Create Tables

Professor, Counselor, Student, Administrator, Section, Course, Prerequisites, Academic Department, Has_Taken, User

Attributes

User - (AUTO INCREMENT)(PK)(INT) User_ID, FName (VARCHAR(50)), MName (VARCHAR(50)), LName (VARCHAR(50)), (DATE)DOB, (FK)(VARCHAR(4))Dept_name

Professor - (FK)ID
Counselor - (FK) ID
Student - (FK) ID, (FK) CounselorID
Dean - (FK) ID

ENROLLED_COURSES - Student_ID, Sec_ID(FK)

Section - (Course_ID VARCHAR(12) , (PK) Section_ID (INT AUTOINCREMENT NOT NULL PRIMARY KEY), Section_Number CHAR(3) NOT NULL Professor_ID INT, Max_students INT, Current_students_taking INT)

Course - (PK)(VARCHAR(12))COURSE_ID, (VARCHAR(50))NAME, (INT)CREDIT_HOURS, (FK)(VARCHAR(4))D_NAME, COREQ(VARCHAR(12))

Prerequisites - course_id(FK, PK), prereq_id(PK, FK), VARCHAR(2) REQUIRED_GRADE

Academic Department - (PK)(VARCHAR(4))D_NAME

Has_Taken - VARCHAR(2)GRADE, (FK)Section_ID, (FK)Student_ID

User_Accounts - VARCHAR(15) Username, VARCHAR(100) Hash_Password, (FK) ID_user-symbol

Administrator - Admin_ID(FK)

```
CREATE TABLE USER
        (User_ID                INT                     AUTO_INCREMENT NOT
NULL PRIMARY KEY,
         FName                  VARCHAR(50)             NOT NULL,
         MName                  VARCHAR(50)             DEFAULT “N/A”,
         LName                  VARCHAR(50)             NOT NULL,
         DOB                    DATE,
         Dept_Name              VARCHAR(4),
        FOREIGN KEY(Dept_Name) REFERENCES ACADEMIC_DEPARTMENT(D_Name)
        ON DELETE SET NULL ON UPDATE CASCADE);

CREATE TABLE PROFESSOR
        (Professor_ID           INT             PRIMARY KEY,
        FOREIGN KEY(Professor_ID) REFERENCES USER(User_ID)
        ON DELETE CASCADE ON UPDATE CASCADE);
```

```
CREATE TABLE COUNSELOR
        (Counselor_ID           INT                 PRIMARY KEY,
        FOREIGN KEY(Counselor_ID) REFERENCES USER(User_ID)
        ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE STUDENT
        (Student_ID             INT         PRIMARY KEY,
        Counselor_ID            INT,
        FOREIGN KEY(Counselor_ID) REFERENCES COUNSELOR(Counselor_ID)
        ON DELETE SET NULL ON UPDATE CASCADE,
        FOREIGN KEY(Student_ID) REFERENCES USER(User_ID)
        ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE DEAN
        (Dean_ID                INT         PRIMARY KEY,
        FOREIGN KEY(Dean_ID) REFERENCES USER(User_ID)
        ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE ENROLLED_COURSES
        (Student_ID             INT         NOT NULL,
         Section_ID             INT         NOT NULL,
        PRIMARY KEY (Student_ID, Section_ID),
        FOREIGN KEY(Section_ID) REFERENCES SECTION(Section_ID)
        ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY(Student_ID) REFERENCES USER(User_ID)
        ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE SECTION
        (Course_ID        VARCHAR(12) ,
         Section_ID                   INT           AUTO_INCREMENT NOT NULL
PRIMARY KEY,
        Section_Number    CHAR(3)           NOT NULL,
         Professor_ID                 INT,
         Max_students                 INT,
        Current_students_taking       INT,
        FOREIGN KEY(Course_ID) REFERENCES COURSE(Course_ID)
        ON DELETE  CASCADE ON UPDATE CASCADE,
        FOREIGN KEY(Professor_ID) REFERENCES USER(User_ID)
ON DELETE SET NULL ON UPDATE CASCADE
        );
```

```
CREATE TABLE COURSE
        (Course_ID                      VARCHAR(12)                     NOT NULL PRIMARY KEY,
         Name                           VARCHAR(50)                     NOT NULL,
         Credit_Hours                   INT                             NOT NULL,
         D_Name                         VARCHAR(4)                      NOT NULL,
        Corequisite_ID                  VARCHAR(12),
        FOREIGN KEY(Corequisite_ID) REFERENCES COURSE(Course_ID)
        ON DELETE  SET NULL ON UPDATE CASCADE,
        FOREIGN KEY(D_Name) REFERENCES ACADEMIC_DEPARTMENT(D_Name)
        ON DELETE RESTRICT ON UPDATE CASCADE);

CREATE TABLE PREREQUISITES
        (Course_ID                              VARCHAR(12)                     NOT NULL PRIMARY
KEY,
         REQUIRED_GRADE                         VARCHAR(2)                      NOT NULL,
         Prerequisite_ID                        VARCHAR(12)                     NOT NULL PRIMARY
KEY,
        FOREIGN KEY(Course_ID) REFERENCES COURSE(Course_ID)
        ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY(Prerequisite_ID) REFERENCES COURSE(Course_ID)
        ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE ACADEMIC_DEPARTMENT
        (D_Name                         VARCHAR(4)                      NOT NULL PRIMARY KEY,
        Dean_ID                         INT,
        FOREIGN KEY(Dean_ID) REFERENCES DEAN(Dean_ID)
        ON DELETE SET NULL
        ON UPDATE CASCADE);

CREATE TABLE HAS_TAKEN
        (Student_ID                             INT                             ,
        Course_ID                               VARCHAR(12)                     ,
         Grade                  VARCHAR(2)                      NOT NULL,
PRIMARY KEY(Student_ID, Course_ID),
        FOREIGN KEY(Student_ID) REFERENCES STUDENT(Student_ID)
        ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY(Course_ID) REFERENCES COURSE(Course_ID)
        ON DELETE RESTRICT ON UPDATE CASCADE);
```

```
CREATE TABLE USER_ACCOUNT
        (Username                       VARCHAR(15)             ,
         Password_Hash                  VARCHAR(100)            NOT NULL,
        User_ID                         INT,
        PRIMARY KEY(Username),
        FOREIGN KEY(User_ID) REFERENCES USER(User_ID)
        ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE ADMINISTRATOR
(Admin_ID           INT             PRIMARY KEY,
PRIMARY KEY(Admin_ID),
FOREIGN KEY(Admin_ID) REFERENCES USER(User_ID)
ON DELETE CASCADE ON UPDATE CASCADE);
```

