Aim: SQL Languages and Constraints.

Objectives:

1. To understand how to create and modify tables in MySQL and apply constraints for data integrity.
2. To practice adding, updating, and deleting records in a MySQL database.
3. To implement and manage primary and foreign keys along with unique and check constraints, ensuring proper data relationships and data validity.

Tools Used: MySQL Workbench.

Concept:

Theoretical Concepts:

- SQL (Structured Query Language): SQL is a widely used language designed to manage and work with relational databases. It provides commands that allow us to create tables, retrieve data, and update or delete information efficiently.

Example:
Creating a table:

```
CREATE TABLE student (student_id int, student_name varchar(20));
```

- Table Modification in SQL: With SQL commands like ALTER TABLE, we can add, rename, or modify columns in an existing table. This allows us to adjust tables as data needs change over time.

Example:
Adding new columns:

```
alter table student add age int, add phone_no int;
```

- Constraints in SQL: Constraints are rules applied to columns in a table to ensure data accuracy and reliability. They help maintain relationships and prevent invalid data entries.
    - Primary Key: A unique identifier for each row in a table. It ensures that each record is unique and that this field cannot be empty, promoting data accuracy.
    - Unique Key: Guarantees that each value in a column is unique, preventing duplicates.
    - Foreign Key: Links columns in one table to the primary key of another table, creating relationships between tables. This ensures that values in the related column exist in the other table.
    - Check Constraint: Verifies that data entered meets a specified condition. For instance, we can set a check constraint on a marks column to allow only non-negative values, preventing invalid entries.

Example:
Setting primary and unique constraints:

```
alter table student add constraint pk_student primary key (student_id);
```

Permissions in SQL: SQL allows administrators to control who can access or make changes to tables by assigning specific permissions to users. With commands like GRANT, admins can give privileges, ensuring that only authorized users can view or update data.

Example:
Granting permissions to a user:

GRANT ALL PRIVILEGES ON student TO 'XYZ'@'localhost';

- Relational Integrity with Foreign Keys and Constraints: By creating a Marks table that references student_id in the student table, we create a structured link between student information and their marks. Constraints like foreign keys enforce that each mark entry corresponds to a valid student, and check constraints on marks ensure data accuracy by preventing invalid values, such as negative scores.

Problem Statement

1) Create following student table in MySQL student (student_id int, student_name varchar(20))

2) Add two more columns in student table namely (age int, phone_no int)

3) Rename the column name student_name with sname

4) Add the column (class varchar(20)) after sname

5) Rename the datatype of sname to varchar(30)

6) Make student_id a primary key.

7) Make sname an Unique key.

8) Insert following.

student_id	sname	class	age	phone_no
1	Sanjay	symca	23	242543
2	Vaidehi	fymca	24	454354
3	Akshata	symca	21	543543
4	Vidula	fymca	22	435454
5	Pratik	symca	23	345435

9) Modify the age of Akshata to 22

10) Delete the record of Pratik

11) Create one user XYZ and give him a permission to make the changes in the above table.

12) Login to XYZ and make sure he is able to make the changes in student table created by Root user.
13) Create following marks table in MySQL Marks (sid, subject1, subject2, subject3)

14) Make sid a foreign key which refers to student_id of student table

15) Apply check constraint on subject1 to verify that no one can enter negative marks to it.

16) Insert following values in marks table

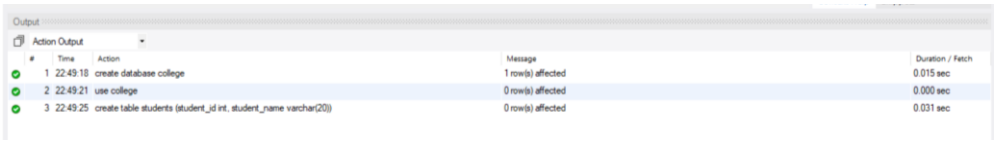
sid	Subject1	Subject2	Subject3
1	89	78	89
3	99	67	56
4	90	66	45
6	89	88	88

Solution:

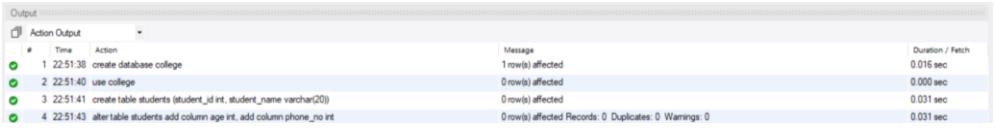
1)create database college;

use college;

create table students (student_id int, student_name varchar(20));



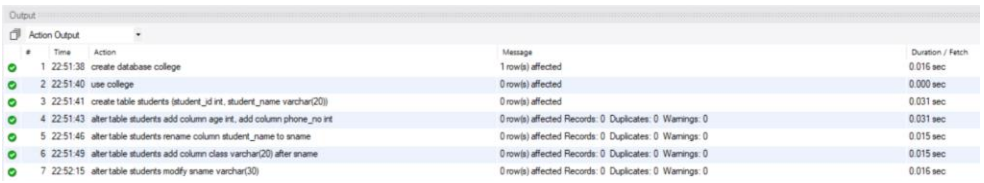
2)alter table students add column age int, add column phone_no int;



3)alter table students rename column student_name to sname;

4)alter table students add column class varchar(20) after sname;

5)alter table students modify sname varchar(30) ;



6)alter table students add constraint primary key(student_id);

7)alter table students add constraint unique (sname);

8)insert into students (student_id, sname, class, age, phone_no)

values

(1, 'Sanjay', 'symca', 23, 242543),

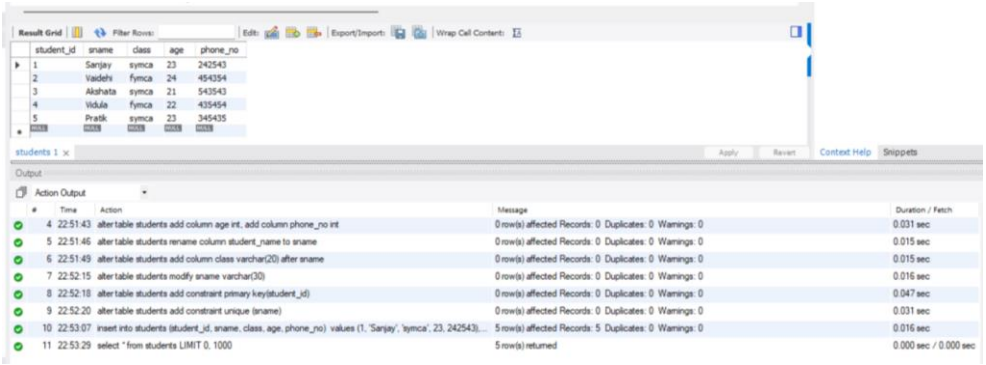
(2, 'Vaidehi', 'fymca', 24, 454354),

(3, 'Akshata', 'symca', 21, 543543),

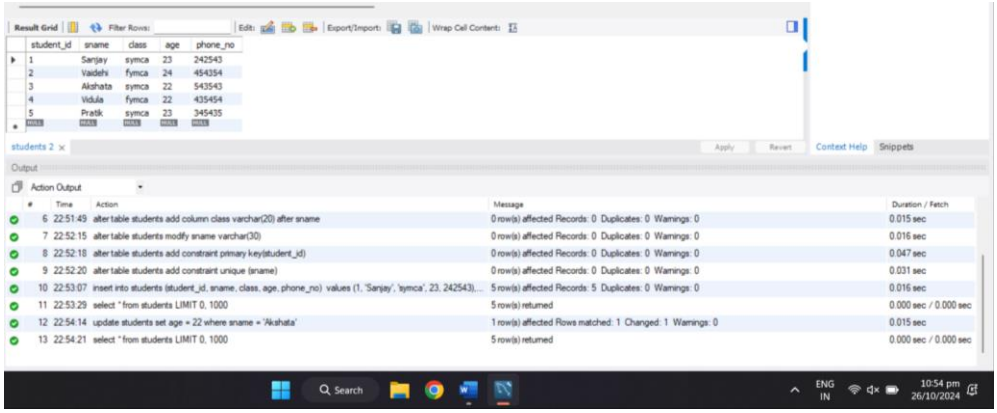
(4, 'Vidula', 'fymca', 22, 435454),

(5, 'Pratik', 'symca', 23, 345435)

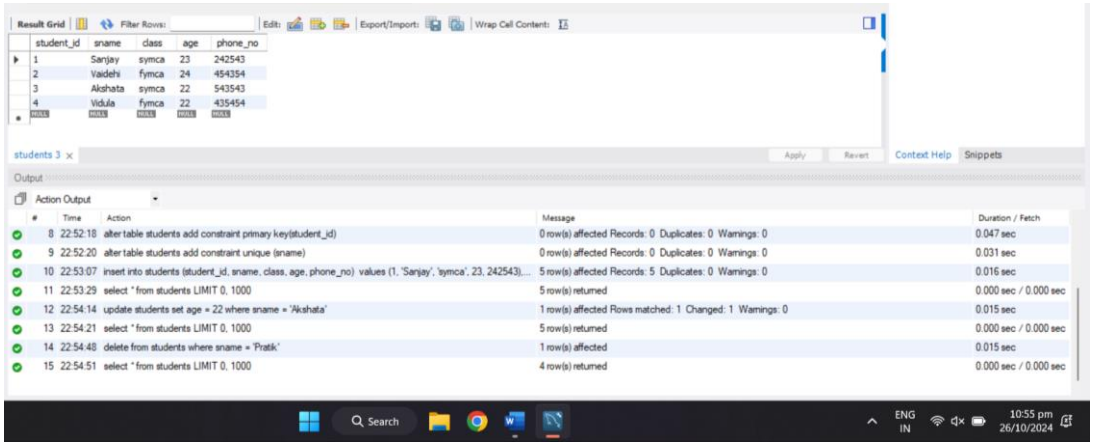
;



9) update students set age = 22 where sname = 'Akshata';



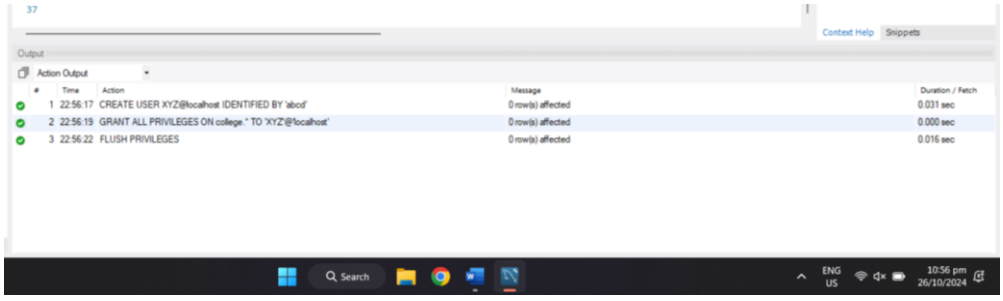
10) delete from students where sname = 'Pratik';



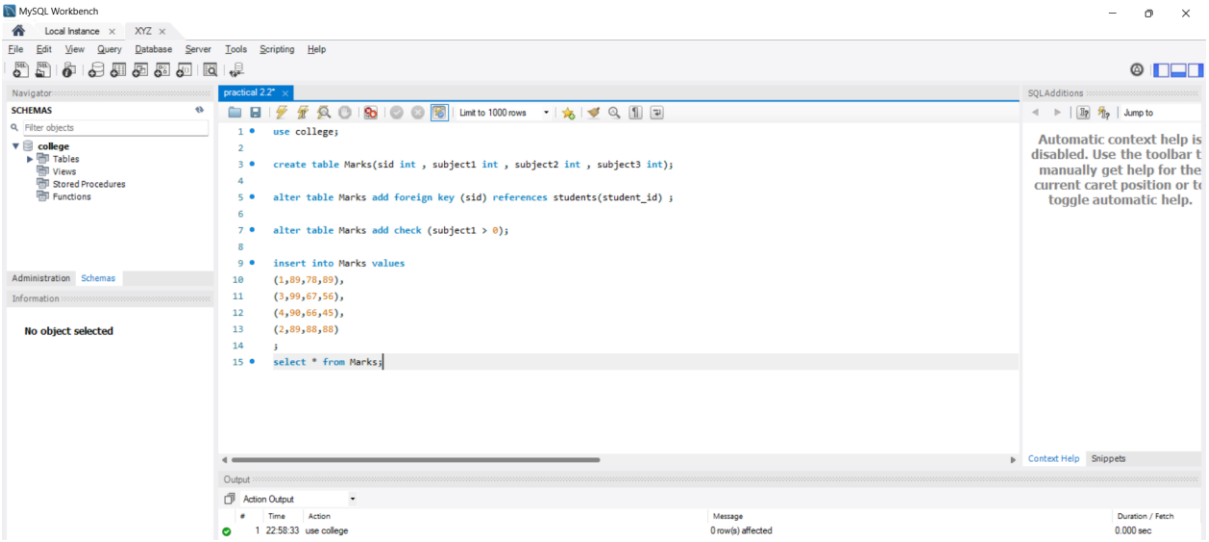
11) create user XYZ@localhost identified by 'abcd';

Grant all privileges on college.* TO 'XYZ'@'localhost';

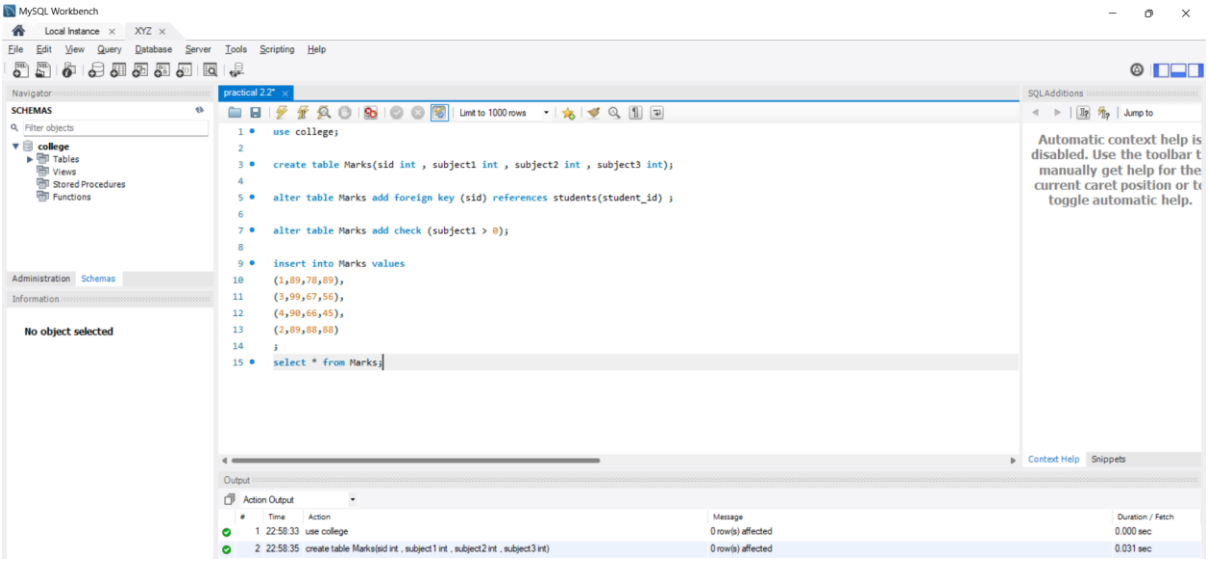
flush privileges;



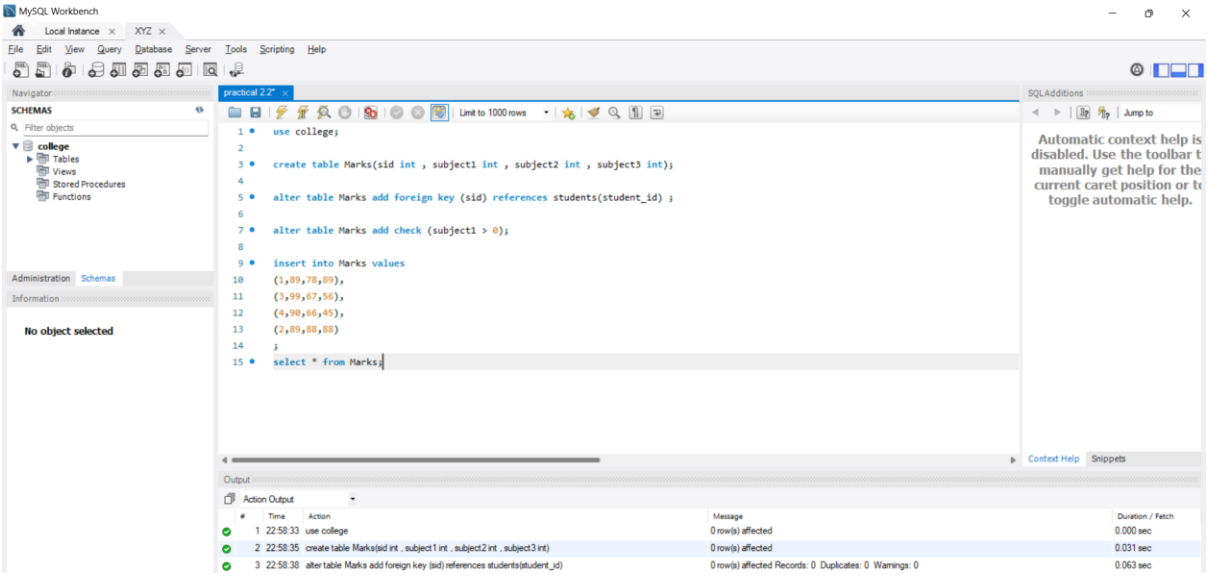
12) use college;



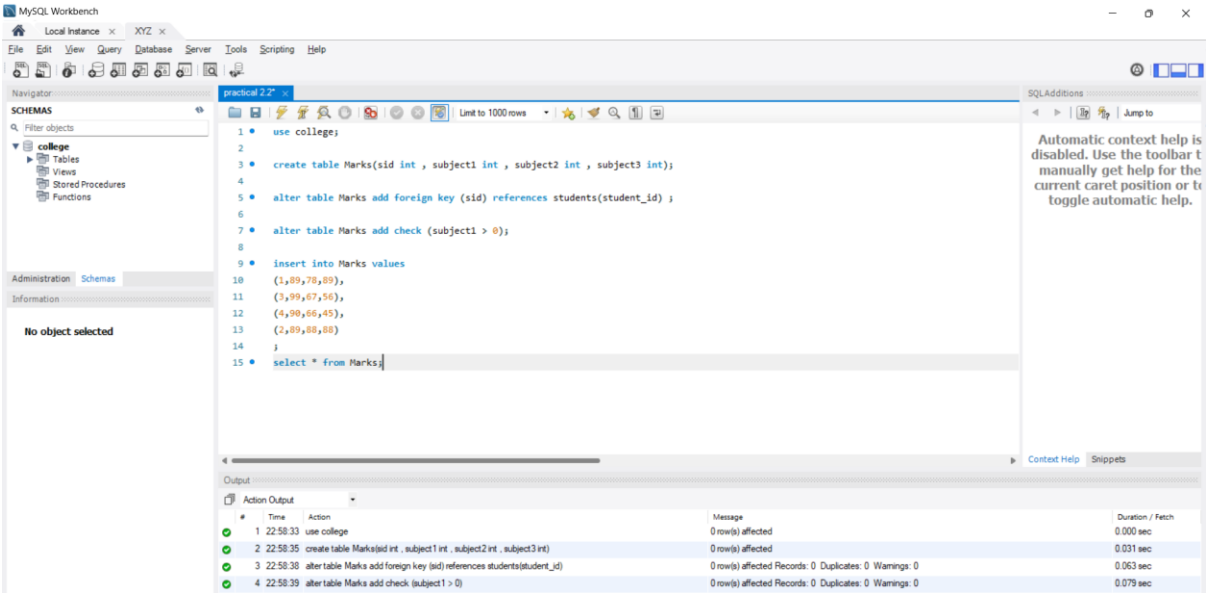
13) create table Marks(sid int , subject1 int , subject2 int , subject3 int);



14) alter table Marks add foreign key (sid) references students(student_id) ;



15) alter table Marks add check (subject1 > 0);



16) insert into Marks values

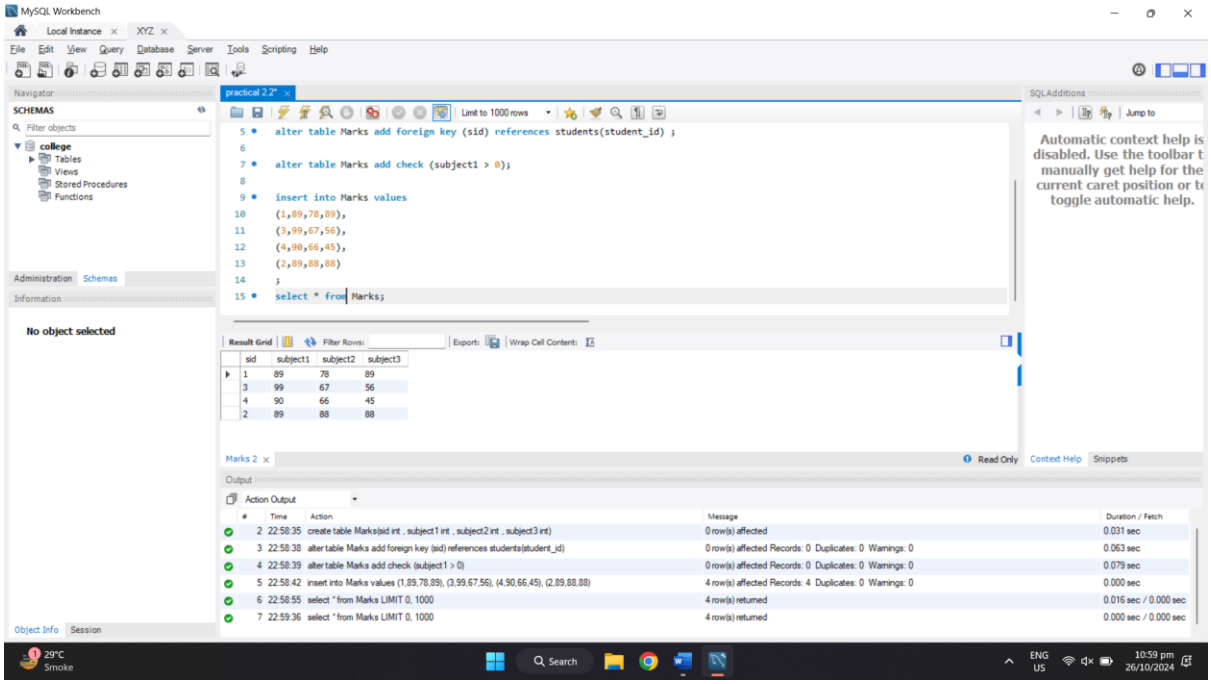
(1,89,78,89),

(3,99,67,56),

(4,90,66,45),

(2,89,88,88)

;



Observation:

Through this experiment, I learned how to create and manipulate tables using SQL commands and apply constraints to manage data integrity. By setting up primary and foreign keys, I observed how tables can be linked to maintain a relational structure. Constraints like unique keys and check constraints were useful in enforcing data validity, ensuring no duplicate or invalid data is entered. Additionally, working with user permissions highlighted the importance of managing access rights in database operations.