Part 1

Task 1.1
Relation A:
1. {EmpID}, {SSN}, {Phone}, {EmpID, SSN}, {SSN, Phone}{EmpID, SSN, Phone}.
2. EmpID, SSN, Phone, Email.
3. EmpID would bee chosen for being the primary key because it shows the ID of a person. It is the main identifier, so I would choose that.
4. No, because employees have their own phone. In the table the logic of the phone number is increasing number from 555-0101.

Relation B:
1. The minimum attributes for being primary key are unique and not null. There can be only one primary key in the table.
2. Primary key is identifier, that means it must determine the code for each object. Objects shouldn't leave without determining. So there are unique and not null attributes. Unique is for avoiding the identifying two different objects like the one.
3. StudentID.

1.2.
Enrollment(StudentID)->Student(StudentID)

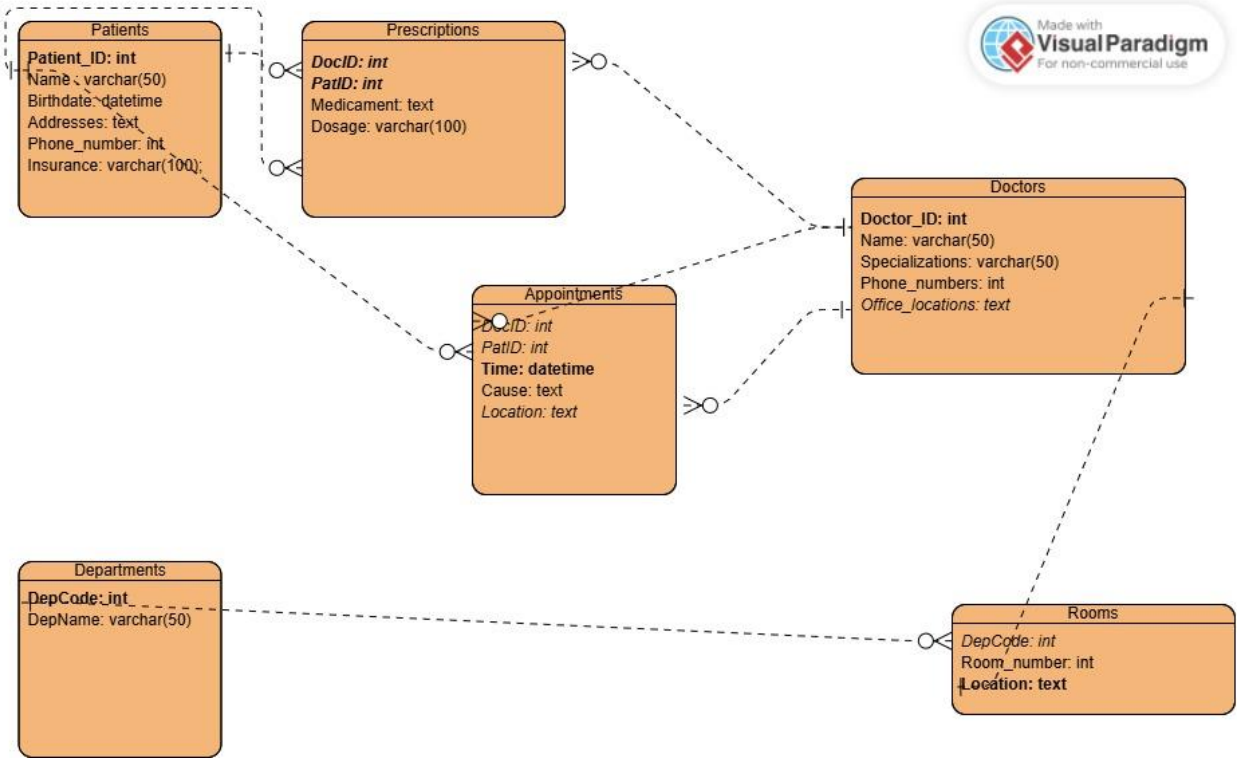
Enrollment(CourseID)->Course(CourseID)

Department(DeptCode)->Course(DepartmentCode)

Department(DeptName)->Professor(Department).

Part 2.

Task 2.1



1.Strong: Patients, Doctors, Departaments, Rooms, Appointments. Weak: Prescriptions.

2. For Patients Patient_ID and Phone Number are simple, Birthdate and Addresses are composite and multivalued, Insurance multi-valued. For Doctors Doctor_ID, Phone_numbers are simple, Office_locations is composite, Name and Specializations are multi-valued and simple. For Prescriptions all attributes are multi-valued, DocID and PatID are simple. For Appointments Time is composite, Cause, DocID and PatID are single and multi-valued, Location is derived (by knowing DocID, you know the location of appointment because each doctor has only one room location). For Rooms DepCode is simple, multi-valued, Room_number is multi-valued, Location is composite like Office_location of Doctors. For Departments DepCode is simple, DepName is simple.

