ASSIGNMENT 2

PRIYANSH CHAUHAN

20BCE1374

Q1)

USE assignment2;

CREATE TABLE student (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50),

age INT,

grade CHAR(1)

);

DESC student;

INSERT INTO student (name, age, grade)

VALUES ('John', 18, 'A'),

('Jane', 17, 'B'),

('Mike', 19, 'C');

select \* from student;

UPDATE student

SET grade = 'A'

WHERE name = 'John';

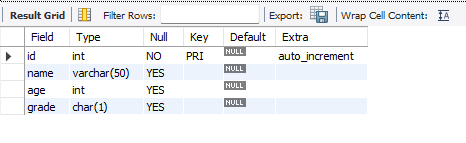
select \* from student;

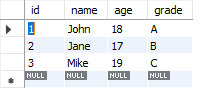
DELETE FROM student

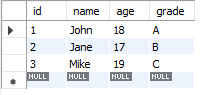
WHERE name = 'Mike';

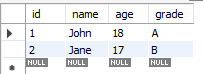
select \* from student;

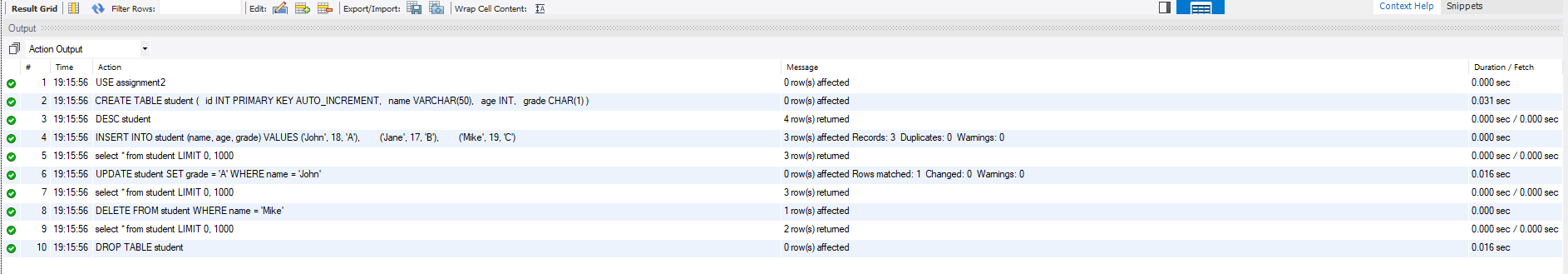
DROP TABLE student;











Q2)

1. Create Tables:

USE assignment2;

CREATE TABLE students (

id INT PRIMARY KEY,

name VARCHAR(50),

age INT,

grade\_id INT

);

CREATE TABLE grades (

id INT PRIMARY KEY,

grade\_name VARCHAR(10)

);

CREATE TABLE courses (

id INT PRIMARY KEY,

course\_name VARCHAR(50),

teacher\_id INT

);

CREATE TABLE teachers (

id INT PRIMARY KEY,

name VARCHAR(50)

);

These commands create four tables: "students", "grades", "courses", and "teachers". The "students" table has columns for student ID, name, age, and grade ID. The "grades" table has columns for grade ID and grade name. The "courses" table has columns for course ID, course name, and teacher ID. The "teachers" table has columns for teacher ID and name.

1. Insert Data:

INSERT INTO students (id, name, age, grade\_id)

VALUES (1, 'John', 18, 1),

(2, 'Jane', 17, 1),

(3, 'Mike', 19, 2);

INSERT INTO grades (id, grade\_name)

VALUES (1, 'Grade 10'),

(2, 'Grade 11');

INSERT INTO courses (id, course\_name, teacher\_id)

VALUES (1, 'Math', 1),

(2, 'English', 2);

INSERT INTO teachers (id, name)

VALUES (1, 'Mr. Smith'),

(2, 'Ms. Johnson');

These commands insert sample data into the respective tables.

1. Inner Join:

SELECT students.name, grades.grade\_name

FROM students

INNER JOIN grades ON students.grade\_id = grades.id;

This command performs an inner join between the "students" and "grades" tables based on the common column "grade\_id", and retrieves the student names along with their respective grade names.

1. Left Join:

SELECT students.name, courses.course\_name

FROM students

LEFT JOIN courses ON students.id = courses.teacher\_id;

This command performs a left join between the "students" and "courses" tables based on the common column "id" and "teacher\_id", respectively. It retrieves the student names along with their respective course names if available.

1. Right Join:

SELECT teachers.name, courses.course\_name

FROM teachers

RIGHT JOIN courses ON teachers.id = courses.teacher\_id;

This command performs a right join between the "teachers" and "courses" tables based on the common column "id" and "teacher\_id", respectively. It retrieves the teacher names along with their respective course names.

1. Full Outer Join:

SELECT students.name, courses.course\_name

FROM students

LEFT JOIN courses ON students.id = courses.teacher\_id

UNION

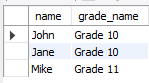
SELECT teachers.name, courses.course\_name

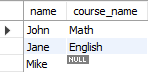
FROM teachers

RIGHT JOIN courses ON teachers.id = courses.teacher\_id;

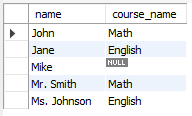
Note: MySQL does not support a direct full outer join. The above query is an alternative way to achieve a similar result. It combines the results of a left join and a right join to simulate a full outer join. It retrieves student names and course names, matching records from both tables.

These SQL commands demonstrate how to create tables for a school and perform different types of joins in MySQL.

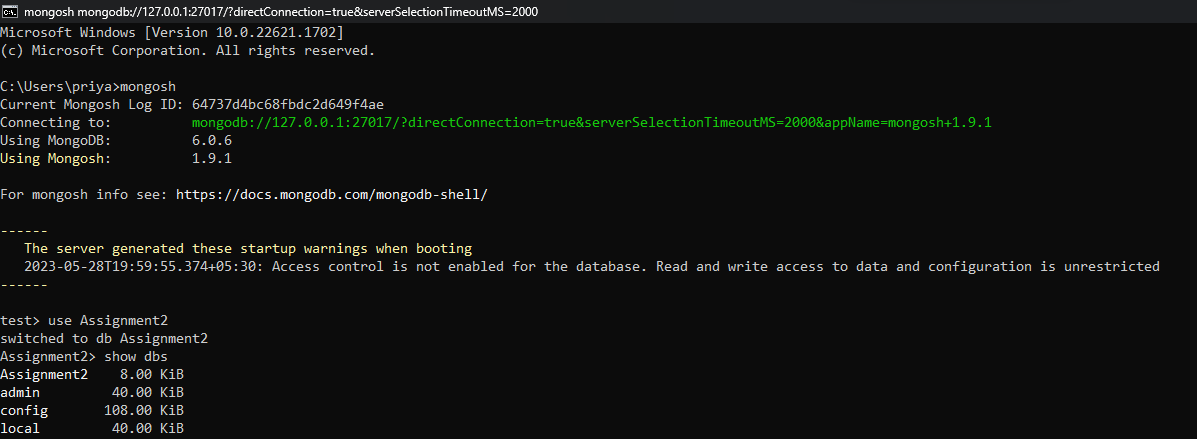




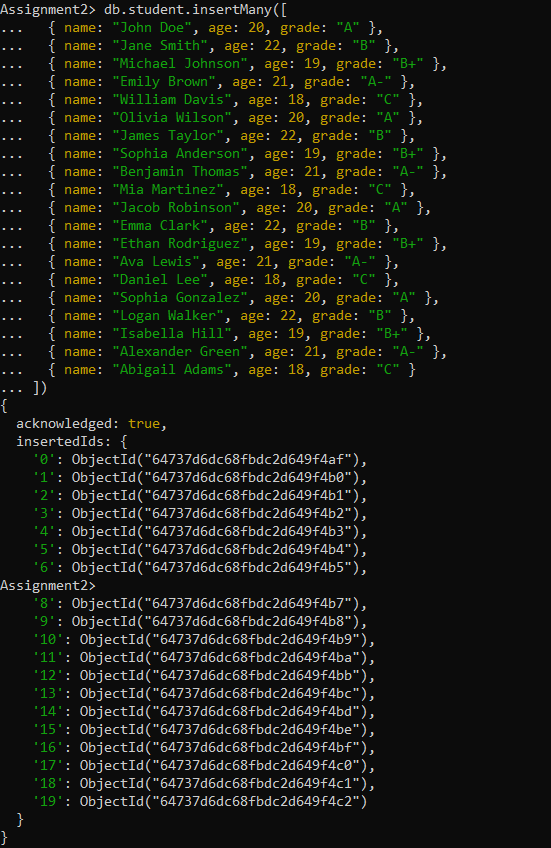




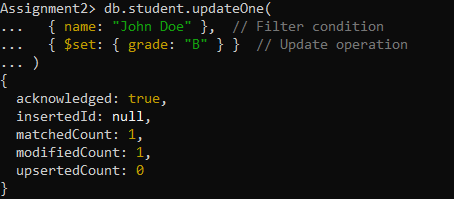
Q3)



1. To create the "student" collection



1. To update a document in the "student" collection, we use the updateOne method. Here's an example updating the "grade" field of a specific student:



1. To delete a document from the "student" collection, we use the deleteOne method. Here's an example deleting a student document:

