ECON485 – Homework Assignment 2

Course Registration System – SQL

Part 1 – Task 1a: Create Tables

```
PRAGMA foreign_keys = ON;

DROP TABLE IF EXISTS CompletedCourses;
DROP TABLE IF EXISTS Prerequisite;
DROP TABLE IF EXISTS Enrollment;
DROP TABLE IF EXISTS Section;
DROP TABLE IF EXISTS Instructor;
DROP TABLE IF EXISTS Course;
DROP TABLE IF EXISTS Student;

CREATE TABLE Student (
  student_id INTEGER PRIMARY KEY,
  first_name TEXT NOT NULL,
  last_name  TEXT NOT NULL,
  email      TEXT UNIQUE
);

CREATE TABLE Course (
  course_id INTEGER PRIMARY KEY,
  course_code TEXT NOT NULL UNIQUE,
  course_title TEXT NOT NULL,
  ects INTEGER NOT NULL
);

CREATE TABLE Instructor (
  instructor_id INTEGER PRIMARY KEY,
  full_name TEXT NOT NULL,
  department TEXT NOT NULL
);

CREATE TABLE Section (
  section_id INTEGER PRIMARY KEY,
  course_id INTEGER NOT NULL,
  instructor_id INTEGER NOT NULL,
  section_number TEXT NOT NULL,
  max_capacity INTEGER NOT NULL,
  FOREIGN KEY (course_id) REFERENCES Course(course_id),
  FOREIGN KEY (instructor_id) REFERENCES Instructor(instructor_id)
);
```

```
CREATE TABLE Enrollment (
  enrollment_id INTEGER PRIMARY KEY,
  student_id INTEGER NOT NULL,
  section_id INTEGER NOT NULL,
  registration_date TEXT DEFAULT (datetime('now')),
  FOREIGN KEY (student_id) REFERENCES Student(student_id),
  FOREIGN KEY (section_id) REFERENCES Section(section_id),
  UNIQUE (student_id, section_id)
);
```

PART 1 – Task 1b: Insert Example Data

```
INSERT INTO Course VALUES
(1,'ECON201','Microeconomics I',6),
(2,'ECON301','Econometrics I',6),
(3,'STAT101','Introduction to Statistics',5);

INSERT INTO Instructor VALUES
(1,'Dr. Ayşe Demir','Economics'),
(2,'Dr. Mehmet Kaya','Economics'),
(3,'Dr. Elif Yılmaz','Statistics');

INSERT INTO Section VALUES
(101,1,1,'01',30),
(102,2,2,'01',30),
(103,3,3,'01',40);

INSERT INTO Student VALUES
(1,'Ali','Yıldız','ali@uni.edu'),
(2,'Zeynep','Kara','zeynep@uni.edu'),
(3,'Mert','Şahin','mert@uni.edu'),
(4,'Elif','Aydın','elif@uni.edu'),
(5,'Deniz','Koç','deniz@uni.edu'),
(6,'Ceren','Aslan','ceren@uni.edu'),
(7,'Efe','Arslan','efe@uni.edu'),
(8,'Selin','Çelik','selin@uni.edu'),
(9,'Burak','Öztürk','burak@uni.edu'),
(10,'Ece','Demir','ece@uni.edu');
```

Part 1 – Task 1c: Add and Drop Actions

```
INSERT INTO Enrollment(student_id, section_id) VALUES
(1,101),(2,101),(3,101),(4,101),(5,101),(6,101),(10,101);
```

```
INSERT INTO Enrollment(student_id, section_id) VALUES
(2,102),(3,102),(4,102),(7,102),(8,102),(9,102),(10,102);

INSERT INTO Enrollment(student_id, section_id) VALUES
(1,103),(5,103),(6,103),(7,103),(8,103),(9,103);

DELETE FROM Enrollment WHERE student_id = 6 AND section_id = 101;
DELETE FROM Enrollment WHERE student_id = 9 AND section_id = 102;
DELETE FROM Enrollment WHERE student_id = 1 AND section_id = 103;
```

Part 1 – Task 1d: Show Final Registrations

```
SELECT
  s.first_name || ' ' || s.last_name AS student_name,
  c.course_code,
  sec.section_number,
  e.registration_date
FROM Enrollment e
JOIN Student s ON s.student_id = e.student_id
JOIN Section sec ON sec.section_id = e.section_id
JOIN Course c ON c.course_id = sec.course_id
ORDER BY c.course_code, sec.section_number, student_name;
```

Part 2 – Task 2b: Prerequisite Tables

```
CREATE TABLE Prerequisite (
  prerequisite_id INTEGER PRIMARY KEY,
  course_id INTEGER NOT NULL,
  required_course_id INTEGER NOT NULL,
  minimum_grade TEXT NOT NULL,
  FOREIGN KEY (course_id) REFERENCES Course(course_id),
  FOREIGN KEY (required_course_id) REFERENCES Course(course_id)
);

CREATE TABLE CompletedCourses (
  completed_id INTEGER PRIMARY KEY,
  student_id INTEGER NOT NULL,
  course_id INTEGER NOT NULL,
  letter_grade TEXT NOT NULL,
  completed_term TEXT,
  FOREIGN KEY (student_id) REFERENCES Student(student_id),
  FOREIGN KEY (course_id) REFERENCES Course(course_id),
  UNIQUE (student_id, course_id)
);
```

Part 2 – Task 2c: Assistive SQL Queries

```
SELECT
  c.course_code AS target_course,
  rc.course_code AS prerequisite_course,
  p.minimum_grade
FROM Prerequisite p
JOIN Course c ON c.course_id = p.course_id
JOIN Course rc ON rc.course_id = p.required_course_id
WHERE p.course_id = :course_id;

SELECT
  s.student_id,
  s.first_name || ' ' || s.last_name AS student_name,
  rc.course_code AS prerequisite_course,
  cc.letter_grade AS student_grade,
  p.minimum_grade,
  CASE
    WHEN cc.letter_grade IS NULL THEN 'NO'
    WHEN
      (CASE
        WHEN cc.letter_grade = 'A' THEN 4
        WHEN cc.letter_grade = 'B' THEN 3
        WHEN cc.letter_grade = 'C' THEN 2
        WHEN cc.letter_grade = 'D' THEN 1
        ELSE 0
      END)
      >=
      (CASE
        WHEN p.minimum_grade = 'A' THEN 4
        WHEN p.minimum_grade = 'B' THEN 3
        WHEN p.minimum_grade = 'C' THEN 2
        WHEN p.minimum_grade = 'D' THEN 1
        ELSE 0
      END)
    THEN 'YES'
    ELSE 'NO'
  END AS prerequisite_met
FROM Prerequisite p
JOIN Course rc ON rc.course_id = p.required_course_id
JOIN Student s ON s.student_id = :student_id
LEFT JOIN CompletedCourses cc
  ON cc.student_id = s.student_id
 AND cc.course_id = p.required_course_id
WHERE p.course_id = :course_id;
```