ALTER COMMANDS

```
ALTER TABLE USER
ADD FOREIGN KEY(Dept_Name) REFERENCES ACADEMIC_DEPARTMENT(D_Name)
        ON DELETE SET NULL
        ON UPDATE CASCADE;
```

INSERT COMMANDS

(**Inserting Users)**

```
INSERT INTO USER
VALUES
(NULL, "Sherman", NULL, "Briar", '1975-05-25', "CS"),
(NULL, "Priya", "M", "Celso", '1962-08-16', "A&H"),
(NULL, "Lanzo", "V", "Gwendal", '1980-11-05', "BBS"),
(NULL, "David", "A", "Olufunke", '1968-09-30', "SOM"),
(NULL, "Janne", "B", "Zafar", '1974-06-25', "IS"),
(NULL, "Denton", "O", "Anwar", '1963-02-12', "NSM")
(NULL, “Jared”, “H”, “Hanes”, 1950-06-01”, NULL);
(NULL, "Jay", 'R', 'Areih', '1976-03-02', 'CS'),
(NULL, 'Diane', 'O', 'Jure', '1965-05-12', 'SOM'),
(NULL, 'Suljo', 'I', 'Teboho', '1945-12-23', 'BBS'),
(NULL, 'Kilikina', 'S', 'Kelly', '1953-04-26', 'A&H'),
(NULL, 'Dennis', 'E', 'Ipati', '1960-01-21', 'IS'),
(NULL, 'Chimo', NULL, 'Donndubhan', '1976-06-04', 'NSM');
```

(**Inserting Administrator)**

```
INSERT INTO ADMINISTRATOR
```

```
VALUES
(19);
```

(Inserting professors)

```
        INSERT INTO PROFESSOR
        VALUES
        (13),
        (14),
        (15),
        (16),
        (17),
        (18);
```

(Inserting Counselors)

```
INSERT INTO COUNSELOR
VALUES (44), (45), (46), (47), (48), (49);
```

(Inserting Students)

```
        INSERT INTO STUDENT
        VALUES
        (38, 45),
        (39, 44),
        (40, 44),
        (41, 48),
        (42, 46),
        (43, 46);
```

(Inserting Deans)

```
        INSERT INTO USER (FName, MName, LName, DOB, Dept_name)
        VALUES ("Paulius", "T", "Jasmina", "1958-01-29", "CS"),
        ("Lucia", "P", "Misa", "1973-05-11", "SOM"),
        ("Ofir", NULL, "Pantaleon", "1950-12-02", "BBS"),
        ( "Aleksandra", "O", "Gayatri", "1972-02-12", "A&H"),
        ("Rong", "T", "Odette", "1964-07-05", "IS"),
        ("Alex", "D", "Stacee", "1970-09-12", "NSM");
```

```
INSERT INTO DEAN (Dean_ID)
VALUES (7),
(8),
(9),
(10),
(11),
(12);

UPDATE ACADEMIC_DEPARTMENT
SET Dean_ID=7
WHERE D_Name=”CS”;
UPDATE ACADEMIC_DEPARTMENT
SET Dean_ID=8
WHERE D_Name=”SOM”;
UPDATE ACADEMIC_DEPARTMENT
SET Dean_ID=9
WHERE D_Name=”BBS”;
UPDATE ACADEMIC_DEPARTMENT
SET Dean_ID=10
WHERE D_Name=”A&H”;
UPDATE ACADEMIC_DEPARTMENT
SET Dean_ID=11
WHERE D_Name=”IS”;
UPDATE ACADEMIC_DEPARTMENT
SET Dean_ID=12
WHERE D_Name=”NSM”;
```

(Inserting Enrolled_Courses)

```
INSERT INTO ENROLLED_COURSES (Student_ID, Section_ID)
VALUES
(42, 1),
(39, 1),
(40, 1),
(41, 1);
```

(Inserting Course)

```
INSERT INTO COURSE (Course_ID, Name, Credit_Hours, D_Name, Corequisite_ID)
VALUES
("GOVT 2306", "State Nameand Local Government", 3, "IS", NULL),
("AHST 1101", "Introduction to Art History", 3, "A&H", NULL),
("CS 4347", "Database Systems", 3, "CS", NULL),
("ECS 4308", "Technical Communications", 3, "CS", NULL),
("MIS 6308", "System Analysis and Project Management", 3, "SOM", NULL),
("NSC 3344", "Anatomy and Physiology of Speech and Hearing", 3, "BBS", NULL),
("CS 4392", "Computer Animation", 3, "CS", NULL),
("MATH 2418", "Linear Algebra", 4, "NSM", NULL),
("CS 4361", "Computer Graphics", 3, "CS", "MATH 2418"),
("MATH 2417", "Calculus 1", 4, "NSM", NULL),
("CS 3162", "Professional Responsibility in Computer Science", 1, "CS", NULL),
("CS 3345", "Data Structures and Algorithms", 3, "CS", NULL),
("GOVT 2107", "Government and Politics", 1, "IS", NULL);
```

(Inserting Section)

```
INSERT INTO SECTION
VALUES
("AHST 1101", 1, "001", 13, 40, 40),
("CS 4361", 55,  "055", 16, 50, 2),
("MIS 6308", 13, "013", 15, 30, 0),
("NSC 3344", 25, "025", 14, 25, 10);
```

(Inserting Prerequisites)

```
    INSERT INTO PREREQUISITES
    VALUES
    ("CS 4392", "D-", "CS 4361"),
    ("MATH 2418", "D-", "MATH 2417"),
    ("CS 3162", "C+", "CS 3345"),
    ("GOVT 2306", "D-", "GOVT 2107");
```

(Inserting Academic_Department)

```
INSERT INTO Academic_Department (D_Name, Dean_ID)
VALUES (‘CS’, 582749),
    (‘BBS’, 695738),
    (‘A&H’, 692749),=
    (‘SOM’, 040586),
    (‘IS’, 599040),
    (‘NSM’, 860030);
```

(Inserting Has_Taken)

```
INSERT INTO HAS_TAKEN (Student_ID, Course_ID, Grade)
    VALUES
    (39, "CS 3345", "A-"),
    (40, "GOVT 2107", "D+"),
    (41, "AHST 1101", "C-"),
    (42, "MIS 6308", "B");
```

(Inserting User_Account)

```
INSERT INTO User_Account  (Username, Password_Hash, User_ID)
VALUES (‘Briar_She02’, ‘d9f8s@HD0!$cdf’, 195628),
       (‘Priya_MCel00’, ‘02hr30H309hxF@H’, 581650),
       (‘Lanzo_VGwe48’, ‘2MFJDS0!nf!$%mf’, 937659),
       (‘David_AOlu’, ‘*j123FH!@hjsSDF’, 703758),
       (‘Jay_RAri’, ‘SD23fn0#!%mds’,  860546),
       (‘Diane_OJur’, ‘62mFS34$^Gk#$’, 472650);
```

Get Snapshots DBMS (Views)

Get Functional Dependencies

Get Snapshots GUI

Assertions Or Constraints

```
/* can do a check when CREATE SECTION */
ALTER TABLE SECTION
ADD CONSTRAINT COURSE_STUDENT_LIMIT
CHECK (Current_students_taking <= Max_students);
```

// not needed

```
/* In case needed, DONT DELETE
SELECT *
            FROM COURSE C1, COURSE C2
            WHERE C1.Corequisite_ID = C2.Course_ID
            AND      C2.Corequisite_ID = C1.Course_ID)

SELECT *
                      FROM PREREQUISITE P1, PREREQUISITE P2
                      WHERE P1.Prerequisite_ID = P2.Course_ID
                      AND       P2.Prerequisite_ID = P1.Course_ID
*/
```

```
DELIMITER |

CREATE TRIGGER PREVENT_COREQ_CYCLE_INSERT
BEFORE INSERT
ON COURSE
FOR EACH ROW

BEGIN

IF EXISTS ( SELECT *
            FROM COURSE
            WHERE Corequisite_ID = NEW.Course_ID
            AND    NEW.Corequisite_ID = Course_ID)
THEN
      SET NEW.Course_ID = NULL;
END IF;

END |

DELIMITER ;
```

```
DELIMITER |

CREATE TRIGGER PREVENT_COREQ_CYCLE_UPDATE
BEFORE UPDATE
ON COURSE
FOR EACH ROW
BEGIN
IF EXISTS ( SELECT *
            FROM COURSE
            WHERE Corequisite_ID = NEW.Course_ID
```

```
               AND    NEW.Corequisite_ID = Course_ID)
THEN
        SET NEW.Course_ID = NULL;
END IF;
END |

DELIMITER ;


DELIMITER |

CREATE TRIGGER PREVENT_PREREQ_CYCLE_INSERT
BEFORE INSERT
ON PREREQUISITES
FOR EACH ROW
BEGIN
IF EXISTS ( SELECT *
               FROM PREREQUISITE
               WHERE Prerequisite_ID = NEW.Course_ID
               AND    NEW.Prerequisite_ID = Course_ID)
THEN
        SET NEW.Course_ID = NULL;
        SET NEW.Prerequisite_ID = NULL;
END IF;
END |

DELIMITER ;




DELIMITER |

CREATE TRIGGER PREVENT_PREREQ_CYCLE_UPDATE
BEFORE UPDATE
ON PREREQUISITES
FOR EACH ROW
BEGIN
IF EXISTS ( SELECT *
               FROM PREREQUISITES
               WHERE Prerequisite_ID = NEW.Course_ID
               AND    NEW.Prerequisite_ID = Course_ID)
```

```
THEN
        SET NEW.Course_ID = NULL;
        SET NEW.Prerequisite_ID = NULL;
END IF;
END |

DELIMITER ;
```

```
DELIMITER |

CREATE TRIGGER TAKEN_PREREQS_INSERT
BEFORE INSERT
ON ENROLLED_COURSES
FOR EACH ROW
BEGIN
IF
    NOT EXISTS ( SELECT *
                FROM SECTION S, PREREQUISITES P, HAS_TAKEN HT
                WHERE S.Section_ID = NEW.Section_ID
                AND P.Course_ID = S.Course_ID
                AND HT.Course_ID = P.Prerequisite_ID
                AND HT.Grade >= P.Required_Grade
                 AND HT.Student_ID = NEW.Student_ID )

   AND

  EXISTS ( SELECT *
                FROM SECTION S, PREREQUISITES P
                WHERE S.Section_ID = NEW.Section_ID
                AND P.Course_ID = S.Course_ID
                  )
    )
THEN
  SET NEW.Section_ID = NULL;
  SET NEW.Student_ID = NULL;

END IF ;

END |

DELIMITER ;
```

```
DELIMITER |

CREATE TRIGGER TAKEN_PREREQS_UPDATE
BEFORE UPDATE
```

```
ON ENROLLED_COURSES
FOR EACH ROW
BEGIN
IF
    NOT EXISTS ( SELECT *
                FROM SECTION S, PREREQUISITES P, HAS_TAKEN HT
                WHERE S.Section_ID = NEW.Section_ID
                AND P.Course_ID = S.Course_ID
                AND HT.Course_ID = P.Prerequisite_ID
                AND HT.Grade >= P.Required_Grade
                 AND HT.Student_ID = NEW.Student_ID
                )

   AND


EXISTS ( SELECT *
                FROM SECTION S, PREREQUISITES P
                WHERE S.Section_ID = NEW.Section_ID
                AND P.Course_ID = S.Course_ID )


THEN
  SET NEW.Section_ID = NULL;
  SET NEW.Student_ID = NULL;

END IF;

END |

DELIMITER ;
DELIMITER |

CREATE TRIGGER TAKEN_COREQS_UPDATE
BEFORE UPDATE
ON ENROLLED_COURSES
FOR EACH ROW
BEGIN
IF
   NOT EXISTS ( SELECT *
            FROM SECTION S, COURSE C
            WHERE S.Section_ID = NEW.Section_ID
            AND C.Course_ID = S.Course_ID
            AND (C.Corequisite_ID, NEW.Student_ID)  IN
                   (SELECT S.Course_ID, EC.Student_ID
                   FROM SECTION S, ENROLLED_COURSES EC
                   WHERE S.Section_ID = EC.Section_ID)
                   )

  AND


EXISTS ( SELECT *
            FROM SECTION S, COURSE C
            WHERE S.Section_ID = NEW.Section_ID
            AND C.Course_ID = S.Course_ID
            AND C.Corequisite_ID IS NOT NULL )


THEN
  SET NEW.Section_ID = NULL;
  SET NEW.Student_ID = NULL;

END IF;

END |

DELIMITER ;
DELIMITER |

CREATE TRIGGER TAKEN_COREQS_INSERT
BEFORE INSERT
ON ENROLLED_COURSES
FOR EACH ROW
BEGIN
IF
   NOT EXISTS ( SELECT *
             FROM SECTION S, COURSE C
             WHERE S.Section_ID = NEW.Section_ID
             AND C.Course_ID = S.Course_ID
             AND (C.Corequisite_ID, NEW.Student_ID)  IN
                     (SELECT S.Course_ID, EC.Student_ID
                     FROM SECTION S, ENROLLED_COURSES EC
                     WHERE S.Section_ID = EC.Section_ID)
                     )

  AND


EXISTS ( SELECT *
             FROM SECTION S, COURSE C
             WHERE S.Section_ID = NEW.Section_ID
             AND C.Course_ID = S.Course_ID
             AND C.Corequisite_ID IS NOT NULL )


THEN
  SET NEW.Section_ID = NULL;
  SET NEW.Student_ID = NULL;

END IF;

END |

DELIMITER ;
```

Make sure current students taking a section_id(section + course) <= max students
Each course must have at least 1 sections

Counselor can override prerequisites and corequisites if they communicate it with the student beforehand

Each lab section has an associated lecture section

When students are registering for a course, they must have all of the prerequisites completed with a sufficient grade.

If adding course B with coreq A, make sure that coreq A does not have B listed as a coreq.

Triggers
Increment students taking a course

```
CREATE TRIGGER STUDENT_ENROLLED
AFTER INSERT
ON ENROLLED_COURSES
FOR EACH ROW
UPDATE SECTION
SET Current_students_taking = Current_students_taking + 1
WHERE Section_ID = NEW.Section_ID;
```

```
delimiter |
CREATE TRIGGER STUDENT_SECTION_UPDATE
AFTER UPDATE
ON ENROLLED_COURSES
FOR EACH ROW
BEGIN
UPDATE SECTION
SET Current_students_taking = Current_students_taking - 1
WHERE Section_ID = OLD.Section_ID;
UPDATE SECTION
SET Current_students_taking = Current_students_taking + 1
WHERE Section_ID = NEW.Section_ID;
END;
```

```
| delimiter ;



CREATE TRIGGER STUDENT_REMOVED
AFTER DELETE
ON ENROLLED_COURSES
FOR EACH ROW
UPDATE SECTION
SET Current_students_taking = Current_students_taking -1
WHERE Section_ID = OLD.Section_ID;
```

Views

```
DELIMITER //
CREATE PROCEDURE MakeStudentView (IN StudentID CHAR(9) )
BEGIN
CREATE VIEW CONCAT(StudentID,’View’)
AS
SELECT  User_ID, FName, MName, LName, DOB, Dept_Name, Counselor_ID FROM
USER, STUDENT
WHERE Student_ID = StudentID AND User_ID = StudentID;
END;
 //
DELIMITER ;

//PREVENT DUPLICATE COURSES FOR SAME STUDENT
DELIMITER |

CREATE TRIGGER NO_DUP_COURSES_INS
BEFORE INSERT
ON ENROLLED_COURSES
FOR EACH ROW
BEGIN
IF
        EXISTS (SELECT * FROM ENROLLED_COURSES EC, SECTION S
                        WHERE S.Course_ID IN (SELECT Course_ID FROM SECTION
                                                                WHERE Section_ID =
NEW.Section_ID)
                        AND EC.Student_ID = NEW.Student_ID
        AND EC.Section_ID = S.Section_ID)
THEN
        SET NEW.Section_ID = NULL;
  SET NEW.Student_ID = NULL;
```

```
END IF;
END |
DELIMITER ;



DELIMITER |

CREATE TRIGGER NO_DUP_COURSES_UPD
BEFORE UPDATE
ON ENROLLED_COURSES
FOR EACH ROW
BEGIN
IF
        EXISTS (SELECT * FROM ENROLLED_COURSES EC, SECTION S
                        WHERE S.Course_ID IN (SELECT Course_ID FROM SECTION
                                                                WHERE Section_ID =
NEW.Section_ID)
                        AND EC.Student_ID = NEW.Student_ID
        AND EC.Section_ID = S.Section_ID)
THEN
        SET NEW.Section_ID = NULL;
   SET NEW.Student_ID = NULL;

END IF;
END |
DELIMITER ;
```

VIEW COMMANDS:

```
CREATE VIEW Students
AS
SELECT  User_ID, FName, MName, LName, DOB, Dept_Name, Counselor_ID FROM
USER,STUDENT
WHERE User_ID IN (SELECT Student_ID FROM STUDENT) AND Student_ID=User_ID;

CREATE VIEW Deans
AS
SELECT  User_ID, FName, MName, LName, DOB, Dept_Name FROM USER
WHERE User_ID IN (SELECT Dean_ID FROM DEAN);

CREATE VIEW Professors
AS
```

```
SELECT  User_ID, FName, MName, LName, DOB, Dept_Name FROM USER
WHERE User_ID IN (SELECT Professor_ID FROM PROFESSOR);
```

```
CREATE VIEW Counselors
AS
SELECT  User_ID, FName, MName, LName, DOB, Dept_Name FROM USER
WHERE User_ID IN (SELECT Counselor_ID FROM COUNSELOR)
```

```
CREATE VIEW         getstudentcourses
AS SELECT           DISTINCT Enrolled_Courses.Section_ID
FROM                Enrolled_Courses, Student
WHERE               Student.Student_ID = Enrolled_Courses.Student_ID
GROUP BY            Enrolled_Courses.Section_ID;
```

```
PsudeoCode in BackEnd Application:
Get studentID from login credentials
Make SQL Query- EXEC MakeStudentView @StudentID = studentID;
```

```
CREATE PROCEDURE GetStudentCourseHistory @StudentID CHAR(9)
AS
BEGIN
CREATE VIEW @StudentID+’CourseHistory’
AS
SELECT CourseID,Grade
FROM Has_Taken
WHERE Student_ID= @StudentID
END;
```

```
PsudeoCode in BackEnd Application:
Get studentID from login credentials
Make SQL Query- EXEC GetStudentCourseHistory @StudentID = studentID;
```

```
CREATE PROCEDURE MakeProfessorView @ProfessorID CHAR(9)
AS
BEGIN
CREATE VIEW @ProfessorID+’View’
AS
SELECT  * FROM Professor
WHERE Professor_ID= @ProfessorID
```

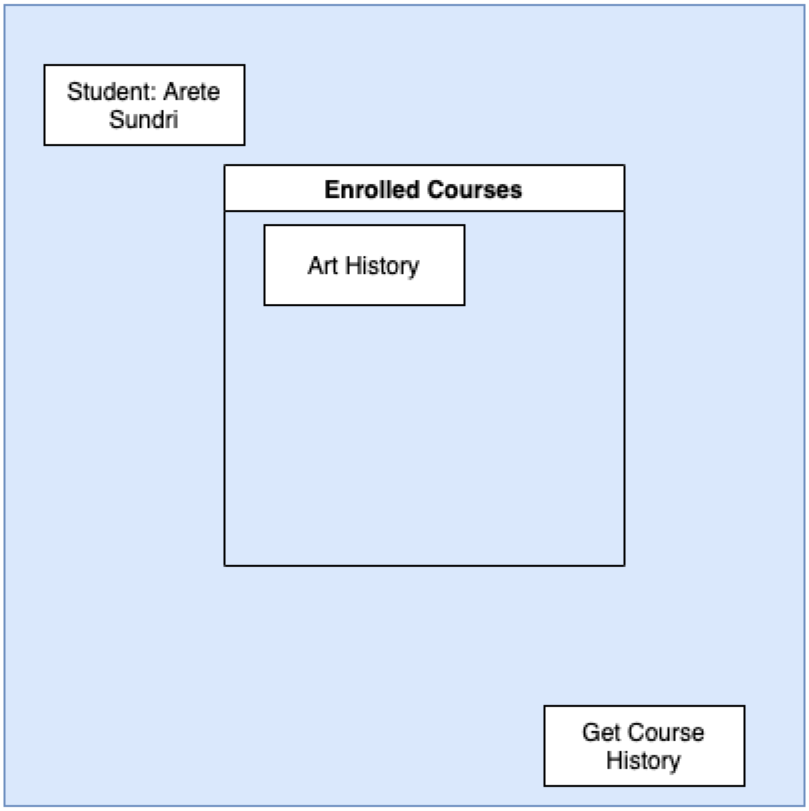
```
END;

CREATE PROCEDURE GetProfessorCourseHistory @ProfessorID CHAR(9)
AS
BEGIN
CREATE VIEW @ProfessorID+’CourseHistory’
AS
SELECT *
FROM Section
WHERE Professor_ID= @ProfessorID
END;

CREATE PROCEDURE MakeCounselorView @CounselorID CHAR(9)
AS
BEGIN
CREATE VIEW @CounselorID+’View’
AS
SELECT  * FROM Counselor
WHERE Counselor_ID= @CounselorID
END;

CREATE PROCEDURE GetCounselorStudents @CounselorID CHAR(9)
AS
BEGIN
CREATE VIEW @CounselorID+’Students’
AS
SELECT  Student_ID FROM Students
WHERE Counselor_ID= @CounselorID
END;

CREATE PROCEDURE MakeDeanView @DeanID CHAR(9)
AS
BEGIN
CREATE VIEW @DeanID+’View’
AS
SELECT * from Dean
WHERE Dean_ID=@DeanID;

CREATE PROCEDURE GetDeanSuboordinates @DeanID CHAR(9)
AS
BEGIN
CREATE VIEW @DeanID+’Suboordinates’
AS
SELECT u.User_ID FROM User u, Dean d, Department a
```

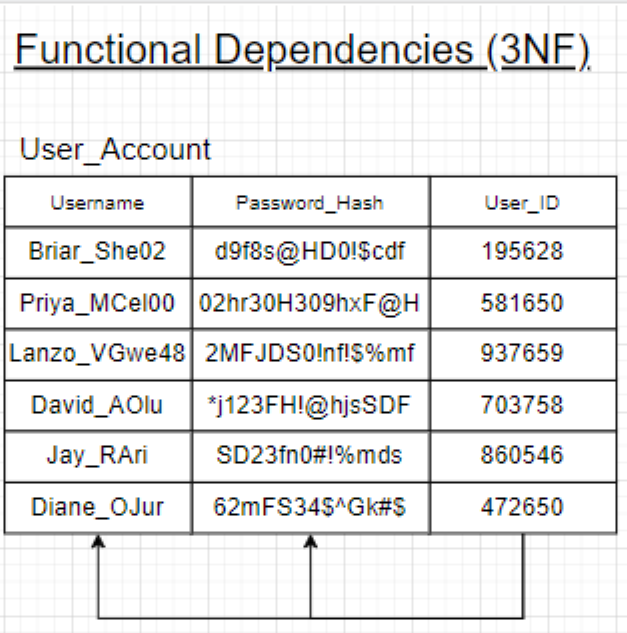
```
WHERE d.DeanID=@DeanID AND d.Dept_name=a.Dept_name AND
u.Dept_name=a.Dept_name;
```

TO DO

Enter sample data entries
Snapshots of tables in DBMS
SQL statements for construction and population (construction done)
Identify functional dependencies (done)
Implementation and demonstration of database system (snapshots of GUI)
indexing(optional)
Additional queries and views (snapshots of query and view implementations)

GUI Design Demo:

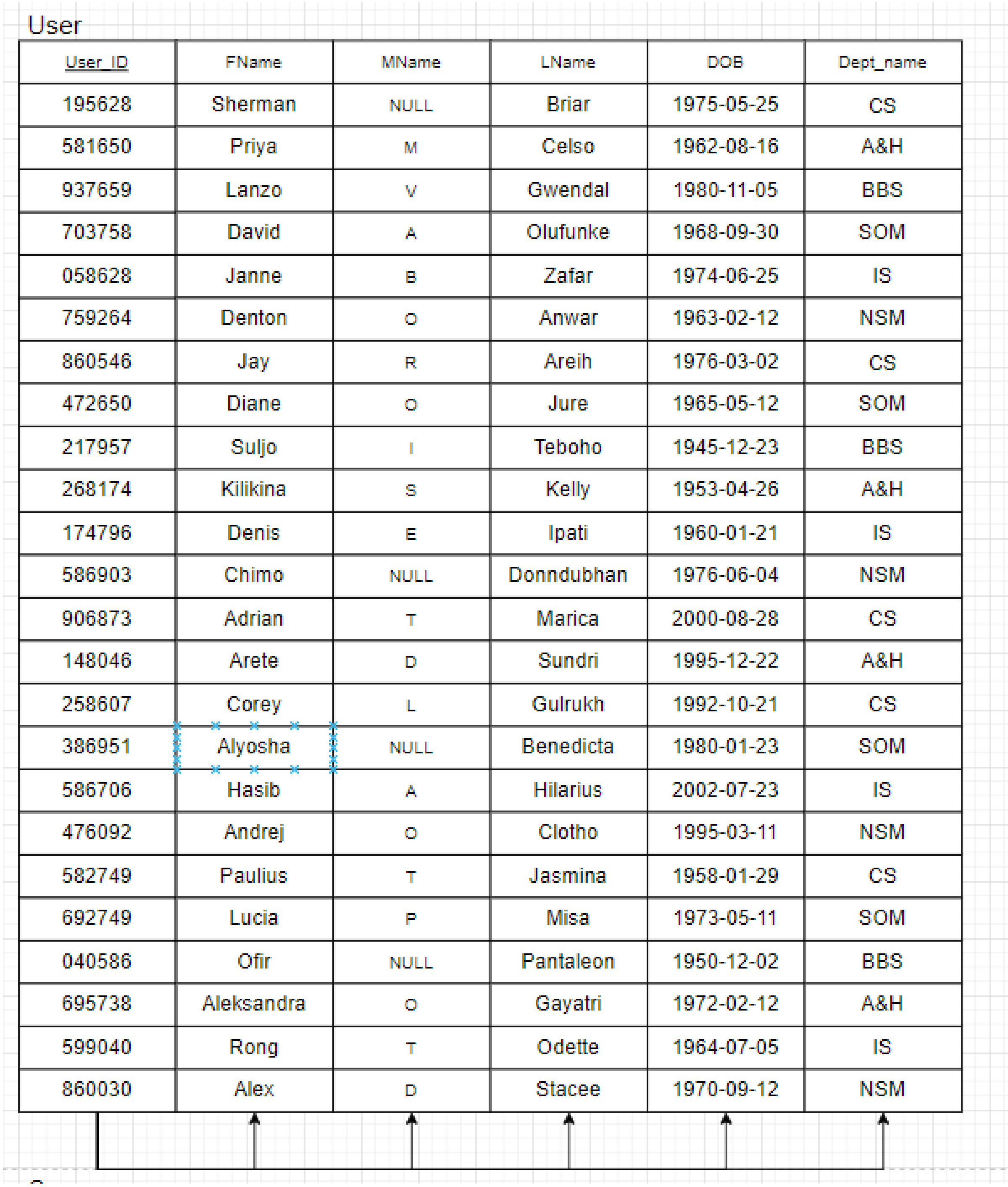




Functional Dependencies (3NF)

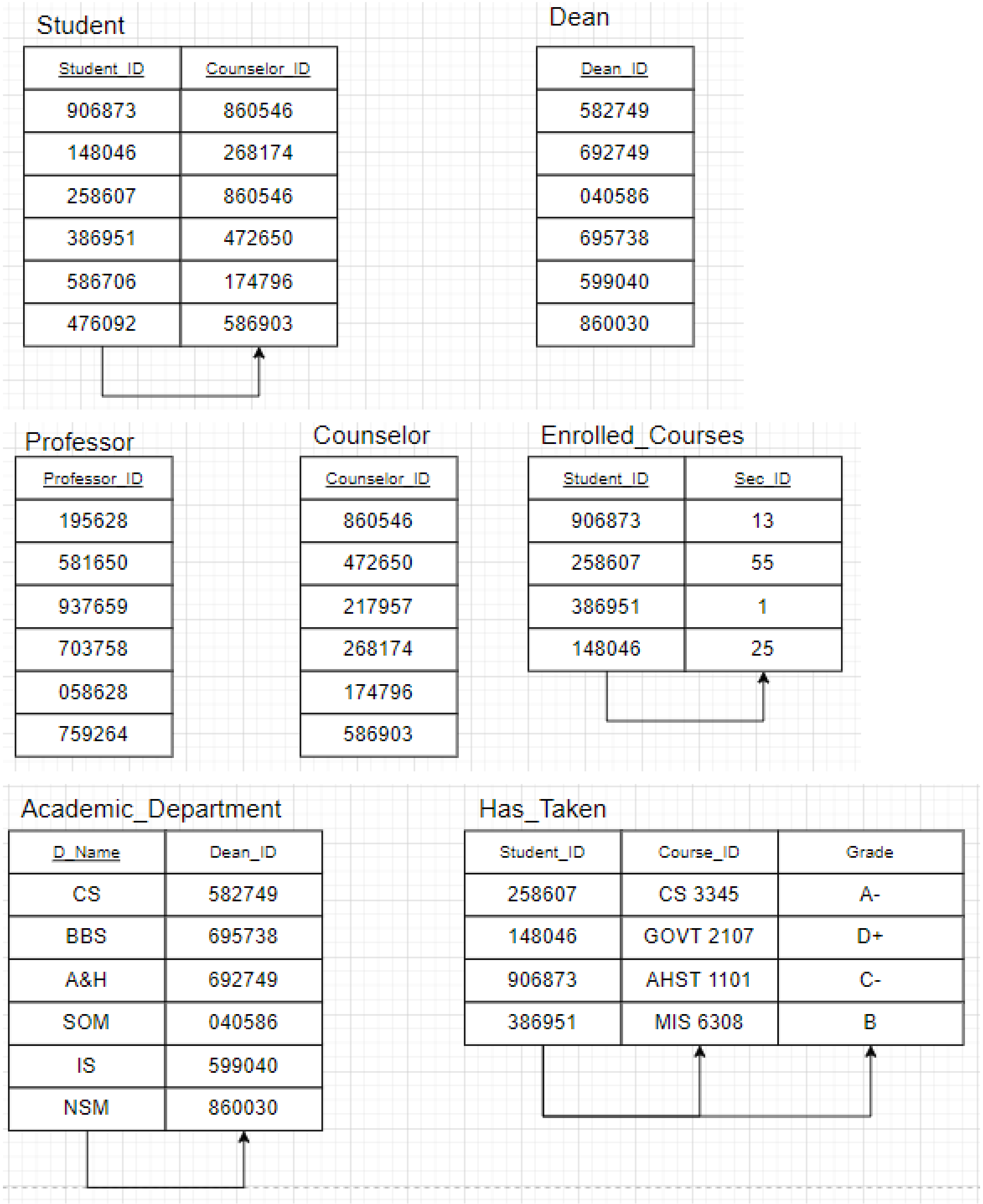
User_Account

Username	Password_Hash	User_ID
Briar_She02	d9f8s@HD0!$cdf	195628
Priya_MCel00	02hr30H309hxF@H	581650
Lanzo_VGwe48	2MFJDS0!nf!$%mf	937659
David_AOlu	*j123FH!@hjsSDF	703758
Jay_RAri	SD23fn0#!%mds	860546
Diane_OJur	62mFS34$^Gk#$	472650



User

User_ID	FName	MName	LName	DOB	Dept_name
195628	Sherman	NULL	Briar	1975-05-25	CS
581650	Priya	M	Celso	1962-08-16	A&H
937659	Lanzo	V	Gwendal	1980-11-05	BBS
703758	David	A	Olufunke	1968-09-30	SOM
058628	Janne	B	Zafar	1974-06-25	IS
759264	Denton	O	Anwar	1963-02-12	NSM
860546	Jay	R	Areih	1976-03-02	CS
472650	Diane	O	Jure	1965-05-12	SOM
217957	Suljo	I	Teboho	1945-12-23	BBS
268174	Kilikina	S	Kelly	1953-04-26	A&H
174796	Denis	E	Ipati	1960-01-21	IS
586903	Chimo	NULL	Donndubhan	1976-06-04	NSM
906873	Adrian	T	Marica	2000-08-28	CS
148046	Arete	D	Sundri	1995-12-22	A&H
258607	Corey	L	Gulrukh	1992-10-21	CS
386951	Alyosha	NULL	Benedicta	1980-01-23	SOM
586706	Hasib	A	Hilarius	2002-07-23	IS
476092	Andrej	O	Clotho	1995-03-11	NSM
582749	Paulius	T	Jasmina	1958-01-29	CS
692749	Lucia	P	Misa	1973-05-11	SOM
040586	Ofir	NULL	Pantaleon	1950-12-02	BBS
695738	Aleksandra	O	Gayatri	1972-02-12	A&H
599040	Rong	T	Odette	1964-07-05	IS
860030	Alex	D	Stacee	1970-09-12	NSM



Student

Student_ID	Counselor_ID
906873	860546
148046	268174
258607	860546
386951	472650
586706	174796
476092	586903

Dean

Dean_ID
582749
692749
040586
695738
599040
860030

Professor

Professor_ID
195628
581650
937659
703758
058628
759264

Counselor

Counselor_ID
860546
472650
217957
268174
174796
586903

Enrolled_Courses

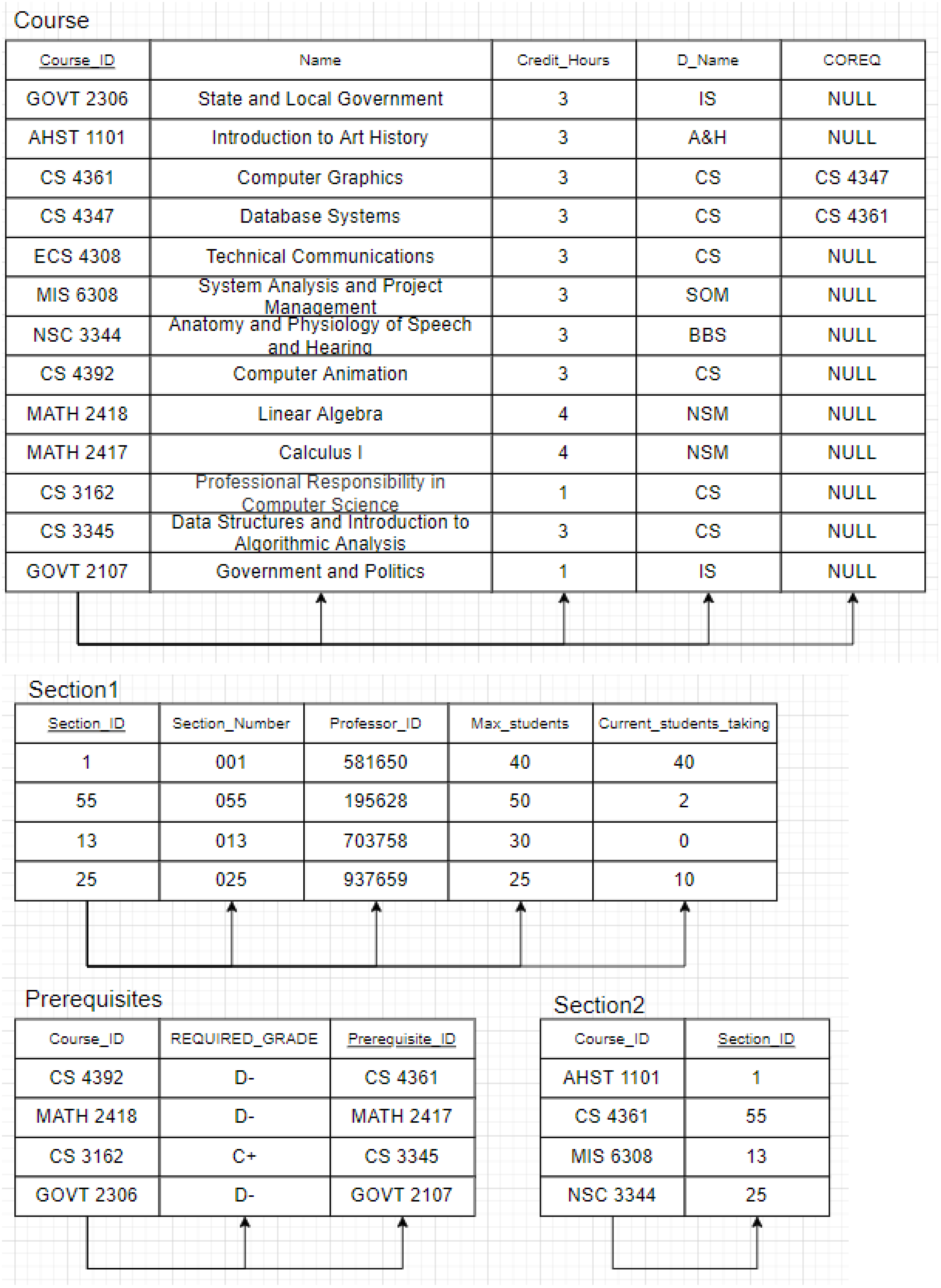
Student_ID	Sec_ID
906873	13
258607	55
386951	1
148046	25

Academic_Department

D_Name	Dean_ID
CS	582749
BBS	695738
A&H	692749
SOM	040586
IS	599040
NSM	860030

Has_Taken

Student_ID	Course_ID	Grade
258607	CS 3345	A-
148046	GOVT 2107	D+
906873	AHST 1101	C-
386951	MIS 6308	B



Course

Course_ID	Name	Credit_Hours	D_Name	COREQ
GOVT 2306	State and Local Government	3	IS	NULL
AHST 1101	Introduction to Art History	3	A&H	NULL
CS 4361	Computer Graphics	3	CS	CS 4347
CS 4347	Database Systems	3	CS	CS 4361
ECS 4308	Technical Communications	3	CS	NULL
MIS 6308	System Analysis and Project Management	3	SOM	NULL
NSC 3344	Anatomy and Physiology of Speech and Hearing	3	BBS	NULL
CS 4392	Computer Animation	3	CS	NULL
MATH 2418	Linear Algebra	4	NSM	NULL
MATH 2417	Calculus I	4	NSM	NULL
CS 3162	Professional Responsibility in Computer Science	1	CS	NULL
CS 3345	Data Structures and Introduction to Algorithmic Analysis	3	CS	NULL
GOVT 2107	Government and Politics	1	IS	NULL

Section1

Section_ID	Section_Number	Professor_ID	Max_students	Current_students_taking
1	001	581650	40	40
55	055	195628	50	2
13	013	703758	30	0
25	025	937659	25	10

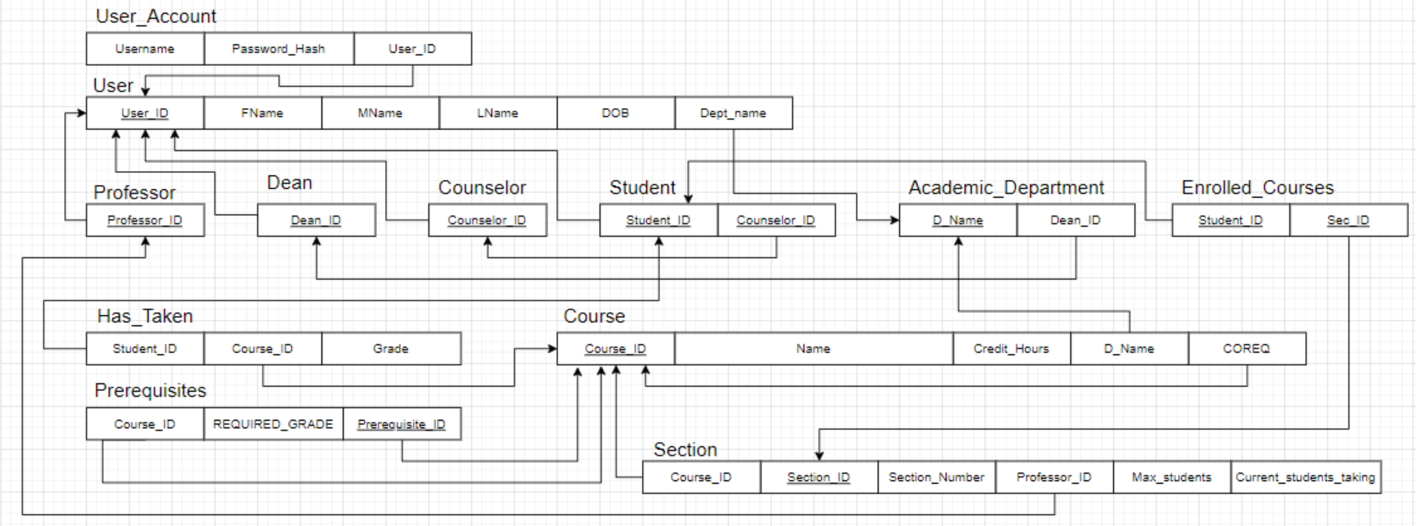
Prerequisites

Course_ID	REQUIRED_GRADE	Prerequisite_ID
CS 4392	D-	CS 4361
MATH 2418	D-	MATH 2417
CS 3162	C+	CS 3345
GOVT 2306	D-	GOVT 2107

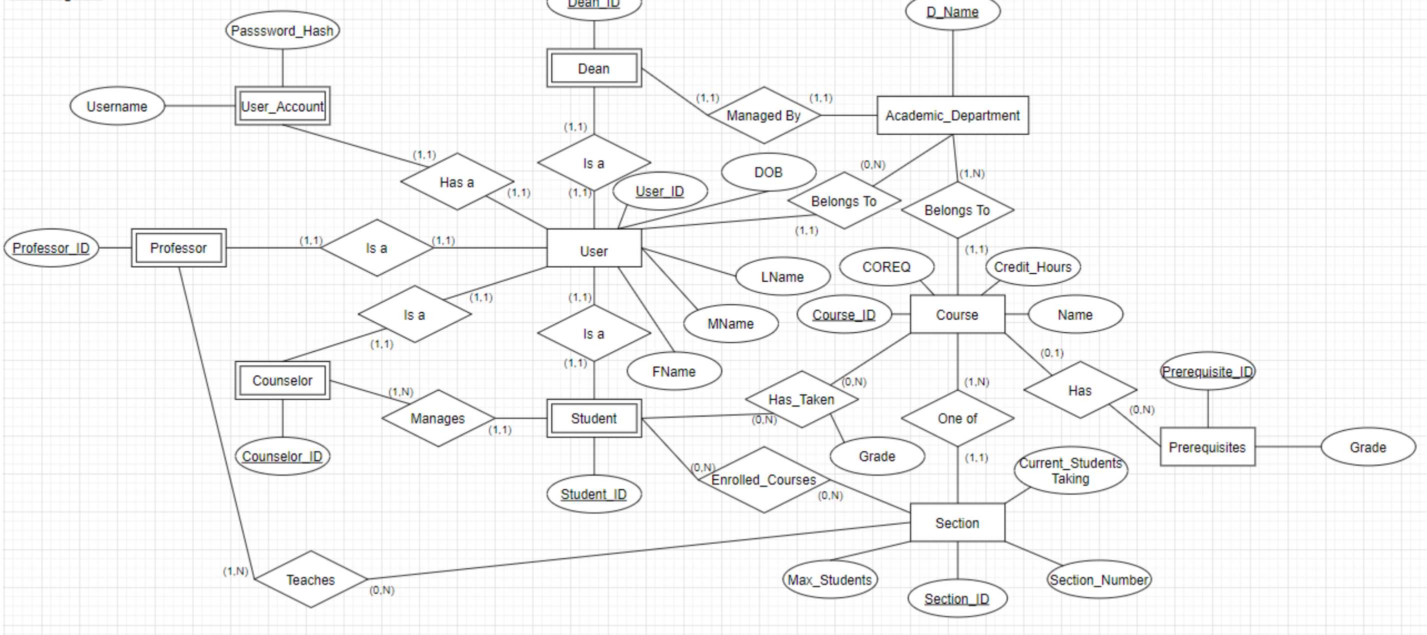
Section2

Course_ID	Section_ID
AHST 1101	1
CS 4361	55
MIS 6308	13
NSC 3344	25

Relations Schema



ER Diagram


Dean_ID
D_Name
Passsword_Hash
Dean
Username
User_Account
(1,1)
Managed By
(1,1)
Academic_Department
(1,1)
Is a
(1,1)
Has a
(0,N)
(1,N)
DOB
(1,1)
(1,1)
User_ID
Belongs To
Belongs To
Professor_ID
Professor
(1,1)
Is a
(1,1)
User
(1,1)
(1,1)
LName
COREQ
Credit_Hours
Is a
(1,1)
(1,1)
MName
Course_ID
Course
Name
Is a
(1,1)
(1,1)
FName
(0,1)
Counselor
(1,N)
Has_Taken
(0,N)
Has
Prerequisite_ID
Manages
(1,1)
Student
(0,N)
(1,N)
One of
(0,N)
Prerequisites
Grade
Counselor_ID
Grade
(1,1)
Current_Students Taking
(0,N)
Enrolled_Courses
Student_ID
(0,N)
Section
(1,N)
Teaches
(0,N)
Max_Students
Section_ID
Section_Number