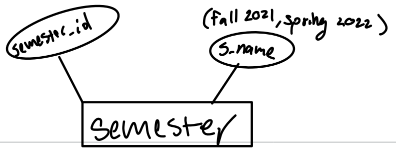
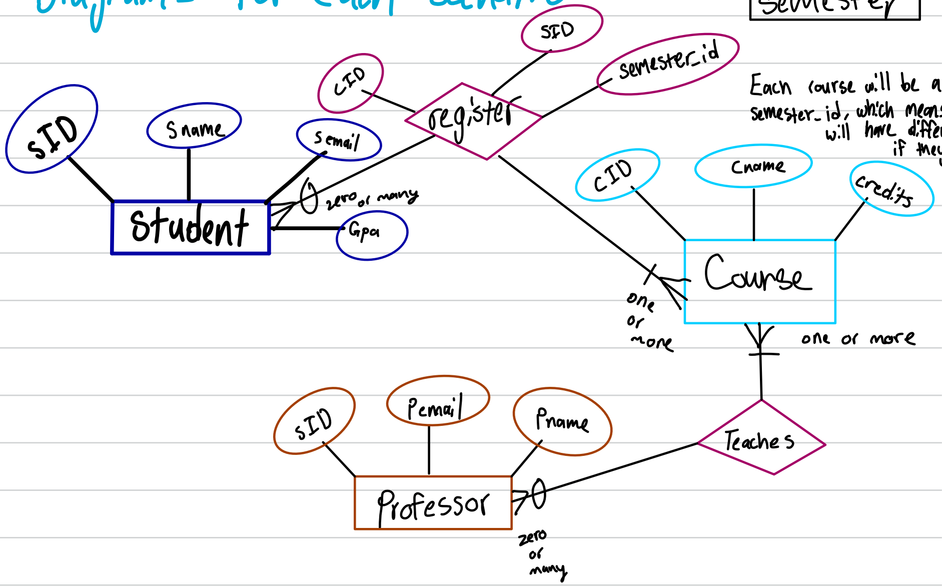
ER Diagrams for each scenario

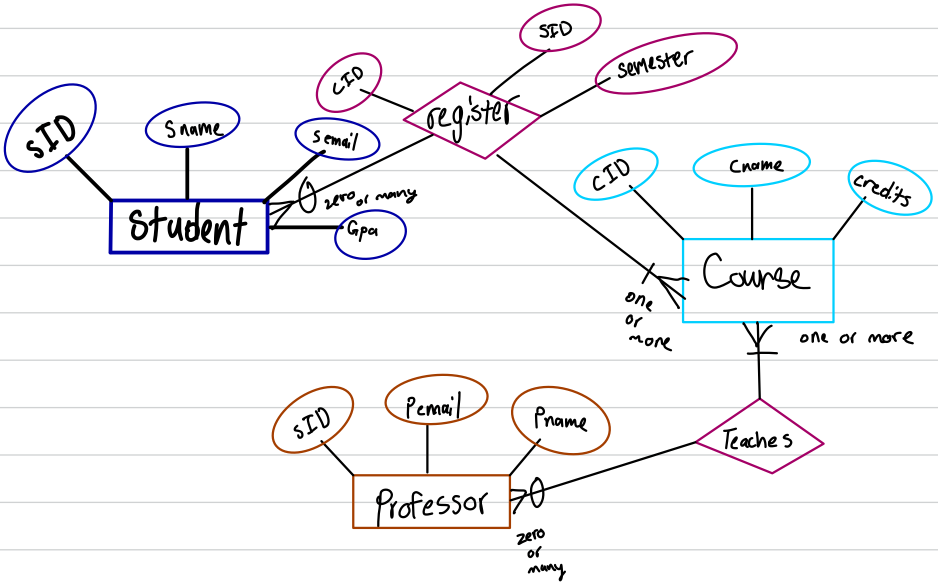


1.)

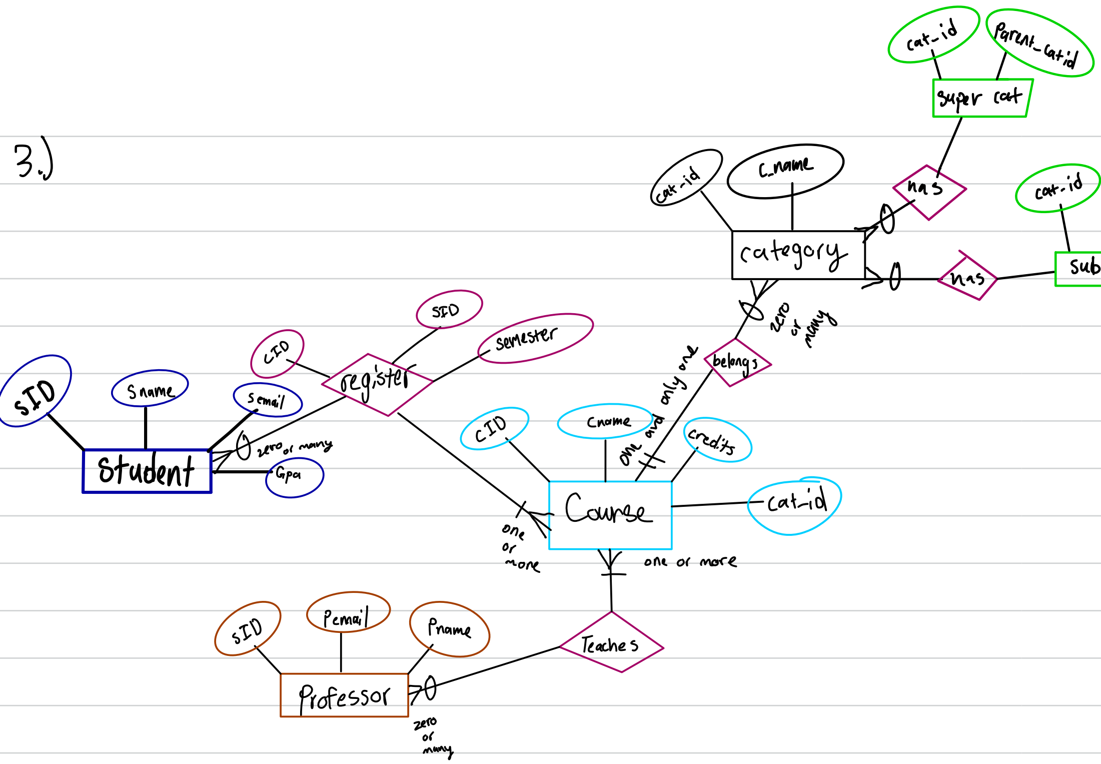


Each course will be assigned to a semester_id, which means same courses will have different semester ids if they're in different semesters.

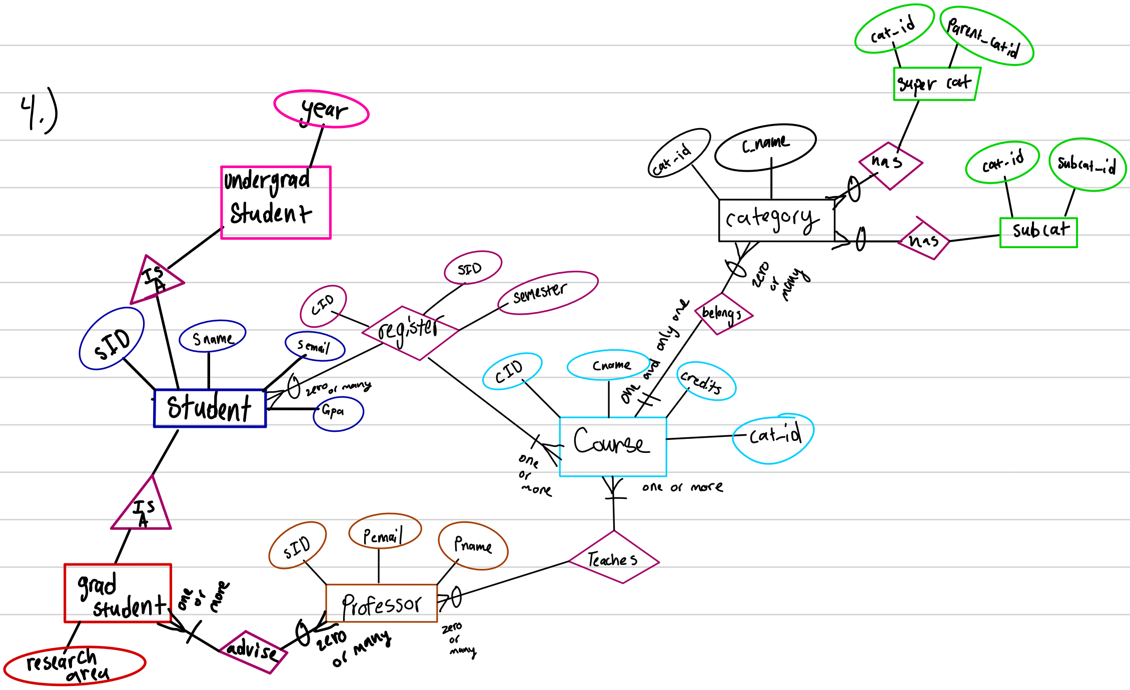
2.)



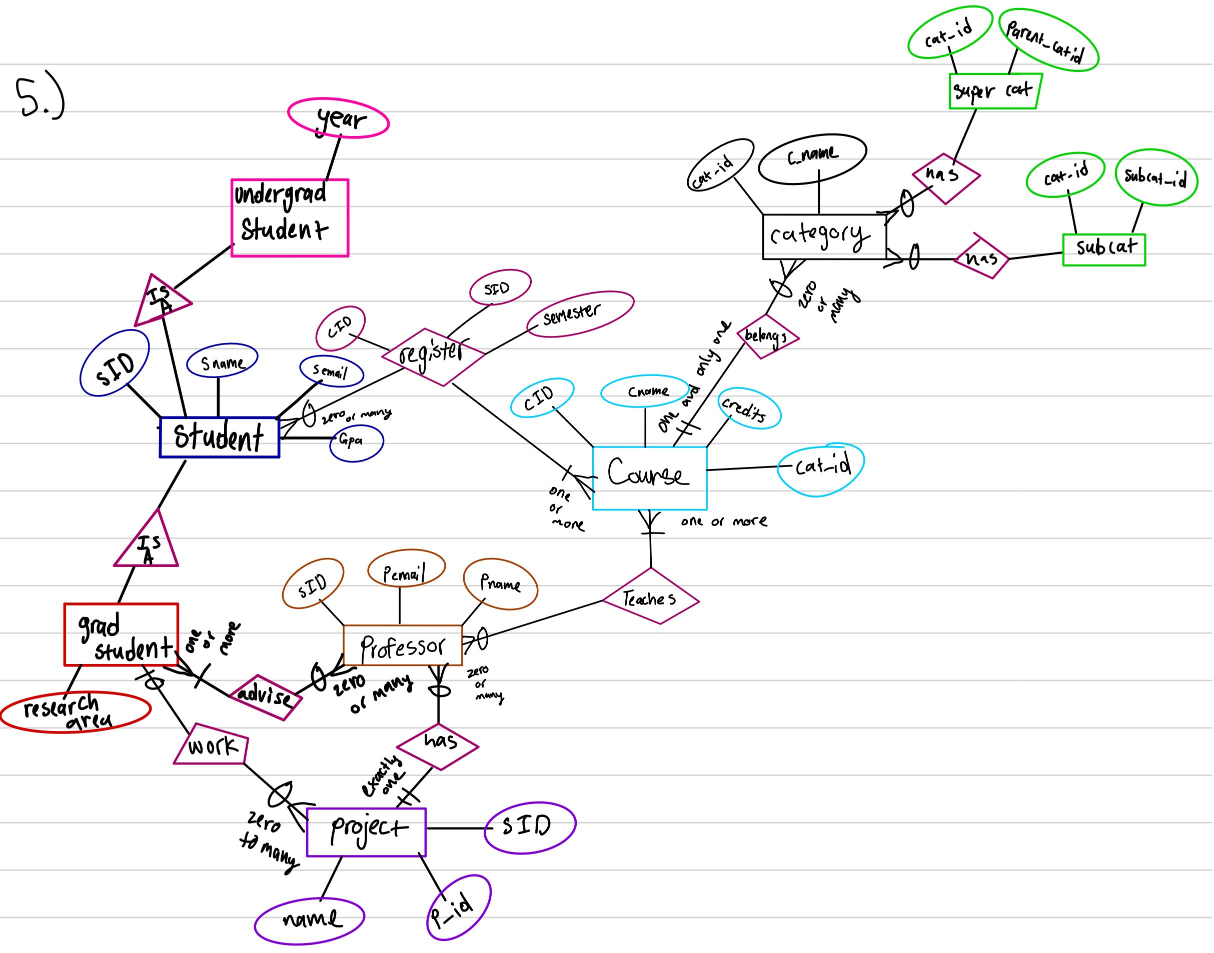
3.)


cat_id
Parent_cat_id
super cat
has
cat-id
c_name
category
cat-id
subcat_id
sub cat
has
zero or many
belongs
one and only one
SID
semester
CID
register
SID
Sname
Semail
Student
zero or many
Gpa
CID
Cname
credits
Course
cat_id
one or more
one or more
SID
Pemail
Pname
Teaches
Professor
zero or many

4.)


year
Undergrad Student
IS A
cat-id
Parent_cat_id
super cat
has
cat-id
c_name
category
cat-id
Subcat_id
subcat
has
zero or many
belongs
one and only one
SID
semester
CID
register
SID
Sname
Semail
Student
zero or many
Gpa
CID
Cname
credits
Course
cat_id
one or more
one or more
IS A
SID
Pemail
Pname
Teaches
grad student
one or more
Professor
zero or many
advise
zero or many
research area

5.)


year
undergrad Student
IS A
SID
Sname
Semail
Student
zero or many
Gpa
CID
SID
register
semester
IS A
grad student
research area
one or more
advise
zero or many
work
zero to many
project
SID
name
P_id
exactly one
has
zero or many
SID
Pemail
Pname
Professor
Teaches
one or more
Course
CID
Cname
credits
cat_id
one or more
one and only one
belongs
zero or many
category
cat-id
c_name
has
super cat
cat_id
Parent_cat_id
has
subcat
cat-id
Subcat_id

Part B.) SQL statements of each scenario

1.)
```
CREATE TABLE Professors(
    staff_id: INT
    P_email: CHAR(200)
    P_name: CHAR(200)
    PRIMARY KEY(staff_id)
)
```

```
CREATE TABLE Courses (
    course_id: INT
    c-name: CHAR(200)
    credits: INT
    PRIMARY KEY course-id)
)
```

```
CREATE TABLE Student (
    student_id: INT
    s_email: CHAR(200)
    s_name: CHAR(200)
    PRIMARY KEY(student_id)
)
```

```
CREATE TABLE Teaches (
    course-id: INT
    staff_id: INT NOT NULL
    PRIMARY KEY(course_id, staff_id)
    FOREIGN KEY(course_id)
    REFERENCES Courses
    FOREIGN KEY(staff_id)
    REFERENCES Professors
)
```

```
CREATE TABLE Register (
    course_id: INT
    student_id: INT NOT NULL
    semester_id: INT
    PRIMARY KEY(course-id, student-id)
    FOREIGN KEY(course_id) REFERENCES courses
    FOREIGN KEY(student_id) REFERENCES Student
    FOREIGN KEY(semester_id) REFERENCES semester
```

```
CREATE TABLE Semester (
    semester_id: INT
    sem_name: CHAR(200)
    PRIMARY KEY(semester_id)
)
```
//this semester table allows students to register for the same class aslong as they are different semesters (using semester_id)

2.) same relational model as $\overset{\text{question}}{\checkmark}$ 1 but now we remove the "semester" table and edit the "Register" table as follows:

```
CREATE TABLE Register (
    course_id: INT
    Student_id: INT NOT NULL
    Semester: CHAR(200)
     FOREIGN KEY(course_id) REFERENCES courses
    FOREIGN KEY(Student_id) REFERENCES Student
 )
```

OR...

```
CREATE TABLE Register (
    course_id: INT
    Student_id: INT NOT NULL

     FOREIGN KEY(course_id) REFERENCES courses
    FOREIGN KEY(Student_id) REFERENCES Student
 )
```

Now when a student registers for a course, since it will only take account the course_id and student_id, if the student_id and course_id are the same, it'll only be able to apply a unique course id once rather than multiple times to a specific student_id, since it wont have to worry about a changing semester_id.

3.) same relational model as question 2, but with a newly added "category" table, with super and subcategory tables.
the "courses" table was also edited to enforce that each course must belong to one category

```
CREATE TABLE category (
    cat_id: INT
    cat-name: CHAR(200)
    PRIMARY KEY(cat-id)
)
```

```
CREATE TABLE Courses (
    course_id: INT
    c-name: CHAR(200)
    credits: INT
    PRIMARY KEY(course-id)
    cat_id: INT NOT NULL
    FOREIGN KEY (cat_id)
    REFERENCES Category
)
```

```
CREATE TABLE Super_cat (
    Cat_id: INT
    Parentcat_id: INT
    PRIMARY KEY (Parentcat_id)
    FOREIGN KEY (cat_id) REFERENCES Category
)
```

```
CREATE TABLE sub_cat (
    Cat_id: INT
    subcat-id: INT
    PRIMARY KEY (subcat-id)
    FOREIGN KEY (cat_id) REFERENCES Category
)
```

4.) Same as relational model from question 3 but now we have an "IS A" for student table, 2 actually. An undergrad student and a grad student.

```
CREATE TABLE Student (
    student_id : INT
    s_email : CHAR(200)
    s_name : CHAR(200)
    PRIMARY KEY (student_id)
)
```

```
CREATE TABLE Grad_Student (
    research_area : CHAR(200)
    Student_id : INT
    S_email : CHAR(200)
    S_name : CHAR(200)
    PRIMARY KEY (Student_id)
    FOREIGN KEY (Student_id) REFERENCES Student
        ON DELETE CASCADE
    FOREIGN KEY (S_email ) REFERENCES Student
        ON DELETE CASCADE
    FOREIGN KEY (S_name ) REFERENCES Student
        ON DELETE CASCADE
)
```

```
CREATE TABLE Undergrad Student (
    year : INT
    Student_id : INT
    S_email : CHAR(200)
    S_name : CHAR(200)
    FOREIGN KEY (Student_id) REFERENCES Student
        ON DELETE CASCADE
    FOREIGN KEY (S_email ) REFERENCES Student
        ON DELETE CASCADE
    FOREIGN KEY (S_name ) REFERENCES Student
        ON DELETE CASCADE
    PRIMARY KEY (Student_id)
)
```

```
CREATE TABLE Professors (
    staff_id : INT
    P_email : CHAR(200)
    P_name : CHAR(200)
    PRIMARY KEY (Staff_id)
)
```

```
CREATE TABLE Advise (
    Student_id : INT
    Staff_id : INT NOT NULL
    FOREIGN KEY (student_id) REFERENCES
        Grad_Student
    FOREIGN KEY (Staff_id) REFERENCES
        Professor
)
```

5.) same relational model as question 4
but now we are introducing another new table
called "projects"

```
CREATE TABLE Grad_Student (
    research_area : CHAR(200)
    student_id: INT
    s_email: CHAR(200)
    s_name: CHAR(200)
    PRIMARY KEY(Student_id)
  FOREIGN KEY(student_id) REFERENCES Student
                          ON DELETE CASCADE
  FOREIGN KEY(s_email ) REFERENCES Student
                          ON DELETE CASCADE
  FOREIGN KEY(s_name ) REFERENCES Student
                          ON DELETE CASCADE
  P_id: INT
  FOREIGN KEY(P_id) REFERENCES Project
)
```

```
CREATE TABLE Project (
    P_name: CHAR(200)
    P_id: INT
    student_id: INT
    staff_id: INT NOT NULL
    PRIMARY KEY(P_id)
    FOREIGN KEY(staff_id)
        REFERENCES Professors
           ON DELETE CASCADE
)
```

```
CREATE TABLE Professors(
    staff_id: INT
    P_email: CHAR(200)
    P_name: CHAR(200)
    PRIMARY KEY(staff_id)
)
```