Assignment 3

Statement : Write the DDL and DML statements for the following.

* Each offering of a course (i.e. a section) can have many Teaching assistants; each teaching assistant is a student. Extend the existing schema(Add/Alter tables) to accommodate this requirement.

-- Each offering of a course (i.e. a section) can have many Teaching

assistants; each teaching assistant is a student.  Extend the existing schema(Add/Alter tables) to accommodate this requirement.

CREATE TABLE IF NOT EXISTS teaching\_assistant (

    StudentID VARCHAR(5),

    CourseID VARCHAR(10),

    SectionID VARCHAR(10),

    Semester VARCHAR(10) CHECK (Semester in ('Fall', 'Winter', 'Spring', 'Summer')),

    Year NUMERIC(4,0) CHECK (Year > 1701 AND Year < 2100),

    PRIMARY KEY (StudentID, CourseID, SectionID, Semester, Year),

    FOREIGN KEY (StudentID) REFERENCES student(ID) ON DELETE CASCADE,

    FOREIGN KEY (CourseID, SectionID, Semester, Year) REFERENCES section(CourseID, SectionID, Semester, Year) ON DELETE CASCADE

);

-- Inserting more teaching assistants for different sections

INSERT INTO teaching\_assistant VALUES

('00128', 'CS-319', '1', 'Spring', '2010'),

('00128', 'BIO-101', '1', 'Summer', '2009'),

('00128', 'CS-101', '1', 'Spring', '2010'),

('00128', 'CS-101', '1', 'Fall', '2009'),

('12345', 'BIO-101', '1', 'Summer', '2009'),

('54321', 'MU-199', '1', 'Spring', '2010'),

('76543', 'CS-101', '1', 'Fall', '2009'),

('98765', 'PHY-101', '1', 'Fall', '2009'),

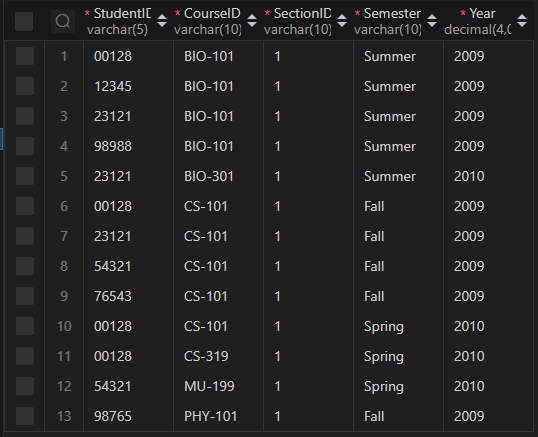
('98988', 'BIO-101', '1', 'Summer', '2009'),

('54321', 'CS-101', '1', 'Fall', '2009'),

('23121', 'BIO-101', '1', 'Summer', '2009'),

('23121', 'BIO-301', '1', 'Summer', '2010'),

('23121', 'CS-101', '1', 'Fall', '2009');



According to the existing schema, one student can have only one advisor.

* Alter the schema to allow a student to have multiple advisors and make sure that you are able to insert multiple advisors for a student.

-- Alter the schema to allow a student to have multiple advisors and make sure

that you are able to insert multiple advisors for a student.

ALTER TABLE advisor DROP FOREIGN KEY advisor\_ibfk\_1;

ALTER TABLE advisor DROP FOREIGN KEY advisor\_ibfk\_2;

ALTER TABLE advisor DROP PRIMARY KEY;

ALTER TABLE advisor ADD PRIMARY KEY (StudentID, InstructorID);

ALTER TABLE advisor ADD FOREIGN KEY (StudentID) REFERENCES student(ID) ON DELETE CASCADE;

ALTER TABLE advisor ADD FOREIGN KEY (InstructorID) REFERENCES instructor(ID) ON DELETE CASCADE;

-- Insert multiple advisors for students

INSERT INTO advisor VALUES

('00128', '10101'),

('00128', '12121'),

('00128', '22222'),

('00128', '32343'),

('12345', '22222'),

('12345', '32343'),

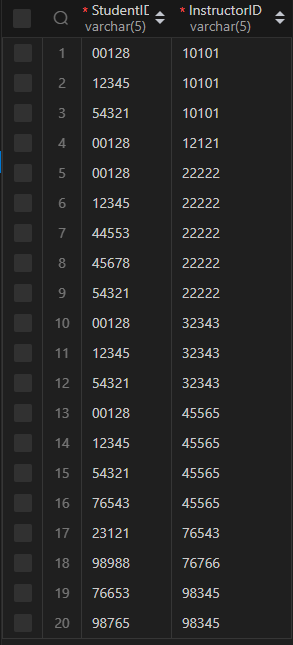
('12345', '45565'),

('54321', '10101'),

('54321', '22222'),

('54321', '32343'),

('54321', '45565');

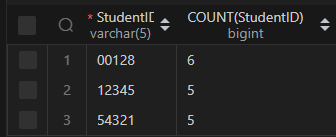


Write SQL queries on the modified schema. You will need to insert data to ensure the query results are not empty.

