Date:23/11/2024

Name:Venkatesh Kumar B

SQL

```
USE learning_college;

 SET sql_safe_updates = 0;

-- Create the Student table

CREATE TABLE IF NOT EXISTS Student (

studentId INT,

 studentName VARCHAR(30),

studentDOB DATE,

studentLocation VARCHAR(30),

departmentId INT,

professorName VARCHAR(30),

PRIMARY KEY (studentId),

FOREIGN KEY (departmentId) REFERENCES Department(departmentId)

);

-- Create the Department table
```

```
CREATE TABLE IF NOT EXISTS Department (
departmentId INT PRIMARY KEY,
departmentName VARCHAR(30),
studentCount INT
);
-- Create the Course table
CREATE TABLE IF NOT EXISTS Course (
courseId INT PRIMARY KEY,
courseName VARCHAR(50),
departmentId INT,
FOREIGN KEY (departmentId) REFERENCES Department(departmentId)
);
-- Create the Professor table
 CREATE TABLE IF NOT EXISTS Professor (
professorId INT PRIMARY KEY,
 professorName VARCHAR(30),
 departmentId INT,
FOREIGN KEY (departmentId) REFERENCES Department(departmentId)
 );
-- Alter the Course table to change courseId to a varchar
ALTER TABLE Course MODIFY courseId VARCHAR(20);
ALTER TABLE Course MODIFY courseName VARCHAR(100);
```

```sql
-- Insert sample data into the Department table

INSERT INTO Department (departmentId, departmentName, studentCount)
VALUES

(101, 'Computer Science', 0),

 (102, 'Mathematics', 0),

(103, 'Physics', 0),

(104, 'Chemistry', 0),

(105, 'Biology', 0);

-- Insert sample data into the Course table

INSERT INTO Course (courseId, courseName, departmentId) VALUES

 ('CS101', 'Introduction to Programming', 101),

('CS102', 'Data Structures and Algorithms', 101),

('MATH101', 'Calculus I', 102),

 ('MATH102', 'Linear Algebra', 102),

('PHYS101', 'Mechanics', 103),

 ('PHYS102', 'Electromagnetism', 103),

 ('CHEM101', 'General Chemistry', 104),

('BIO101', 'Biology Basics', 105);

-- Insert sample data into the Professor table

INSERT INTO Professor (professorId, professorName, departmentId) VALUES

(1, 'Dr. Smith', 101),

(2, 'Dr. Johnson', 102),

(3, 'Dr. Brown', 103),
(4, 'Dr. Taylor', 104),
 (5, 'Dr. White', 105);
-- Insert sample data into the Student table
INSERT INTO Student (studentId, studentName, studentDOB,
studentLocation, departmentId, professorName) VALUES
 (1001, 'Alice', '2000-05-12', 'New York', 101, 'Dr. Smith'),
(1002, 'Bob', '1999-08-22', 'California', 102, 'Dr. Johnson'),
(1003, 'Charlie', '2001-02-14', 'Texas', 103, 'Dr. Brown'),
(1004, 'Diana', '2000-11-30', 'Florida', 104, 'Dr. Taylor'),
(1005, 'Eve', '1998-09-07', 'Washington', 105, 'Dr. White');
-- Query 1: Display all rows and columns from the Student table
SELECT * FROM Student;
-- Query 2: Retrieve only the name and department of all students
SELECT studentName, departmentId FROM Student;
-- Query 3: Find all students whose department is 'Computer Science'
SELECT * FROM Student WHERE departmentId = 101;
-- Query 4: List all students who joined the college in the year 2020
SELECT * FROM Student WHERE YEAR(studentDOB) = 2000;
-- Query 5: Retrieve the details of students whose names start with the
letter 'A'
SELECT * FROM Student WHERE studentName LIKE 'A%';
-- Query 6: Calculate the average age of all students
SELECT AVG(TIMESTAMPDIFF(YEAR, studentDOB, CURDATE())) AS
Average_Age FROM Student;
```

-- Query 7: Find the total number of students in the college

```sql
 SELECT COUNT(studentId) AS Student_Count FROM Student;
```

-- Query 8: Find the highest number of students in any department

```sql
SELECT departmentName, MAX(studentCount) AS Highest_Student_Count FROM Department GROUP BY departmentName;
```

-- Query 9: Find the count of students in each department

```sql
SELECT d.departmentName AS departmentName, COUNT(s.studentId) AS studentCount

FROM Student s

JOIN Department d ON s.departmentId = d.departmentId

GROUP BY d.departmentName;
```

-- Query 10: Retrieve student names along with their professors' names

```sql
 SELECT s.studentName, p.professorName

FROM Student s

JOIN Professor p ON s.professorName = p.professorName;
```

-- Query 11: Retrieve the names of students who are enrolled in more than one course

```sql
SELECT s.studentName, COUNT(c.courseId) AS courseCount

FROM Student s

JOIN Course c ON s.departmentId = c.departmentId

GROUP BY s.studentName

HAVING COUNT(c.courseId) > 1;
```

-- Query 12: Retrieve the names of students who are not assigned to any course

```
SELECT studentName

FROM Student WHERE departmentId NOT IN (SELECT departmentId FROM Course);
```

-- Query 13: List all students enrolled in the 'Introduction to Programming' course

```
SELECT s.studentName

FROM Student s

JOIN Course c ON s.departmentId = c.departmentId

WHERE c.courseName = 'Introduction to Programming';
```

-- Query 14: Add a new student to the Student table

```
 INSERT INTO Student (studentId, studentName, studentDOB, studentLocation, departmentId, professorName)

VALUES (1006, 'Frank', '2002-06-18', 'Oregon', 101, 'Dr. Smith');
```

-- Query 15: Update the department of a student (change studentId 1001 to be in 'Physics' department)

```
UPDATE Student  SET departmentId = 103 WHERE studentId = 1001;
```

-- Query 16: Delete all students who have not been enrolled for more than 3 years

```
DELETE FROM Student WHERE TIMESTAMPDIFF(YEAR, studentDOB, CURDATE()) > 3;
```

-- Query 17: Create a backup of the Department table

```
CREATE TABLE IF NOT EXISTS Department_Backup AS

SELECT * FROM Department WHERE 1=0;

INSERT INTO Department_Backup

SELECT * FROM Department;
```

-- Query 18: Drop the Department_Backup table

DROP TABLE IF EXISTS Department_Backup;

-- Query 19: Add a unique constraint on the studentName column in the Student table

ALTER TABLE Student ADD CONSTRAINT unique_student_name UNIQUE (studentName);

-- Query 20: Drop a foreign key constraint between Student and Department

ALTER TABLE Student DROP FOREIGN KEY student_ibfk_1;