Database Systems Laboratory

B.Tech. 5thSemester

Name : Deepak R

Roll Number : 18ETCS002041

Department : Computer Science and Engineering

Faculty of Engineering & Technology

Ramaiah University of Applied Sciences

Ramaiah University of Applied Sciences

Private University Established in Karnataka State by Act No. 15 of 2013

Faculty	Engineering & Technology
Programme	B. Tech. in Computer Science and Engineering
Year/Semester	3rd Year / 5th Semester
Name of the Laboratory	Database Systems Laboratory
Laboratory Code	19CSL306A

List of Experiments

1. DDL and DML commands
2. Requirement analysis and data modelling
3. Data model to relational model
4. Data constraints and built in functions
5. Java database programming
6. Interface to the system
7. Nested queries and Join queries
8. Procedure in MySQL
9. Multidimensional data modelling

Index Sheet

No.	Lab Experiment	(a) Performing the experiment (20)	(b) Document (10)	(c) Viva (10)
1	DDL and DML commands			
2	Requirement analysis and data modelling			
3	Data model to relational model			
4	Data constraints and built in functions			
5	Java database programming			
6	Interface to the system			
7	Nested queries and Join queries			
8	Procedure in MySQL			
9	Multidimensional data modelling			
	Total Marks (Average of 9 labs)			
	(d) Lab Internal Test conducted along the lines of SEE, valued for 50 Marks and reduced to 10 Marks			
	Lab Internal Marks (50) (a+b+c+d)			
	Lab Internal Marks (25)			

Signature of the Staff In-charge

Laboratory 1

Title of the Laboratory Exercise: DDL and DML commands

1. Introduction and Purpose of Experiment

   Structured Query Language (SQL) is used to pass the query to retrieve and manipulate the information from database. Depending upon the nature of query, SQL is divided into different components such as Data Definition Language (DDL) and Data Manipulation Language (DML). DDL statements create the database, maintain the structure of the database and remove database objects such as tables, indexes, and users. DML statements are used for managing data in database such as insert tuples into, delete tuples from, and modify tuples in the database. By doing this lab, students will be able to execute DDL and DML commands

2. **Aim and Objectives**

   Aim

   - To execute Data Definition Language (DDL) and Data Management Language (DML) commands

   Objectives

   At the end of this lab, the student will be able to

   - Create a database and populate it with data using SQL commands
   - Execute DDL and DML commands for the given database

3. **Experimental Procedure**

   i. Analyse the problem statement
   ii. Execute DDL and DML commands
   iii. Create a database for the given schema
   iv. Design SQL commands using DDL and DML commands
   v. Test the executed commands
   vi. Analyse and discuss the outcomes of your experiment
   vii. Document the work

4. **Questions**

   a. Practice DDL and DML commands

   b. Consider the following relational schema that keeps track of the students in a college. Enter at least five tuples for the relation. Assume appropriate domain and data type for each field.

   STUDENT (StudId, StudName, StudAddress)

Execute the following queries based on the above schema

i. Display the details of all the students
ii. Display the name and address of the student with id=101
iii. Insert a new student <105, ‘John’, ‘Bangalore’>
iv. Change the address of the student John to ‘Delhi’
v. Delete the details of a student with student id=105
vi. Add a column to the schema Student with appropriate data type

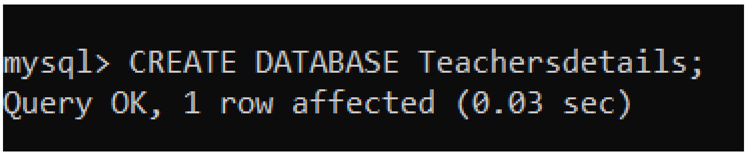
5. Presentation of Results

Question 1

Practice DDL and DML commands

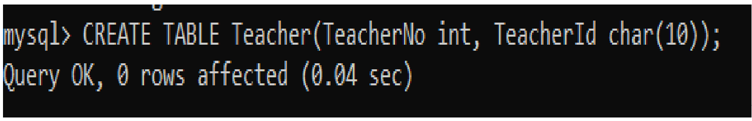
- **DDL Commands**

1. **CREATE**



```
mysql> CREATE DATABASE Teachersdetails;
Query OK, 1 row affected (0.03 sec)
```

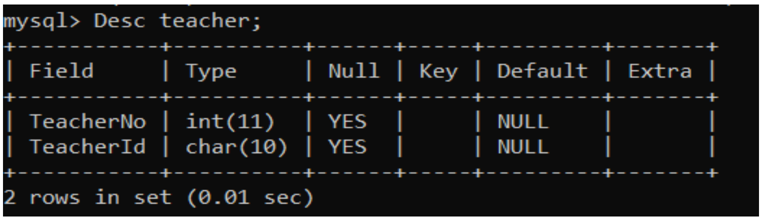
Fig 1.5.1.1 Command to create Database.



```
mysql> CREATE TABLE Teacher(TeacherNo int, TeacherId char(10));
Query OK, 0 rows affected (0.04 sec)
```

Fig 1.5.1.2 Command to create table in selected Database.

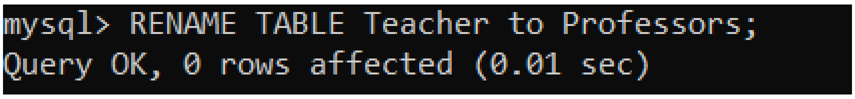
2. **DESC**



```
mysql> Desc teacher;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| TeacherNo | int(11)  | YES  |     | NULL    |       |
| TeacherId | char(10) | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

Fig 1.5.1.3 Command To describe the structure of an specified Table

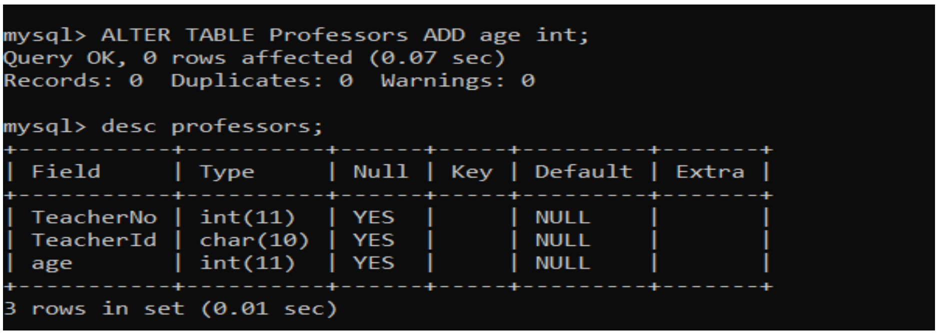
3. **RENAME**



```
mysql> RENAME TABLE Teacher to Professors;
Query OK, 0 rows affected (0.01 sec)
```

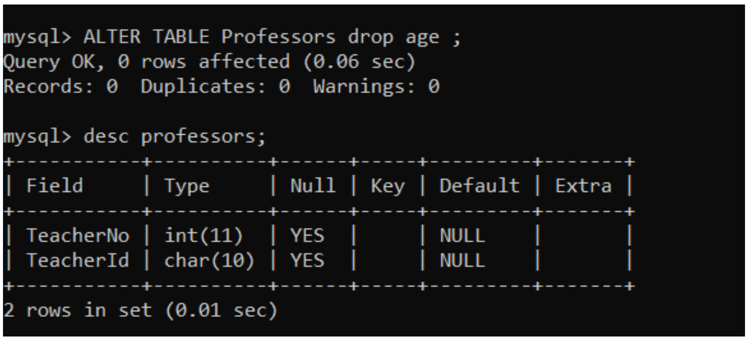
Fig 1.5.1.4 Command To change the name of the table

4. **ALTER**



```
mysql> ALTER TABLE Professors ADD age int;
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc professors;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| TeacherNo | int(11)  | YES  |     | NULL    |       |
| TeacherId | char(10) | YES  |     | NULL    |       |
| age       | int(11)  | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

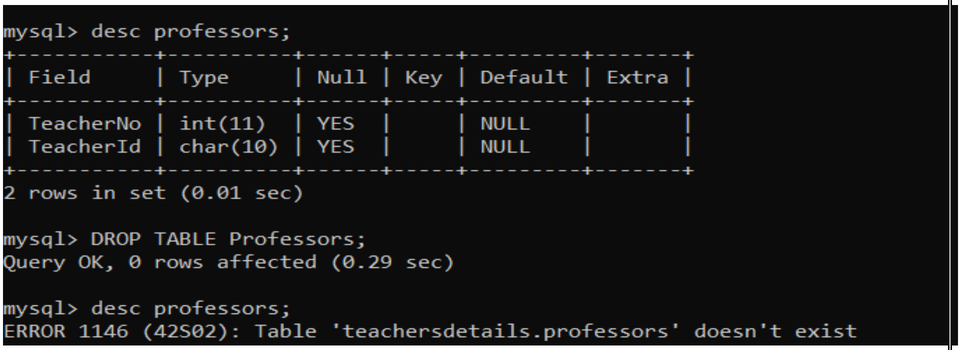
Fig 1.5.1.5 Command to Adding New Column.



```
mysql> ALTER TABLE Professors drop age ;
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc professors;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| TeacherNo | int(11)  | YES  |     | NULL    |       |
| TeacherId | char(10) | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

Fig 1.5.1.6 Command to drop Existing Column.

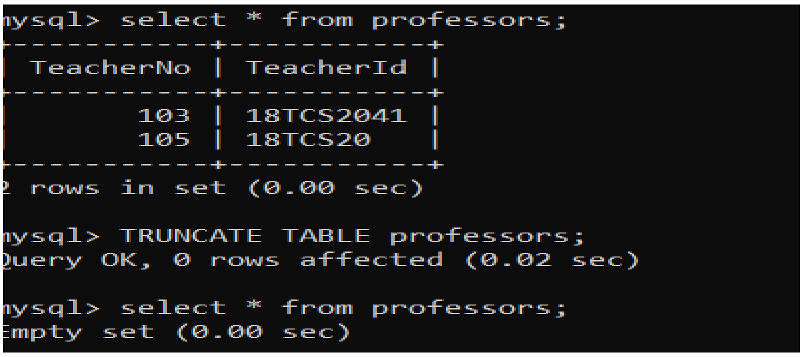
5. **DROP**



