HW4

1 Problem 1

Our database is designed to support course selection planning, with two primary entity sets: "courses" and "students". These include key attributes such as "section" and "major", enabling filtering and matching capabilities. Additional entity and relationship sets are incorporated to enforce constraints and provide comprehensive filtering across various aspects.
The "requirements" entity set defines the conditions students must meet to graduate and earn a degree in their chosen major. Requirements are standardized into attributes such as included courses and units required for fulfillment. These requirements are categorized and linked to "courses" via the "courses_requires" relationship set.
To manage course dependencies, we define two relationship sets: "pre-requisite" and "anti-requisite", which respectively establish the sequencing and exclusion constraints between courses.
Additionally, "taken" and "to be taken" sets track courses a student has completed or plans to take. These sets ensure course selection aligns with requirements and respects constraints from previously taken courses.

2 Problem 2

2.1 Relational schema:

Courses_info(<u>title</u>, type_id, units)
Courses(<u>course_id</u>, instructor, title, semester, session_id, days, start_time, end_time, status_id)
Students(<u>id</u>, major, subject_id)
Requirements(<u>requirement_id</u>, units_needed, requirement_type)
Courses_requirement(<u>course_id</u>, <u>requirement_id</u>, major, subject_id)
Pre_requisite(id_first, id_second, <u>id_or</u>)
Anti_requisite(<u>id_1</u>, <u>id_2</u>)
Courses_taken(<u>course_id</u>, <u>student_id</u>)
Courses_to_be_taken(<u>student_id</u>, <u>course_id</u>, semester, session_id)
Plan(<u>student_id</u>, <u>course_id</u>)

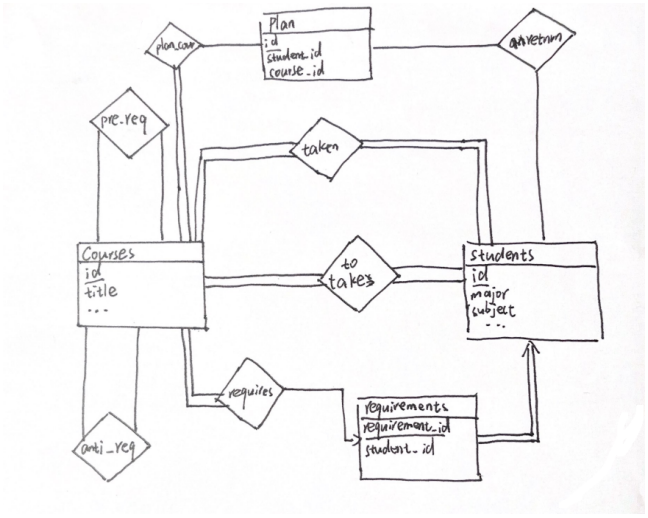

Figure 1: ER diagram

2.2 Explanation:

2.2.1 Entity Sets:

Entity Set: Courses
We convert the entity set **Courses** into two relations: Courses_info, Courses.

(1)
Entity Set: Courses
Primary Key: title
Relational schema: Courses_info(title, type_id, units)
Explanation: The Courses_info relation stores general information about each course. Title is the primary key.

```
CREATE TABLE Courses_info (
    title VARCHAR(20) PRIMARY KEY,
    type_id VARCHAR(4),
    units NUMERIC(2,1) CHECK (units >= 0 AND units <= 4)
);
```

(2)
Entity Set: Courses
Primary Key: course_id
Foreign Key: title references Courses_info(title)
Relational schema: Courses(course_id, instructor, title, semester, session_id, days, start_time, end_time, status_id)
Explanation: The Courses relation stores detailed information about each

course. The primary key is course_id. The title column in the Courses table is a foreign key that references the title column in the Courses_info table.

```
CREATE TABLE Courses (
    course_id INT PRIMARY KEY,
    instructor VARCHAR(25),
    title VARCHAR(20),
    semester VARCHAR(10),
    session_id INT,
    days VARCHAR(12),
    start_time TIME,
    end_time TIME,
    status_id VARCHAR(10),
    FOREIGN KEY (title) REFERENCES Courses_info(title)
);
```

