the database school

Drop database school;



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| students           |
| sys                |
+--------------------+
5 rows in set (0.03 sec)

mysql> DROP DATABASE students;
Query OK, 0 rows affected (0.03 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql> 
```

CREATE TABLE Command:

This command is used to create a table in the database.

- To create an *employee* table whose scheme is as follows :

Ecode	Ename	Sex	Grade	Gross

- Employee (ecode, ename, sex, grade, gross)

```
CREATE TABLE Employee    (Ecode
          integer,
          Ename         char(20),
          Sex          char(1),
          Grade        char(2),
          Gross        decimal);
```

Create a *student* table whose scheme is as follows :

Student(roll, sname, sex, grade, dob, phoneno)

```
                CREATE TABLE student
                      (   roll          integer, sname
                          char(20),
                          Sex           char(1),
                          Grade         char(2),
                          Dob           date,
                          Phoneno       integer ) ;
```

Constraints In Create Table Commands:

Constraint:

A Constraint is a condition or check applicable on a field or set of fields.

Types of Constraints:

- Unique constraint
- Primary key constraint
- Default constraint
- Check constraint
- Not null

Unique Constraint:

The **UNIQUE constraint** maintains the **uniqueness** of a column in a table. More than one **UNIQUE** column can be used in a table.

```
        CREATE TABLE employee
                ( ecode      integer, ename  char(20),
                  sex          char (1),
                  grade        char (2),
                  gross        decimal      UNIQUE );
```

Not Null Constraint:

- The **NOT NULL constraint** enforces a column to **NOT** accept **NULL** values.
- The **NOT NULL constraint** enforces a field to always contain a value.

```
CREATE TABLE employee

(  ecode     integer     NOT NULL PRIMARY KEY ,
   ename     char (20)   NOT NULL,

   sex       char (1)    NUT NULL,

   grade     char (2),

   gross     decimal ) ;
```

Primary Key Constraint:

This constraint declares a column as the **primary key** of the table. The primary keys cannot allow NULL values.

```
CREATE TABLE employee

       (  ecode     integer     NOT NULL PRIMARY EY , ename
                    char (20)   NOT NULL,

          sex       char (1)    NUT NULL,

          grade       char (2),

          gross       decimal ) ;
```

Default Constraint:

A default value can be specified for a column using the DEFAULT clause. When a user does not enter a value for the column automatically the defined default value is assigned.

```
CREATE TABLE employee

( ecode    integer   NOT NULL PRIMARY KEY,

ename       char (20) NOT NULL,

 sex       char (1) NOT NULL,

 grade  char (2)   DEFAULT = 'E1' ,

  gross decimal ) ;
```

Check Constraint:

This constraint limits values that can be inserted into a column of a table. For instance, consider the following SQL statement:

```
CREATE TABLE employee
 ( ecode    Integer     NOT NULL PRIMARY KEY,
   ename    char (20)   NOT NULL,
   sex      char (1)    NUT NULL,
   grade    char (2)    DEFAULT = 'E1',
   gross    Decimal     CHECK (gross = 2000) ) ;
```

Applying Table Constraints:

When a constraint is to be applied on a group of columns of the table, it is called ***table constraint***. The table constraints appear in the end of table definition.

```
CREATE        TABLE          OrderItem
( OrderNum    INTEGER          PRIMARY      KEY,
  ItemNum     INTEGER,
  Quantity    INTEGER,
  Price       INTEGER,
  NOT NULL (Quantity, Price) );
```

CREATE TABLE Example in MySQL

1.

```
CREATE TABLE STUDENTS
         (ID      INT      NOT NULL,
          NAME VARCHAR (20) NOT NULL,
          AGE INT NOT NULL,
          ADDRESS CHAR (25),
          PRIMARY KEY (ID));
```

2.

```
CREATE TABLE Employee

          (EmployeeID int,

           FirstName varchar(255),
           LastName varchar(255),

           Email varchar(255),

          AddressLine varchar(255),

          City varchar(255));
```

3.

```
CREATE TABLE Employee

        (EmployeeID NOT NULL,

         FirstName varchar(255) NOT NULL,

         LastName varchar(255),

         City varchar(255));
```

4.

```
CREATE TABLE Persons

        (ID int NOT NULL,

        LastName varchar(255) NOT NULL,

        FirstName varchar(255),

        Age int UNIQUE);
```

5.

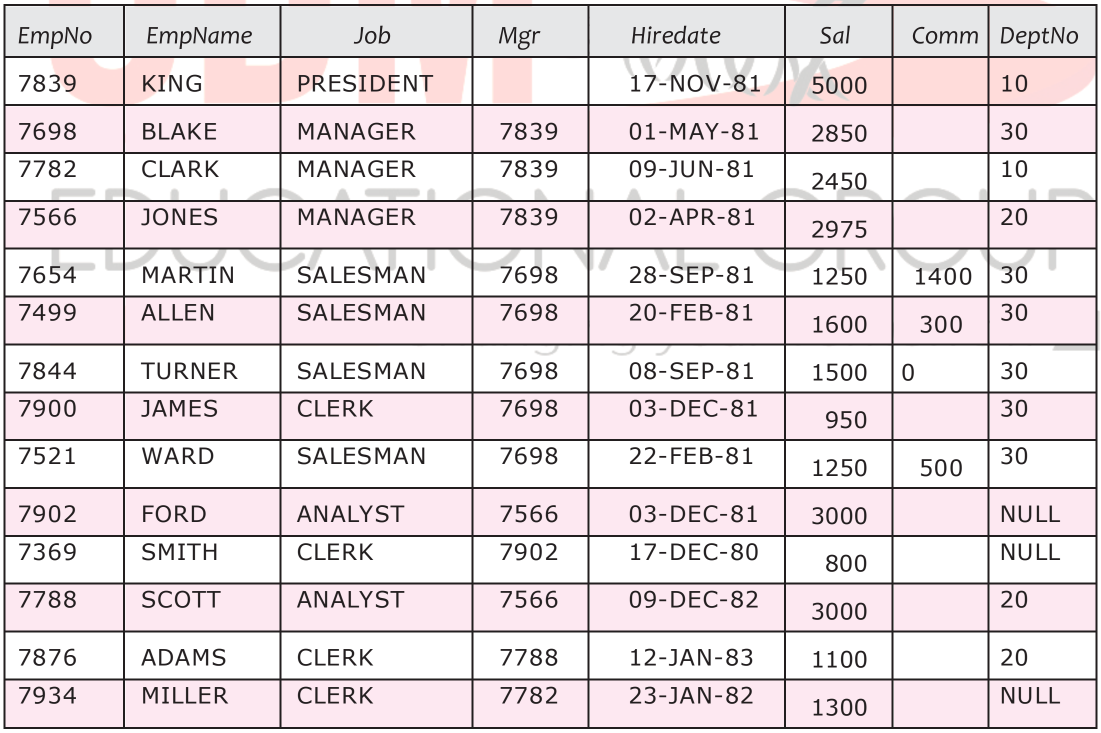
```
CREATE TABLE Persons

        (ID int NOT NULL,
        LastName varchar(255) NOT NULL,
        FirstName varchar(255),
        Age int);
```

The SELECT Command:

The SELECT command of SQL lets you make queries on the database. A query is a command that is given to produce certain specified information from the table.

In its simplest form, SELECT statement is used as :

SELECT <*column name*> [, <column name>,…] FROM <*table name*> ;



EmpNo	EmpName	Job	Mgr	Hiredate	Sal	Comm	DeptNo
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7902	FORD	ANALYST	7566	03-DEC-81	3000		NULL
7369	SMITH	CLERK	7902	17-DEC-80	800		NULL
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		NULL

Table – EMP

Examples:

- SELECT Empno, Empname FROM emp ;

Empno	*EmpName*
7839	*KING*
7698	*BLAKE*
7782	*CLARK*
:	:
:	:

SELECT * FROM emp;

This will display all the rows present in the emp table.

Where clause (used to add condition)

SELECT empname, sal FROM emp WHERE sal > 2900;

Empname	**Sal**
KING	5000
JONES	2975
FORD	3000
SCOTT	3000

Relational Operators:

To compare two values, a relational operator is used. The result of the comparison is true or false. The SQL recognizes following relational operators:

=, >, <, > =, < =, <> (not equal to)

To list all the members not from 'DELHI'

SELECT *FROM Suppliers WHERE city < > 'DELHI' ;

Logical Operators:

The logical operators OR, AND and NOT are used to combine multiple conditions in the WHERE clause.

For example, To list the employees' details working in deptno 10 or 20 from table emp.

SELECT * FROM employee WHERE (deptno= 10 OR deptno= 20) ;

EmpNo	*EmpName*	*Job*	*Mgr*	*Hiredate*	*Sal*	*Comm*	*DeptNo*
7839	KING	PRESIDENT		17-NOV-81	5000		10
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20

To list all the employees' details working as manager in department number 30.

SELECT * from emp WHERE (job= 'MANAGER' AND deptno=30) ;

To list all the employees' details whose department number are other than 30.

SELECT * FROM emp WHERE (NOT deptno= 30);

OR

EmpNo	EmpName	Job	Mgr	Hiredate	Sal	Comm	DeptNo
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30

EmpNo	EmpName	Job	Mgr	Hiredate	Sal	Comm	DeptNo
7839	KING	PRESIDENT		17-NOV-81	5000		10
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7902	FORD	ANALYST	7566	03-DEC-81	3000		NULL
7369	SMITH	CLERK	7902	17-DEC-80	800		NULL
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		NULL

Condition Based On A Range (Between):

The BETWEEN operator defines a range of values. The range includes both lower value and the upper value.

For example:

To list the items whose QOH falls between 30 to 50 (both inclusive)

SELECT icode, descp, QOH FROM items WHERE QOH BETWEEN 30 AND 50 ;

Icode	Descp	Price	QOH	ROL	ROQ
I01	Milk	15.00	20	10	20
I02	Cake	5.00	60	20	50
I03	Bread	9.00	40	10	40
I04	Biscuit	10.00	50	40	60
I05	Namkeen	15.00	100	50	70
I06	Cream Roll	7.00	10	20	30

NOT BETWEEN:

SELECT icode, descp, QOH FROM items WHERE QOH NOT BETWEEN 30 AND 100;

Icode	Descp	QOH
101	milk	20
106	Cream Roll	10

Reordering Columns in Query Results:

While giving a querying, the result can be obtained in any order. For example, if you give

SELECT job, empno, sal FROM emp;

The result will be having *job* as first column, *empno* as second column, and *sal* as third column. You can write the column names in any order and the output will be having information in exactly the same order.

Job	EmpNo	Sal
PRESIDENT	7839	5000
MANAGER	7698	2850
MANAGER	7782	2450
MANAGER	7566	2975
SALESMAN	7654	1250
SALESMAN	7499	1600
SALESMAN	7844	1500
CLERK	7900	950
SALESMAN	7521	1250
ANALYST	7902	3000
CLERK	7369	800

ANALYST	7788	3000
CLERK	7876	1100
CLERK	7934	1300

Eliminating Redundant Data (Keyword Distinct):

The DISTINCT keyword eliminates duplicate rows from the results of a SELECT statement.

For example, if we write a command as

SELECT job FROM emp;

OR

SELECT ALL job FROM emp;

It will display the entire job column with the duplicate values.

Job
PRESIDENT
MANAGER
SALESMAN
CLERK
ANALYST

So if u want to display only the unique values we have to write the command as:

SELECT DISTINCT job FROM emp;

Condition Based On A List (In/ Not In):

To specify a list of values, IN operator is used. The IN operator selects values that match any value in a given list of values.

To display the name and salary from Emp table working as clerk, analyst or manager.

SELECT EmpName, Sal FROM Emp WHERE Job IN (‘CLERK’, ‘ANALYST’, ‘MANAGER’) ;

OR

```
SELECT EmpName, Sal FROM Emp WHERE (Job =‘CLERK’) OR (Job = ‘ANALYST’ ) OR
Job=‘MANAGER’);
```

EmpName	Sal
BLAKE	2850
CLARK	2450
JONES	2975
JAMES	950
FORD	3000
SMITH	800
SCOTT	3000
ADAMS	1100
MILLER	1300

EmpName	*Sal*
KING	5000
MARTIN	1250
ALLEN	1600
TURNER	1500
WARD	1250

```
SELECT EmpName,Sal FROM Emp WHERE Job NOT IN (‘CLERK’, ‘ANALYST’,‘MANAGER’);
```

OR

```
SELECT EmpName,Sal FROM Emp WHERE Job <>‘CLERK’ OR Job <> ‘ANALYST’ OR Job<>‘MANAGER’ ;
```

Searching For Null (Is / Is Not):

The NULL value in a column can be searched using IS NULL in the WHERE clause. (Relational operators like =, <> etc. can’t be used with NULL).

For example, to list details of all employees whose comm contain NULL

```
SELECT * FROM emp WHERE comm IS NULL ;
```

EmpNo	EmpName	Job	Mgr	Hiredate	Sal	Comm	DeptNo
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30

7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		NULL
7369	SMITH	CLERK	7902	17-DEC-80	800		NULL
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		NULL

To list details of all employees whose departments doesn't contain NULL values.

SELECT *FROM empWHERE DeptNo IS NOT NULL ;

EmpNo		Job	Mgr	Hiredate	Sal	Comm	DeptNo
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20

Sorting Results (Order By Clause):

You can sort the results or a query in a specific order using ORDER BY clause. The ORDER BY clause allows sorting of query results by one or more columns. The sorting can be done either in *ascending* or *descending* order, the default order is ascending.

DESC for descending order and ASC for ascending order.

Example:

To display the list of employees in the alphabetical order of their names

SELECT *FROM employee ORDER BY ename;

OR

SELECT *FROM employee ORDER BY ename ASC;

To display the list of employees no, name and job having salary more than 2500 in the alphabetical order of their names:

SELECT empno, empname, job FROM empWHERE sal> 2500 ORDER BY ename ;

To display the list of employees in the descending order of employee code, you use the command:

SELECT *FROM empORDER BY ecode DESC;

How To Perform Simple Calculation?

Simple calculations can be done via a SELECT command. SQL provides a dummy table called ***Dual*** which has just one row and one column. It can be used for obtaining calculation results and also system date.

The following query:

SELECT 4 * 3 FROM dual;

will produce the result:

```
4 * 3
- - - - -
 12
```

The current date can be obtained from the ***Dual*** table using ***sys_date***, as shown below:

SELECT sysdate FROM dual ;

The output produced by above query will show the current date:

Aggregate Functions:

The following Aggregate functions can be applied to the entire table or to specific rows by a WHERE clause.

SUM()

MAX()

MIN()

COUNT( )

COUNT(*)

To calculate the total gross for employees of grade 'E2' :

```
SELECT sum(gross) FROM employee WHERE grade = 'E2' ;
```

To display the average gross of employees with grades 'E1' or 'E2' :

```
SELECT avg(gross) FROM employee WHERE (grade = 'E1' OR grade = 'E2' ) ;
```

To count the number of employees in *employee* table, the *SQL :*

```
SELECT count(*) FROM employee ;
```

To count the number of cities, the different members belong to:

```
SELECT count(DISTINCT city) FROM members ;
```

Here the **DISTINCT** keyword ensures that multiple entries of the same *city* are ignored.

The * is the only argument that includes **NULL**s when it is used only with

COUNT, functions other than **COUNT** disregard **NULL**s in any case.

INo	*Iname*	*Price*	*SNo*
T01	Mother Board	12000	S01
T02	Hard Disk	5000	S01
T03	Keyboard	500	S02
T04	Mouse	300	S01

T05	Mother Board	13000	S02
T06	Key Board	400	S03
T07	LCD	6000	S04
T08	LCD	5500	S05
T09	Mouse	350	S05
T10	Hard Disk	4500	S03

- To display the total price of all the items.
- To display the total price of all the LCDs.
- To list the average price of all the items whose sno is s01.
- To display the maximum price among the mouse.
- Count the no. of items.
- Count the no. of mouse.
- Count the different types of items from item table.

```
Select SUM(price) FROM ITEM;

Select SUM(price) FROM ITEM WHERE Iname = ‘LCD’;

Select AVG(price) FROM ITEM WHERE Sno = ‘S01’;
Select MAX(price) FROM ITEM WHERE Iname = ‘Mouse’;
Select COUNT(*)FROM ITEM;
Select COUNT(Mouse)
```

Grouping Result: Group By

The GROUP BY clause is used in SELECT statements to divide the table into groups.

Grouping can be done by a column name, or with aggregate functions in which case the aggregate produces a value for each group.

Example:

To display the job, number of employees in each job and total comm for each job of employees:

SELECT job, count(*), sum(comm) FROM emp GROUP BY job ;

Job	Count(*)	Sum(comm)
PRESIDENT	1	0
MANAGER	3	0
SALESMAN	4	2200
CLERK	4	0
ANALYST	2	0

Placing Conditions On Groups (Having Clause):

The HAVING clause places conditions on groups in contrast to WHERE clause that places conditions on individual rows WHERE conditions cannot include aggregate functions, HAVING conditions can do so.

Example:

To calculate the average gross and total gross for employees belonging to 'E4' grade, the command would be:

SELECT avg(gross), sum(gross) FROM employee GROUP BY grade HAVING grade = 'E4' ;

To display the jobs where the number of employee are less than 3:

SELECT job, count(*) FROM emp GROUP BY job HAVING count(*) < 3 ;

Scalar Expressions With Selected Fields:

We can perform simple numeric computations on the data to put it in a form as per our need.

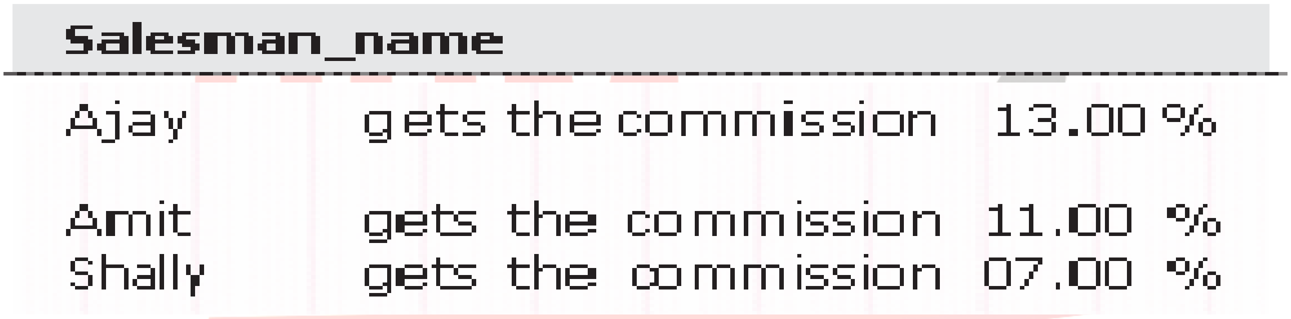
SELECT Iname, Price+100 FROM ITEM;

SELECT Iname, Price*2 FROM ITEM

WHERE Iname = 'Mouse';

Putting Text In The Query Output :

SELECT salesman_name, 'gets the commission', comm*100, '%' FROM salesman;



```
Salesman_name
----------------------------------------------------------------
Ajay        gets the commission  13.00 %

Amit        gets the commission  11.00 %
Shally      gets the commission  07.00 %
```

The Insert Command:

The rows (tuples) are added to relations using INSERT command of *SQL*.

For example, to enter a row into *employee* table (defined earlier), you could use the following statement:

INSERT INTO employee VALUES (1001, 'Ravi', 'M', 'E4', 4670.00) ;

The same can be done with an alternate command as shown below:

INSERT INTO employee (ecode, ename, sex, grade, gross) VALUES (1001, 'Ravi', 'M', 'E4', 4670.00) ;

For instance, if you want to insert only *ecode*, *ename*and *sex* columns, you use the command:

INSERT INTO employee (ecode, ename, sex) VALUES (2014, 'Manju', 'F') ;

The columns that are not listed in the INSERT command will have their default value, if it is defined for them, otherwise, NULL value.

Inserting The Results Of A Query:

INSERT command can also be used to take or derive values from one table and place them in another by using it with a query.

INSERT INTO branch1 (SELECT *FROM branch 2 WHERE gross >7000);

It will extract all those rows from branch2 that have gross more than 7000.00 and insert this result into the table branch1.

The Delete Command:

The DELETE command removes rows from a table.

This removes the entire rows, not individual field values.

To remove all the contents of *items* table

DELETE FROM *items* ;

To remove the tuples from *employee* that have *gross* less than 2200 :

DELETE FROM employee WHERE gross <2200.00 ;

The UPDATE COMMAND (UPDATE + SET)

Update command is used to change some or all of the values in an existing row.

To change the *ROL* of all items to 250:

UPDATE items SET ROL = 250 ;

To change *ROL* to 400 only for those items that have *ROL* as 300:

UPDATE items SET ROL = 400 WHERE ROL = 300;

UPDATING MULTIPLECOLUMNS

To update multiple columns, multiple column assignments can be specified with SET clause, separated by commas.

To update the ***ROL to 400***and ***QOH to 700***for items having **icode lessthan 'I040'**, we shall write

```
UPDATE items SET ROL = 400, QOH = 700 WHERE icode<'I040' ;
```

USING EXPRESSIONS IN UPDATE

if you want to increase the gross pay of all the employees by Rs. 900/-:

```
UPDATE employee SET gross = gross + 900;
```

To double the gross pay of employees of grade 'E3' and 'E4':

```
UPDATE employee SET gross = gross * 2

WHERE (grade = 'E3' OR grade = 'E4') ;
```

THE CREATE VIEW COMMAND

```
CREATE VIEW taxpayee AS ( SELECT * FROM employee WHERE gross >8000 );
```

SOME BUILT-IN FUNCTIONS

SELECT lower ("HELLOW") FROMDual;	**hello**
SELECT upper ("friends") FROMDual;	**FRIENDS**
SELECT replicate ("*#", 4) FROMDual;	***#*#*#*#**
SELECTsubstr ("Pointer", 3, 2) FROMDual;	**in**
SELECT getdate()FROMDual;	**12-07-2020**

will return the current system date of your computer.

THE ALTER TABLE COMMAND(ALTER+ADD /ALTER+MODIFY)

ALTER+ADD

It adds new column to the existing table.

To add a new column *tel_number*of type *integer* in table *Emp*:

```
ALTER TABLE Emp

ADD (tel_number integer) ;
```

ALTER+MODIFY

To modify existing columns of table, ALTER TABLE command can be used.

To modify column *Job* of table *Emp*to have new width of 30 characters:

ALTER TABLE Emp MODIFY (Job char(30)) ;

THE DROP TABLE COMMAND

The DROP TABLE command of *SQL* lets you drop or delete a table from the database.

The *SQL* requires you to empty a table before you eliminate from the database.

To remove all the rows from your table:

DELETE FROM items ;

Then you can drop the empty table *items* as follows :

DROP TABLE items ;

The Drop View Command:

To delete a view from the database the DROP VIEW command is used.

For example

DROP VIEW taxpayee;

When a view is dropped, it does not cause any change in its base table.

After the removal of view *taxpayee*, its base table *employee* remains intact.

JOIN:

A join is a query that combines rows from two or more tables.

The function of combining data from multiple tables is called *joining*. Join is off following two types.

1.Equi join

2. Natural join

The Joi in which columns are compared for equality is called **Equi- Join**.

The Join in which only one of the identical columns (coming from joined tables) exists, is called **Natural Join**.

Write SQL queries for (i) to (iv) and find outputs for SQL queries (v) to (viii), which are based on the tables.

VEHICLE

CODE	VTYPE	PERKM
101	VOLVO BUS	160
102	AC DELUXE BUS	150
103	ORDINARY BUS	90
105	SUV	40
104	CAR	20

NO	NAME	TDATE	KM	CODE	NOP
101	Janish Kin	2015-11-13	200	101	32
103	VedikaSahai	2016-04-21	100	103	45
105	Tarun Ram	2016-03-23	350	102	42
102	John Fen	2016-02-13	90	102	40
107	Ahmed Khan	2015-01-10	75	104	2
104	Raveena	2016-05-28	80	105	4
106	Kripal Anaya	2016-02-06	200	101	25

travel

- To display NO, NAME TDATE from the table TRAVEL in descending order ofNO.

- To display the NAME of all the travellers from the table TRAVEL who are travelling by vehicle with code 101 or102.
- To display the NO and NAME of those travellers from the table TRAVEL who travelled between ‘2015-12-31’ and‘2015-04-01’.
- To display all the details from table TRAVEL for the travellers, who have travelled distance more than 100 KM in ascending order of NOP.
- SELECT COUNT(*), CODE FROM TRAVEL GROUP BY CODE HAVING COUNT(*)>1;
- SELECT DISTINCT CODE FROM TRAVEL;
- SELECT A.CODE, NAME, VTYPE FROM TRAVEL A, VEHICLE B WHERE A.CODE = B.CODE AND KM < 90;
- SELECT NAME, KM * PERKM FROM TRAVEL A, VEHICLE B WHERE A.CODE = B.CODE AND A.CODE = ‘105’;

Consider the following tables EMPLOYEE and SALGRADE and answer (A1) and (A2) parts of this question:

Table :EMPLOYEE

ECODE	NAME	DESIG	SGRADE	DOJ	DOB
101	Abdul Ahmad	EXECUTIVE	S03	23-Mar-2003	13-Jan-1980
102	Ravi Chander	HEAD-IT	S02	12-Feb-2010	22-Jul-1987
103	John Ken	RECEPTIONIST	S03	24-Jun-2009	24-Feb-1983
105	Nazar Ameen	GM	S02	11-Aug-2006	03-Mar-1984
108	Priyam Sen	CEO	S01	29-Dec-2004	19-Jan-1982

Table :SALGRADE

SGRADE	SALARY	HRA
S01	56000	18000
S02	32000	12000
S03	24000	8000

(A1) write commands for the followings:

- To display the details of all EMPLOYEEs in descending order of DOJ.
- To display NAME and DESIG of those EMPLOYEEs, whose SALGRADE is either S02 orS03.
- To display the content of all the EMPLOYEEs table, whose DOJ is in between ‘09-Feb-2006’ and‘08-Aug-2009’.
- To add a new row with the following: 19, ‘Harish Roy’, ‘HEAD-IT’, ‘S02’, ‘09-Sep-2007’,‘21-Apr-1983’

(A2) write outputs:

- Select * from employee order by dojdesc;
- Select name, desig from employee where salgrade in(‘s02’, s03’);
- Select * from employee where doj between ‘09-feb-2006’ and ‘08-aug- 2009’;
- Insert into employee values(19,‘harish roy’,‘head-it’,‘s02’,‘09-sep- 2007’,‘21-apr-1983’);
- **SELECT COUNT (SGRADE), SGRADE FROMEMPLOYEE**

  GROUP BY SGRADE;
- **SELECT MIN(DOB), MAX(DOJ) FROMEMPLOYEE;**
- **SELECT NAME,SALARY**

  FROM EMPLOYEE E, SALGRADE S

  WHERE E.SGRADE = S.SGRADE AND E.ECODE<103;
- **SELECT SGRADE, SALARY + HRA FROMSALGRADE**

  WHERE SGRADE = ‘S02’;

```
COUNT       SGRADE                              13-Jan-1980        12-Feb-2010

2           503

2           502

1           501
```

NAME	SALARY
Abdul Ahmad	24000
Ravi Chander	32000

SGRADE	SALARY + HRA
502	44000