```
mysql> desc professors;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| TeacherNo | int(11)  | YES  |     | NULL    |       |
| TeacherId | char(10) | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
2 rows in set (0.01 sec)

mysql> DROP TABLE Professors;
Query OK, 0 rows affected (0.29 sec)

mysql> desc professors;
ERROR 1146 (42S02): Table 'teachersdetails.professors' doesn't exist
```

Fig 1.5.1.7 Command for which a table is dropped .

6. **TRUNCATE**

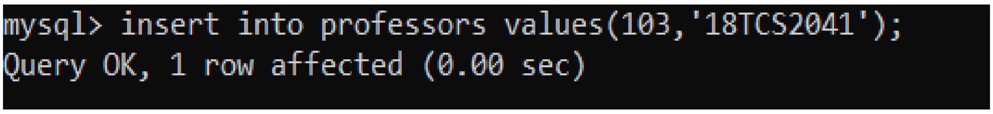


```
ysql> select * from professors;
-----------+-----------+
 TeacherNo | TeacherId |
-----------+-----------+
       103 | 18TCS2041 |
       105 | 18TCS20   |
-----------+-----------+
 rows in set (0.00 sec)

ysql> TRUNCATE TABLE professors;
uery OK, 0 rows affected (0.02 sec)

ysql> select * from professors;
mpty set (0.00 sec)
```

Fig 1.5.1.8 Command To destroy the data in an existing table.

- ## DML Commands

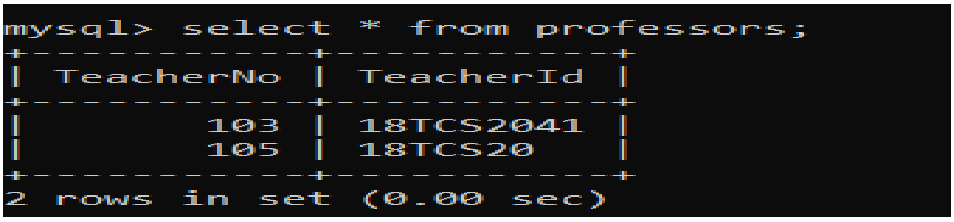
1) **INSERT**



```
mysql> insert into professors values(103,'18TCS2041');
Query OK, 1 row affected (0.00 sec)
```

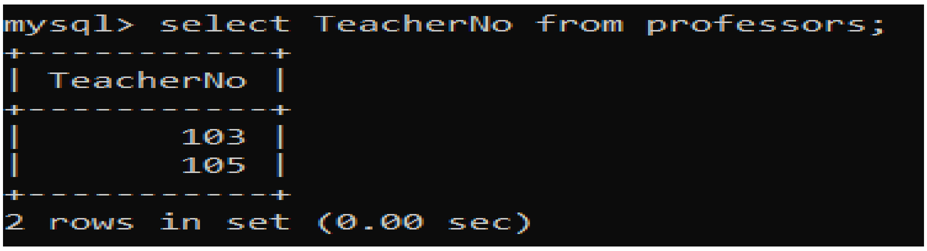
Fig 1.5.1.9 Command to insert data into a table.

2) **SELECT**



```
mysql> select * from professors;
+-----------+-----------+
| TeacherNo | TeacherId |
+-----------+-----------+
|       103 | 18TCS2041 |
|       105 | 18TCS20   |
+-----------+-----------+
2 rows in set (0.00 sec)
```

Fig 1.5.1.10 Command to query or retrieve data from a table in the database.

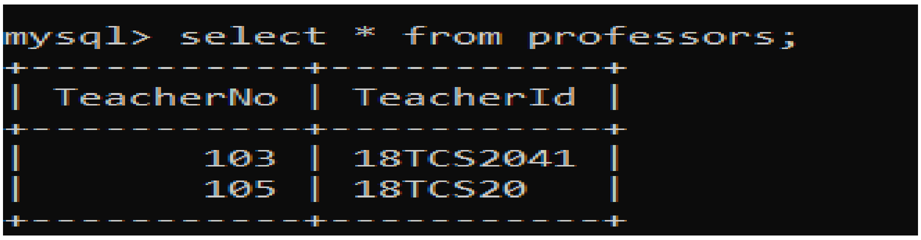


```
mysql> select TeacherNo from professors;
+-----------+
| TeacherNo |
+-----------+
|       103 |
|       105 |
+-----------+
2 rows in set (0.00 sec)
```

Fig 1.5.1.11 Command To get a particular column

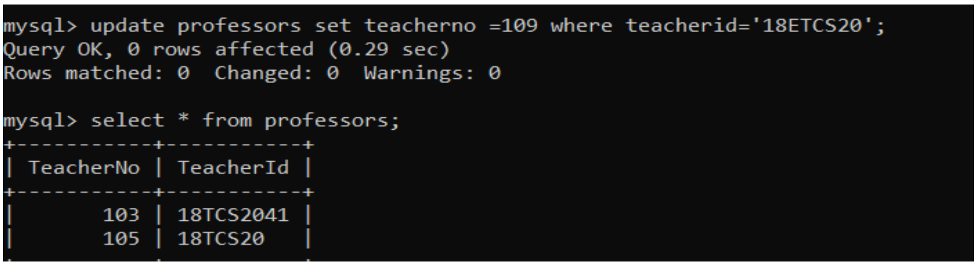
3) **UPDATE**

Before



```
mysql> select * from professors;
+-----------+-----------+
| TeacherNo | TeacherId |
+-----------+-----------+
|       103 | 18TCS2041 |
|       105 | 18TCS20   |
+-----------+-----------+
```

After

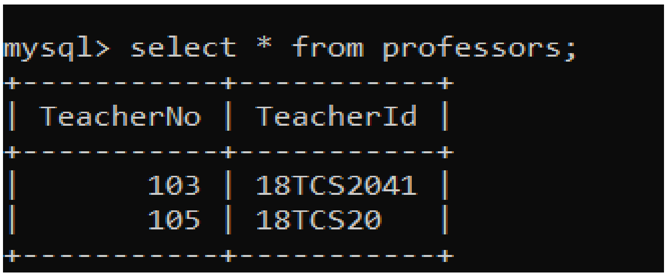


```
mysql> update professors set teacherno =109 where teacherid='18ETCS20';
Query OK, 0 rows affected (0.29 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql> select * from professors;
+-----------+-----------+
| TeacherNo | TeacherId |
+-----------+-----------+
|       103 | 18TCS2041 |
|       105 | 18TCS20   |
```

Fig 1.5.1.12 Command to update a row of a table.

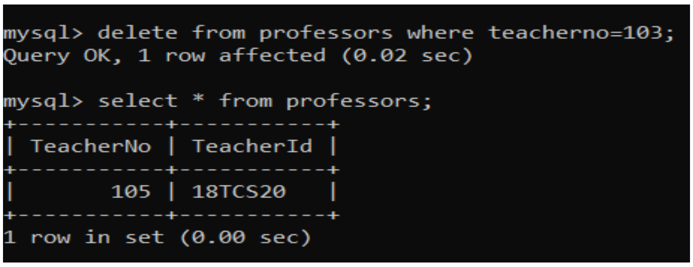
4) DELETE

Before



```
mysql> select * from professors;
+-----------+-----------+
| TeacherNo | TeacherId |
+-----------+-----------+
|       103 | 18TCS2041 |
|       105 | 18TCS20   |
+-----------+-----------+
```

After



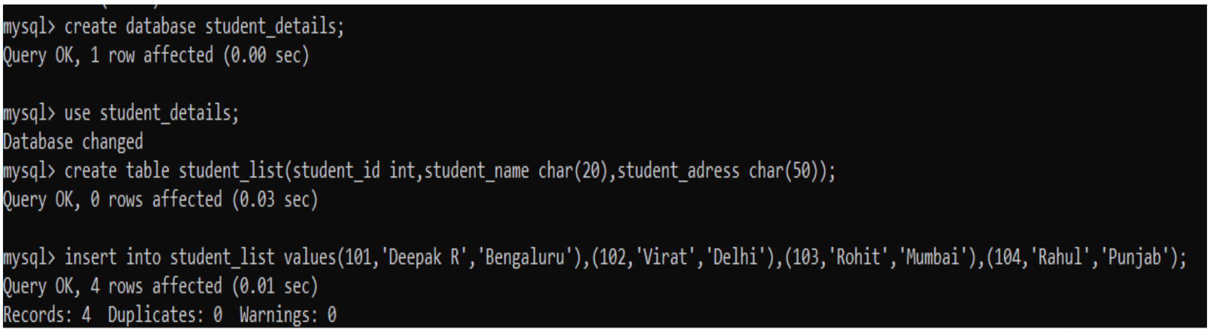
```
mysql> delete from professors where teacherno=103;
Query OK, 1 row affected (0.02 sec)

mysql> select * from professors;
+-----------+-----------+
| TeacherNo | TeacherId |
+-----------+-----------+
|       105 | 18TCS20   |
+-----------+-----------+
1 row in set (0.00 sec)
```

Fig 1.5.1.13 Command to removes one or more records from a table

Question 2

Consider the following relational schema that keeps track of the students in a college. Enter at least five tuples for the relation. Assume appropriate domain and data type for each field.

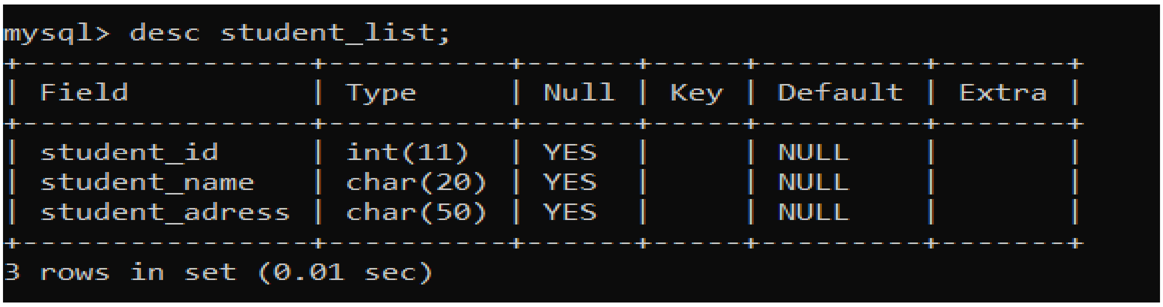
Create a Table and Insert data into it.



```
mysql> create database student_details;
Query OK, 1 row affected (0.00 sec)

mysql> use student_details;
Database changed
mysql> create table student_list(student_id int,student_name char(20),student_adress char(50));
Query OK, 0 rows affected (0.03 sec)

mysql> insert into student_list values(101,'Deepak R','Bengaluru'),(102,'Virat','Delhi'),(103,'Rohit','Mumbai'),(104,'Rahul','Punjab');
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

Fig 1.5.2.1 Creating table and inserting data into it

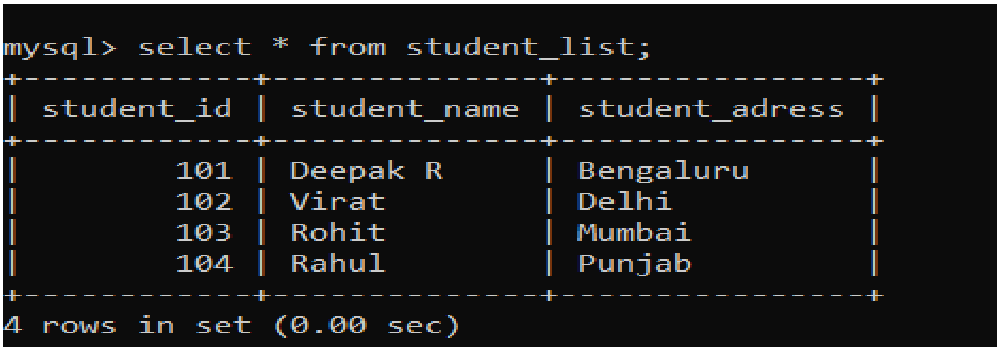
Describe Table



```
mysql> desc student_list;
+----------------+----------+------+-----+---------+-------+
| Field          | Type     | Null | Key | Default | Extra |
+----------------+----------+------+-----+---------+-------+
| student_id     | int(11)  | YES  |     | NULL    |       |
| student_name   | char(20) | YES  |     | NULL    |       |
| student_adress | char(50) | YES  |     | NULL    |       |
+----------------+----------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

Fig 1.5.2.2 To describe the structure of an specified Table

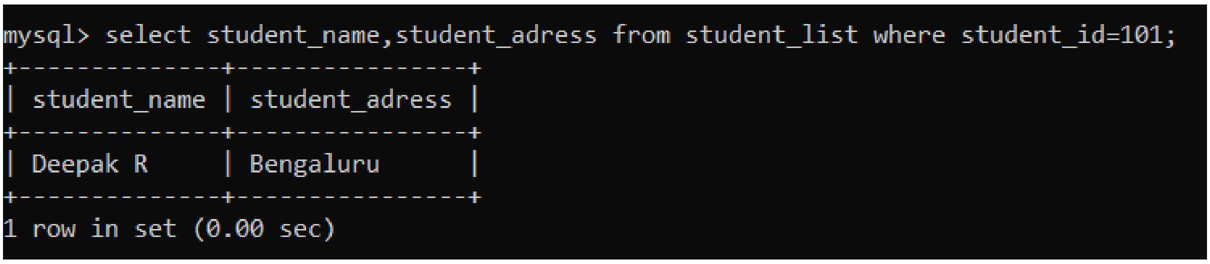
i. **<u>Display the details of all the students</u>**



```
mysql> select * from student_list;
+------------+--------------+----------------+
| student_id | student_name | student_adress |
+------------+--------------+----------------+
|        101 | Deepak R     | Bengaluru      |
|        102 | Virat        | Delhi          |
|        103 | Rohit        | Mumbai         |
|        104 | Rahul        | Punjab         |
+------------+--------------+----------------+
4 rows in set (0.00 sec)
```

Fig 1.5.2.3 Displays the details of students specified in Student_list.

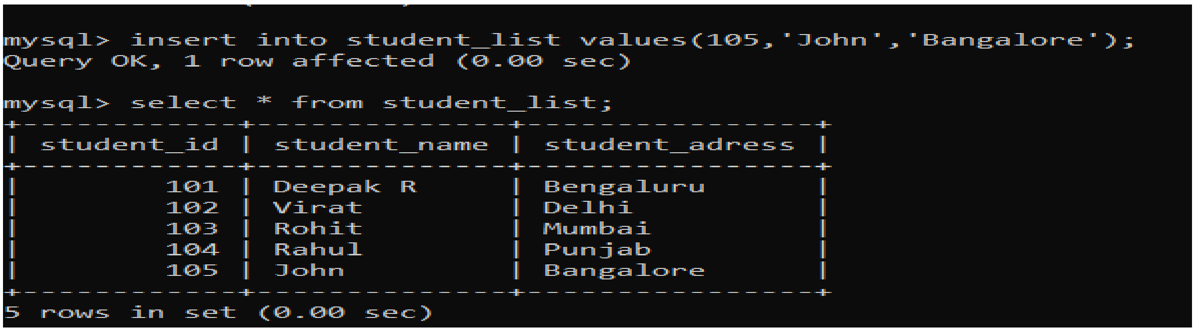
ii. **<u>Display the name and address of the student with id=101</u>**



```
mysql> select student_name,student_adress from student_list where student_id=101;
+--------------+----------------+
| student_name | student_adress |
+--------------+----------------+
| Deepak R     | Bengaluru      |
+--------------+----------------+
1 row in set (0.00 sec)
```

Fig 1.5.2.4 Displays the name and address of the student with id=101

iii. **<u>Insert a new student <105, ‘John’, ‘Bangalore’></u>**

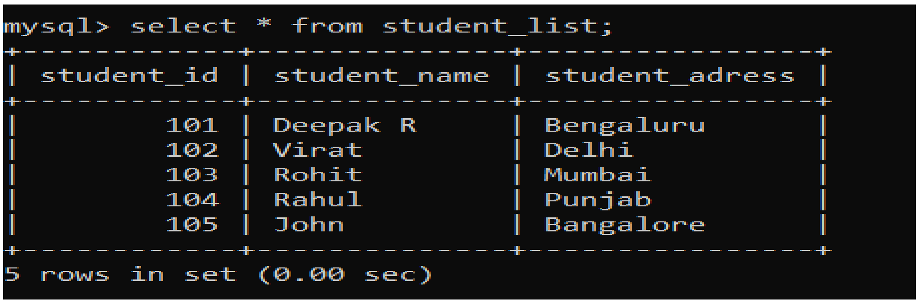


```
mysql> insert into student_list values(105,'John','Bangalore');
Query OK, 1 row affected (0.00 sec)

mysql> select * from student_list;
+------------+--------------+----------------+
| student_id | student_name | student_adress |
+------------+--------------+----------------+
|        101 | Deepak R     | Bengaluru      |
|        102 | Virat        | Delhi          |
|        103 | Rohit        | Mumbai         |
|        104 | Rahul        | Punjab         |
|        105 | John         | Bangalore      |
+------------+--------------+----------------+
5 rows in set (0.00 sec)
```

Fig 1.5.2.5 Inserts a new student <105, ‘John’, ‘Bangalore’> into the Student_list table.

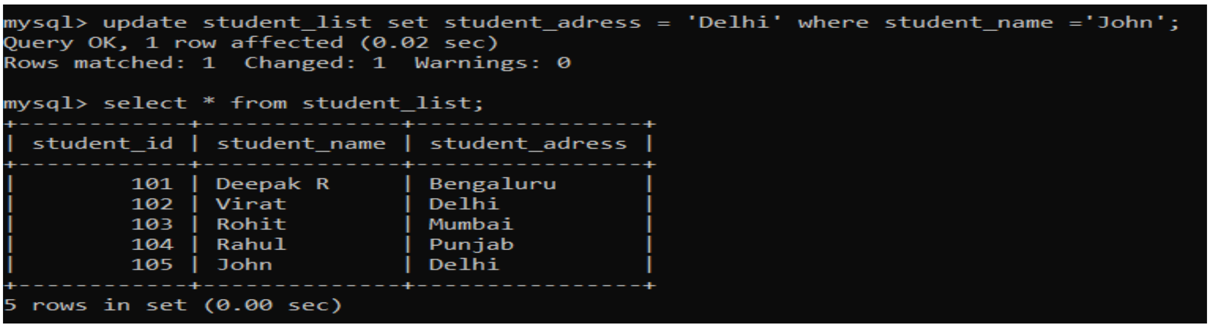
iv. **<u>Change the address of the student John to ‘Delhi’</u>**

Before



```
mysql> select * from student_list;
+------------+--------------+----------------+
| student_id | student_name | student_adress |
+------------+--------------+----------------+
|        101 | Deepak R     | Bengaluru      |
|        102 | Virat        | Delhi          |
|        103 | Rohit        | Mumbai         |
|        104 | Rahul        | Punjab         |
|        105 | John         | Bangalore      |
+------------+--------------+----------------+
5 rows in set (0.00 sec)
```

After

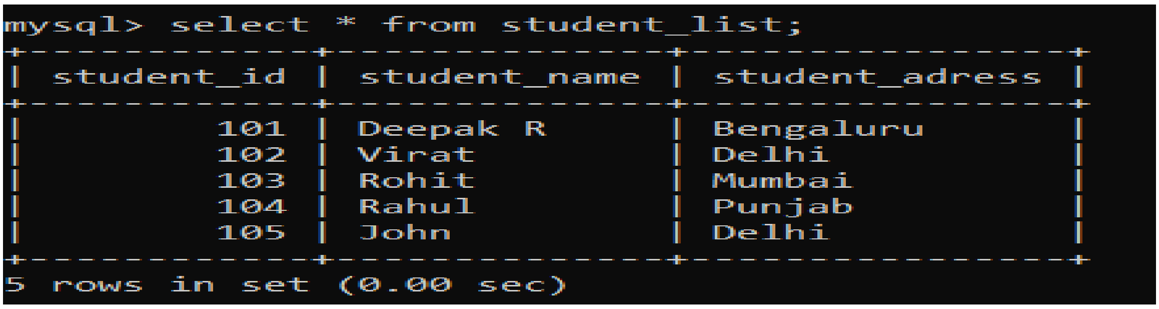


```
mysql> update student_list set student_adress = 'Delhi' where student_name ='John';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from student_list;
+------------+--------------+----------------+
| student_id | student_name | student_adress |
+------------+--------------+----------------+
|        101 | Deepak R     | Bengaluru      |
|        102 | Virat        | Delhi          |
|        103 | Rohit        | Mumbai         |
|        104 | Rahul        | Punjab         |
|        105 | John         | Delhi          |
+------------+--------------+----------------+
5 rows in set (0.00 sec)
```

Fig 1.5.2.5 Changing the address of the student John to ‘Delhi’

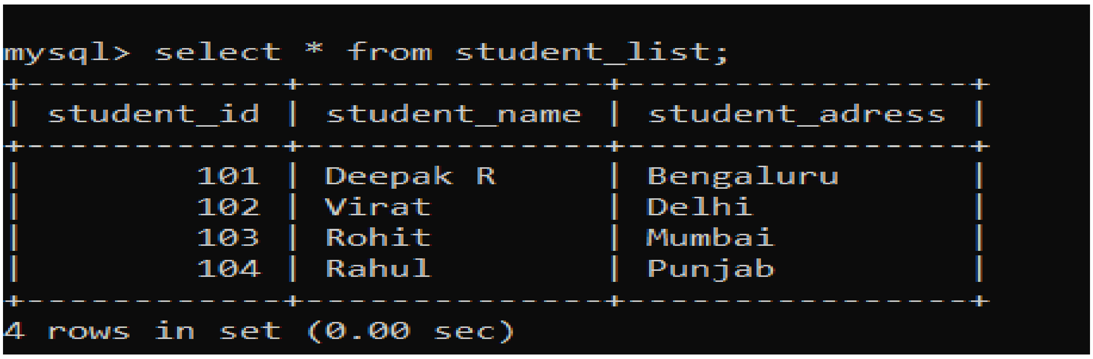
v. **<u>Delete the details of a student with student id=105</u>**

Before



```
mysql> select * from student_list;
+------------+--------------+----------------+
| student_id | student_name | student_adress |
+------------+--------------+----------------+
|        101 | Deepak R     | Bengaluru      |
|        102 | Virat        | Delhi          |
|        103 | Rohit        | Mumbai         |
|        104 | Rahul        | Punjab         |
|        105 | John         | Delhi          |
+------------+--------------+----------------+
5 rows in set (0.00 sec)
```

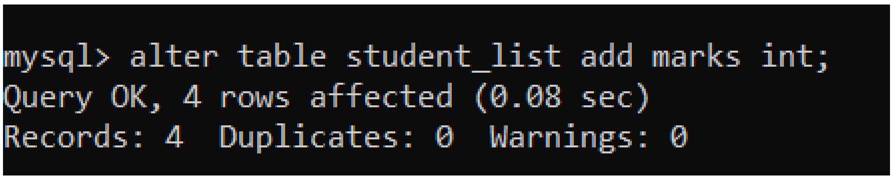
After



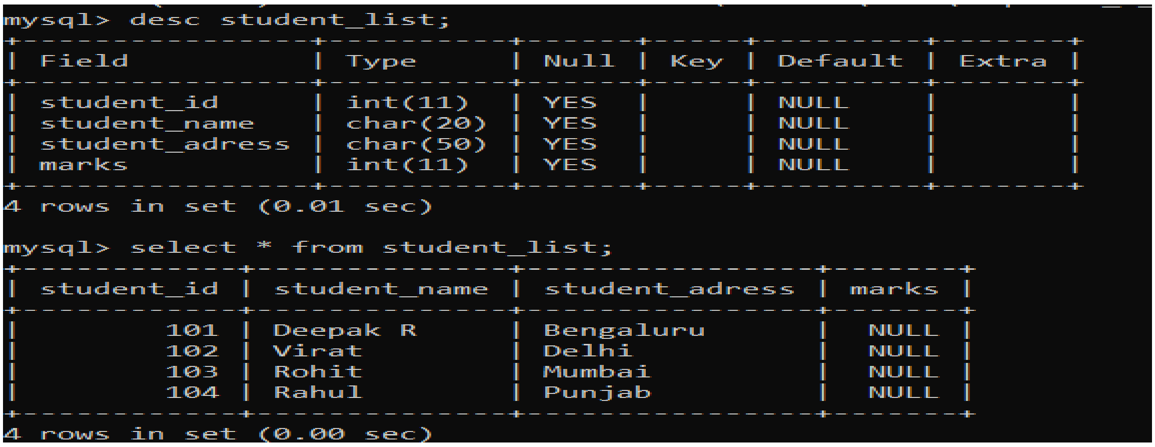
```
mysql> select * from student_list;
+------------+--------------+----------------+
| student_id | student_name | student_adress |
+------------+--------------+----------------+
|        101 | Deepak R     | Bengaluru      |
|        102 | Virat        | Delhi          |
|        103 | Rohit        | Mumbai         |
|        104 | Rahul        | Punjab         |
+------------+--------------+----------------+
4 rows in set (0.00 sec)
```

Fig 1.5.2.6 Deletes the details of a student with student id=105

vi. **Add a column to the schema Student with appropriate data type**



```
mysql> alter table student_list add marks int;
Query OK, 4 rows affected (0.08 sec)
Records: 4  Duplicates: 0  Warnings: 0
```



```
mysql> desc student_list;
+-----------------+----------+------+-----+---------+-------+
| Field           | Type     | Null | Key | Default | Extra |
+-----------------+----------+------+-----+---------+-------+
| student_id      | int(11)  | YES  |     | NULL    |       |
| student_name    | char(20) | YES  |     | NULL    |       |
| student_adress  | char(50) | YES  |     | NULL    |       |
| marks           | int(11)  | YES  |     | NULL    |       |
+-----------------+----------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> select * from student_list;
+------------+--------------+----------------+-------+
| student_id | student_name | student_adress | marks |
+------------+--------------+----------------+-------+
|        101 | Deepak R     | Bengaluru      |  NULL |
|        102 | Virat        | Delhi          |  NULL |
|        103 | Rohit        | Mumbai         |  NULL |
|        104 | Rahul        | Punjab         |  NULL |
+------------+--------------+----------------+-------+
4 rows in set (0.00 sec)
```

Fig 1.5.2.7 Adds a column to the Student_list with appropriate data type

6. Conclusions

Data Definition Language (DDL) and Data Manipulation Language (DML) together forms a Database Language. The basic difference between DDL and DML is that DDL (Data Definition Language) is used to Specify the database schema database structure.

On the other hand, DML (Data Manipulation Language) is used to access, modify or retrieve the data from the database. Let us discuss the differences between DDL and DML, with the help of comparison chart shown below.

some commands of DDL:

- CREATE is command used to create a new Database or Table.
- ALTER command is used to alter the content in the Table.
- DROP is used to delete some content in the database or table.
- TRUNCATE is used to delete all the content from the table.
- RENAME is used to rename the content in the database.

commands used in DML are as follow:

- SELECT used to retrieve the data from the Table.
- INSERT used to push the data in the Table.
- UPDATE used to reform the data in the Table.
- DELETE used to delete the data from the Table.

7. Comments

1. Limitations of Experiments

The experiment uses command line utilities to call sql commands, while a programming language with a sql connector should be used. For understanding how a sql query works the current approach is fine, but for data manipulation a programming language should be used.

2. Limitations of Results

The results are limited to the number of queries we run, the extents of the database are not tested in this experiment, such as heavy database loads and large file storage.

3. Learning happened

We learnt how to use DDL and DML commands for SQL.

4. Recommendations

Use a programming language connector for data manipulation and perform extensive testing for the database.