Entity Set: Students
Primary Key: id
Relational schema: Students(id, major, subject_id)
Explanation: The Students relation stores student information. The primary key is id.

```
CREATE TABLE Students (
    id VARCHAR(8) PRIMARY KEY,
    major VARCHAR(64),
    subject_id VARCHAR(15)
);
```

Entity Set: requirements
Primary Key: requirement_id
Relational schema: Requirements(requirement_id, units_needed, requirement_type)
Explanation: Stores the graduation requirements for students.

```
CREATE TABLE Requirements (
    requirement_id INT PRIMARY KEY,
    units_needed NUMERIC(4,1),
    requirement_type VARCHAR(20)
);
```

2.2.2 Relationship Sets

Relationship Set: courses_requires
Composite Primary Key: (course_id, requirement_id)
Foreign Keys: course_id references Courses(course_id), requirement_id references Requirements(requirement_id)

Relational schema: Courses_requirement(course_id, requirement_id, major, subject_id)
Explanation: Stores which courses satisfy certain graduation requirements. The requirement_id column in the Course_requirement table is a foreign key that references the requirement_id column in the Requirements table.

```
CREATE TABLE Courses_requirement (
    course_id INT,
    requirement_id INT,
    major VARCHAR(64),
    subject_id VARCHAR(15),
    PRIMARY KEY (course_id, requirement_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id),
    FOREIGN KEY (requirement_id) REFERENCES Requirements(requirement_id)
);
```

Relationship Set: pre_requisite
Primary Key: id_or
Foreign Keys: id_first references Courses_info(title), id_second references Courses_info(title)
Relational schema: Pre_requisite(id_first, id_second, id_or)
Explanation: Stores prerequisite relationships between courses.

```
CREATE TABLE Pre_requisite (
    id_first VARCHAR(20),
    id_second VARCHAR(20),
    id_or INT PRIMARY KEY,
    FOREIGN KEY (id_first) REFERENCES Courses_info(title),
    FOREIGN KEY (id_second) REFERENCES Courses_info(title)
);
```

Relationship Set: anti-requisite
Composite Primary Key: (id_1, id_2)
Foreign Keys: id_1 references Courses_info(title), id_2 references Courses_info(title)
Relational schema: Anti_requisite(id_1, id_2)
Explanation: Stores anti-requisite relationships between courses.

```
CREATE TABLE Anti_requisite (
    id_1 VARCHAR(20),
    id_2 VARCHAR(20),
    PRIMARY KEY (id_1, id_2),
    FOREIGN KEY (id_1) REFERENCES Courses_info(title),
    FOREIGN KEY (id_2) REFERENCES Courses_info(title)
);
```

Relationship Set: Taken
Composite Primary Key: (course_id, student_id)

Foreign Keys: course_id references Courses(course_id), student_id references Students(id)
Relational schema: Courses_taken(course_id, student_id)
Explanation: Tracks courses each student has taken.

```
CREATE TABLE Courses_taken (
    course_id INT,
    student_id VARCHAR(8),
    PRIMARY KEY (course_id, student_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id),
    FOREIGN KEY (student_id) REFERENCES Students(id)
);
```

Relationship Set: to be taken
Composite Primary Key: (student_id, course_id)
Foreign Keys: course_id references Courses(course_id), student_id references Students(id)
Relational schema: Courses_to_be_taken(student_id, course_id, semester, session_id)
Explanation: Stores courses to be taken by each student.

```
CREATE TABLE Courses_to_be_taken (
    student_id VARCHAR(8),
    course_id INT,
    semester VARCHAR(10),
    session_id INT,
    PRIMARY KEY (student_id, course_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id),
    FOREIGN KEY (student_id) REFERENCES Students(id)
);
```

Relationship Set: Plan
Composite Primary Key: (student_id, course_id)
Foreign Keys: course_id references Courses(course_id), student_id references Students(id)
Relational schema: Plan(student_id, course_id)
Explanation: Stores course selection plan of each student.

```
CREATE TABLE Plan (
    student_id VARCHAR(8),
    course_id INT,
    PRIMARY KEY (student_id, course_id),
    FOREIGN KEY (course_id) REFERENCES Courses(course_id),
    FOREIGN KEY (student_id) REFERENCES Students(id)
);
```

2.3 Visualization: