Question #1

Student(SNUM: integer primary key, SName: string, Major: string, level: string, age: integer)

Class(Name: string primary key, meets at: time, room: string, fid: integer)

Enrolled(SNUM: integer, CName: string, primary key(SNUM, CName))

Professor(PID: integer primary key, FName: string, DeptID: integer)

1.1) Write the SQL statements required to create these relations, including appropriate versions of all primary and foreign key integrity constraints

CREATE TABLE Student (SNUM integer,

SName varchar(255),

major varchar(255),

level varchar(255),

age integer,

PRIMARY KEY (SNUM)) ;

CREATE TABLE Professor (PID integer,

FName varchar(255),

DeptID integer,

PRIMARY KEY (PID));

Here we need to reference the foreign keys to their respective tables which will be done at the end

CREATE TABLE Class ( Name varchar(255),

meets at TIME(0),

room varchar(255),

fid integer,

PRIMARY KEY (Name),

FOREIGN KEY (fid) REFERENCES Professor(PID));

In the enrolled table we need to reference the foreign keys to their respective tables

CREATE TABLE Enrolled (SNUM integer,

CName varchar(255),

PRIMARY KEY (SNUM, CName),

FOREIGN KEY (SNUM) REFERENCES Student),

FOREIGN KEY (CName) REFERENCES Class(Name));

1.2)

a) Every class has a minimum enrollment of 15 students and a maximum enrollment of 30 students. In this case we will need to use a check constraint in the Enrolled table to make sure the conditions are satisfied

CREATE TABLE Enrolled (SNUM integer,

CName varchar(255),

PRIMARY KEY (SNUM, CName),

FOREIGN KEY (SNUM) REFERENCES Student),

FOREIGN KEY (CName) REFERENCES Class)

CHECK (( SELECT COUNT (SNUM)

FROM Enrolled

GROUP BY CName) >= 15),

CHECK (( SELECT COUNT (SNUM)

FROM Enrolled

GROUP BY CName) <= 30));

b) At least one class meets in each room.

We do not require to add any additional constraints due to the fact that this condition is already satisfied because each room is associated with classes through the use of foreign keys meaning that a room cannot exist without an association to a class.

c) Every professor must teach at least two courses.

In this case, we will use a combination of an Assertion with check constraints to make sure the condition is satisfied. We will also use the COUNT command to aid us in determining if each professor teaches at least 2 classes.

CREATE ASSERTION instance

CHECK ((SELECT COUNT (\*)

FROM Professor P, Class C WHERE P.PID = C.fid

GROUP BY C.fid

HAVING COUNT (\*) < 2) = 0)

d) Two classes cannot meet in the same room at the same time.

In this case, we will need to modify the Class table to include a CHECK constraint that will make sure condition is satisfied.

CREATE TABLE Class ( Name varchar(255),

meets at TIME(0),

room varchar(255),

fid integer,

PRIMARY KEY (Name),

FOREIGN KEY (fid) REFERENCES Professor(PID));

CHECK ((SELECT COUNT (\*)

FROM ( SELECT room, meets at

FROM Class

GROUP BY room, meets at

HAVING COUNT (\*) > 1)) = 0));

This Check constraint will look for any two classes that meets in the same room and the same time and if it returns something then we know that condition is not satisfied

e) Professors from different departments cannot teach in the same room.

This one is a bit similar to the previous question however it deals with Professors. Therefore, we need to implement a Check constraint that will make sure that the condition is satisfied

CREATE ASSERTION instance

CHECK ((SELECT COUNT (\*)

FROM Professor P, Professor P2, Class C1, Class C2

WHERE P.PID = C1.fid

AND P2.PID = C2.fid AND C1.room = C2.room AND P.DeptID = P2.DeptID) = 0);