* Find all students who have more than 3 advisors.

SELECT StudentID, COUNT(StudentID) FROM advisor GROUP BY StudentID HAVING

COUNT(StudentID) > 3;



* Find all students who are co-advised by Prof. Srinivas and Prof. Ashok.

-- Insert data into the instructor table

INSERT INTO instructor VALUES ('22105', 'Ashok', 'Finance', '50000');

-- Insert data into student\_advisor table

INSERT INTO advisor VALUES

('12345', '22105'),

('00128', '22105'),

('54321', '22105');

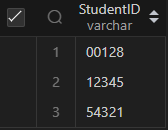
-- Find all students who are co-advised by Prof. Srinivas and Prof. Ashok.

SELECT StudentID FROM advisor

JOIN instructor ON advisor.InstructorID = instructor.ID

WHERE Name = 'Srinivasan' or Name = 'Ashok'

GROUP BY StudentID HAVING COUNT(StudentID) > 1;



* Find students advised by instructors from different departments. etc.

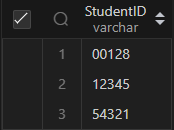
-- Find students advised by instructors from different departments. etc.

SELECT DISTINCT(advisor.StudentID) FROM advisor

JOIN instructor ON InstructorID = ID, advisor as P

JOIN instructor as t on P.InstructorID = T.ID

where advisor.StudentID = P.StudentID and instructor.DepartmentName != T.DepartmentName;



Write SQL queries for the following:

* Delete all information in the database which is more than 10 years old. Add data as necessary to verify your query.

-- Insert data into section table

INSERT INTO section VALUES

('CS-319', '2', 'Spring', '2021', 'Taylor', '3128', 'C'),

('BIO-301', '1', 'Summer', '2020', 'Painter', '514', 'A'),

('CS-190', '1', 'Spring', '2019', 'Taylor', '3128', 'E'),

('CS-101', '1', 'Fall', '2020', 'Packard', '101', 'H'),

('CS-347', '1', 'Fall', '2019', 'Taylor', '3128', 'A'),

('EE-181', '1', 'Spring', '2021', 'Taylor', '3128', 'C'),

('FIN-201', '1', 'Spring', '2020', 'Packard', '101', 'B'),

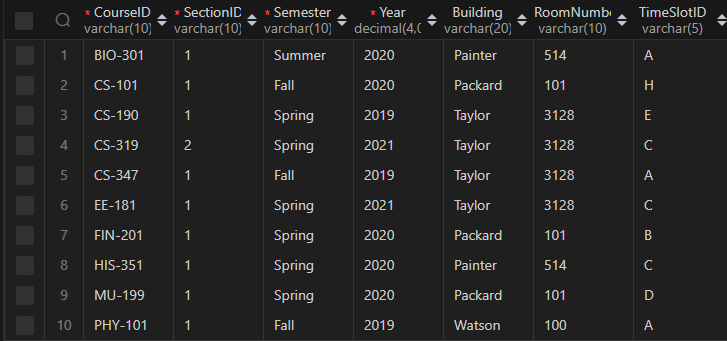
('HIS-351', '1', 'Spring', '2020', 'Painter', '514', 'C'),

('MU-199', '1', 'Spring', '2020', 'Packard', '101', 'D'),

('PHY-101', '1', 'Fall', '2019', 'Watson', '100', 'A');

-- Delete all information in the database which is more than 10 years old. Add data as necessary to verify your query.

DELETE FROM section WHERE Year < YEAR(CURDATE()) - 10;



* Delete the course CS 101.  Any course which has CS 101 as a prereq should remove CS 101 from its prereq set.  Create a cascade constraint to enforce the above rule, and verify that it is working.

-- Add the cascade constraint to the prerequisite table

ALTER TABLE prerequiste DROP FOREIGN KEY prerequiste\_ibfk\_2;

ALTER TABLE prerequiste ADD FOREIGN KEY (PrerequisteID) REFERENCES course(CourseID) ON DELETE CASCADE;

SET FOREIGN\_KEY\_CHECKS = 0;

-- Delete the course CS-101

DELETE FROM course WHERE CourseID = 'CS-101';

