**Individual Assignment 3: SQL DDL and DML**

Answer all questions using SQL statements. Include both **SQL commands** and **screenshots** of the running result (No need to show all rows in the result table; just the first few rows would suffice).

Grading policy:

50% - the correctness of the results;

50% - SQL syntax (partial credit will be given)

**Question 1 (15 points)**

Create the following simplified university database. Provide justification for the choice of data type for each attribute.

STUDENT (StudentID, StudentName, DOB, Major)

COURSE (CourseID, CourseName, Credits)

STUDENT\_COURSE (StudentID, CourseID, EnrollmentDate, Grade)

**Question 2 (15 points)**

Populate the tables you created in Question 1 with the following data:

STUDENT:

Student 1: ID = 101, Name = “Alice Johnson”, DOB = “2002-03-15”, Major = “Computer Science”

Student 2: ID = 102, Name = “Bob Smith”, DOB = “2001-07-22”, Major = “Business”

Student 3: ID = 103, Name = “Catherine Green”, DOB = “2003-11-05”, Major = “Mathematics”

COURSE

Course 1: ID = 501, Name = “Database Systems”, Credits = 4

Course 2: ID = 502, Name = “Software Engineering”, Credits = 3

Course 3: ID = 503, Name = “Marketing”, Credits = 2

STUDENT\_COURSE:

Enrollment 1: Student ID = 101, Course ID = 501, Date = “2024-01-15”, Grade = “A”

Enrollment 2: Student ID = 102, Course ID = 503, Date = “2024-01-17”, Grade = “B+”

Enrollment 3: Student ID = 102, Course ID = 502, Date = “2024-01-20”, Grade = “B”

Enrollment 4: Student ID = 103, Course ID = 502, Date = “2024-01-20”, Grade = “A-”

**Question 3 (20 points)**

Please update the tables to meet the following requirements:

Part A: Updating Data

* Bob Smith decided to change his major to “Marketing.” Write an SQL statement to update his major in the STUDENT table.
* The university has increased the credits for the course “Software Engineering” from 3 to 4. Write an SQL statement to update the credits column in the COURSE table for this course.
* Catherine Green noticed that her Software Engineer course grade was incorrectly recorded as “A-.” Her correct Grade is “A.” Write an SQL statement to update her grade in the STUDENT\_COURSE table.

Part B: Altering Table Structure

* The university has decided to track students' email addresses in the Students table. Write an SQL statement to add a new column EmailAddress (of type varchar(100)) to the STUDENT table.
* After reviewing the table design, it was decided that the column CourseName in the COURSE table should be renamed to CourseTitle. Write the SQL statement to rename this column.
* The university also wants to track the instructors who teach each course. Add a new column InstructorName (of type varchar(50)) to the Courses table.

**Question 4 (50 points)**

For the Fin and Finicky Security Consultants example (ER examples we discussed in class), please use SQL DDL and DML to create and populate this database. Make sure that all necessary foreign key constraints are included. Each table in your database should contain at least 5 records.

USE testdb;

DROP TABLE IF EXISTS STUDENT;

DROP TABLE IF EXISTS COURSE;

DROP TABLE IF EXISTS STUDENT\_COURSE;

#Question 1 (15 points)

#Create the following simplified university database. Provide justification for the choice of data type for each attribute.

CREATE TABLE STUDENT (

StudentID INTEGER PRIMARY KEY, -- This is the primary key and should be a whole number integer

StudentName VARCHAR(100) NOT NULL, -- A students full name, needs to be filled in, otherwise whats the point, 100 characters sufficient length

DOB DATE, -- A date

Major VARCHAR(60) -- we may want to eventually make this into a foriegn key for a seperate table

);

CREATE TABLE COURSE (

CourseID INTEGER PRIMARY KEY, -- Since CoourseIDs in this school are only numeric this is all thats needed

CourseName VARCHAR(200) NOT NULL, -- Course name needs to be filled in, otherwise whats the point, 200 characters sufficient length

Credits SMALLINT -- Most courses are not more than 4 credits but allows for a larger variation

);

CREATE TABLE STUDENT\_COURSE (

StudentID INTEGER NOT NULL, -- From above

CourseID INTEGER NOT NULL, -- From above

EnrollmentDate DATE NOT NULL, -- A date

Grade VARCHAR(2) -- short letter grades (A, A-, B+, …) or NULL in-progress

);

#Question 2 (15 points)

#Populate the tables you created in Question 1 with the following data:

INSERT INTO STUDENT (StudentID, StudentName, DOB, Major) VALUES

(101, 'Alice Johnson', '2002-03-15', 'Computer Science'),

(102, 'Bob Smith', '2001-07-22', 'Business'),

(103, 'Catherine Green', '2003-11-05', 'Mathematics');

INSERT INTO COURSE (CourseID, CourseName, Credits) VALUES

(501, 'Database Systems', 4),

(502, 'Software Engineering', 3),

(503, 'Marketing', 2);

INSERT INTO STUDENT\_COURSE (StudentID, CourseID, EnrollmentDate, Grade) VALUES

(101, 501, '2024-01-15', 'A'),

(102, 503, '2024-01-17', 'B+'),

(102, 502, '2024-01-20', 'B'),

(103, 502, '2024-01-20', 'A-');

SELECT \*

FROM STUDENT;

SELECT \*

FROM COURSE;

SELECT \*

FROM STUDENT\_COURSE;