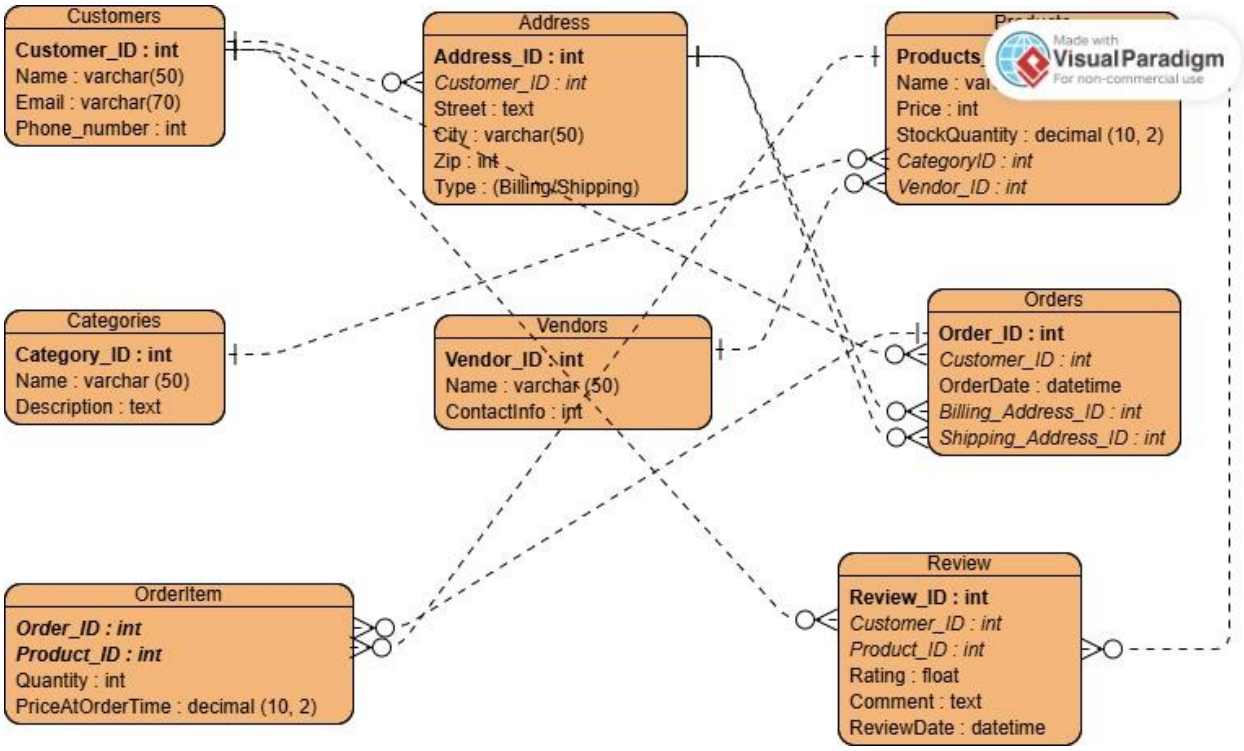
3. Appointments(Pat_ID)-> Patients(Patient_ID)[M:1], Appointments(Doc_ID)->Doctors(Doctor_ID)[M:1], Appointments(Location)->Doctors(Office_location)[M:1]. Prescriptions->Patients and ->Doctors [M:1]. Doctors(Office_location)->Rooms(Location)[1:1], Rooms(DepCode)->Departments(DepCode).

4. It is drawn.

5. Primary keys are written by bold texts.

Task 2.2

1.



2. OrderItem is weak entity, because it does not have primary independent keys for identifying its examples by only their attributes. The main keys reference to two entities, so this is not strong entity.

3. Orders and Products could have M:N relationship. One order can contain several Products, One product can be in several orders. But there is no necessary attributes Quantity and PriceAtOrderTime, OrderItem gives that.

Part 4.

Task 4.1.

StudentProject(

StudentID, StudentName, StudentMajor,

ProjectID, ProjectTitle, ProjectType,

SupervisorID, SupervisorName, SupervisorDept,

Role, HoursWorked, StartDate, EndDate

)

1. FD (Functional Dependences) are:

StudentID -> StudentName, StudentMajor

ProjectID -> ProjectTitle, ProjectType, SupervisorID

SupervisorID -> SupervisorName, SupervisorDept

StudentID, ProjectID -> Role, HoursWorked, StartDate, EndDate

2. Redundacy:

Student data repeat if he participates in several projects. Project data repeat if several students participate in this project.

For updating Supervisor or Student or Project data, we must do that with several entries in the table. We cannot insert new project without a student and new student without project, because they define Role, HoursWorked, StartDate, EndDate together. When we delete last student, we lose data about project and supervisor.

3. 1NF:
All attributes are simple, atomaric.

4. 2NF:

There are four partial dependences, that violate 2NF. For fixing that, we divide them into several tables. Students(StudentID, StudentName, StudentMajor), Projects(ProjectID, ProjectTitle, ProjectType), Supervisors (SupervisorID, SupervisorName, SupervisorDept). StudentProject (StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate). Now 2NF is done.

5. 3NF:
There is no transitive dependences because each table's attributes depend on only primary key.

Task 4.2.

CourseSchedule(StudentID, StudentMajor, CourseID, CourseName, InstructorID, InstructorName, TimeSlot, Room, Building)

1. Primary keys are StudentID and CourseID.

2. StudentID -> StudentName

CourseID -> CourseName, InstructorID, Timeslot, Room

Room -> Building

InstructorID -> InstructorName, CourseID

TimeSlot, Room -> CourseID

3. The table is not in BCNF, because table contains dependences X -> Y, where X is not superkey. For example, Room -> Building.

4. Students(StudentID, StudentMajor, StudentName)

Courses(CourseID, CourseName, InstructorID, Timeslot, Room)

RoomInfo(Room, Building)

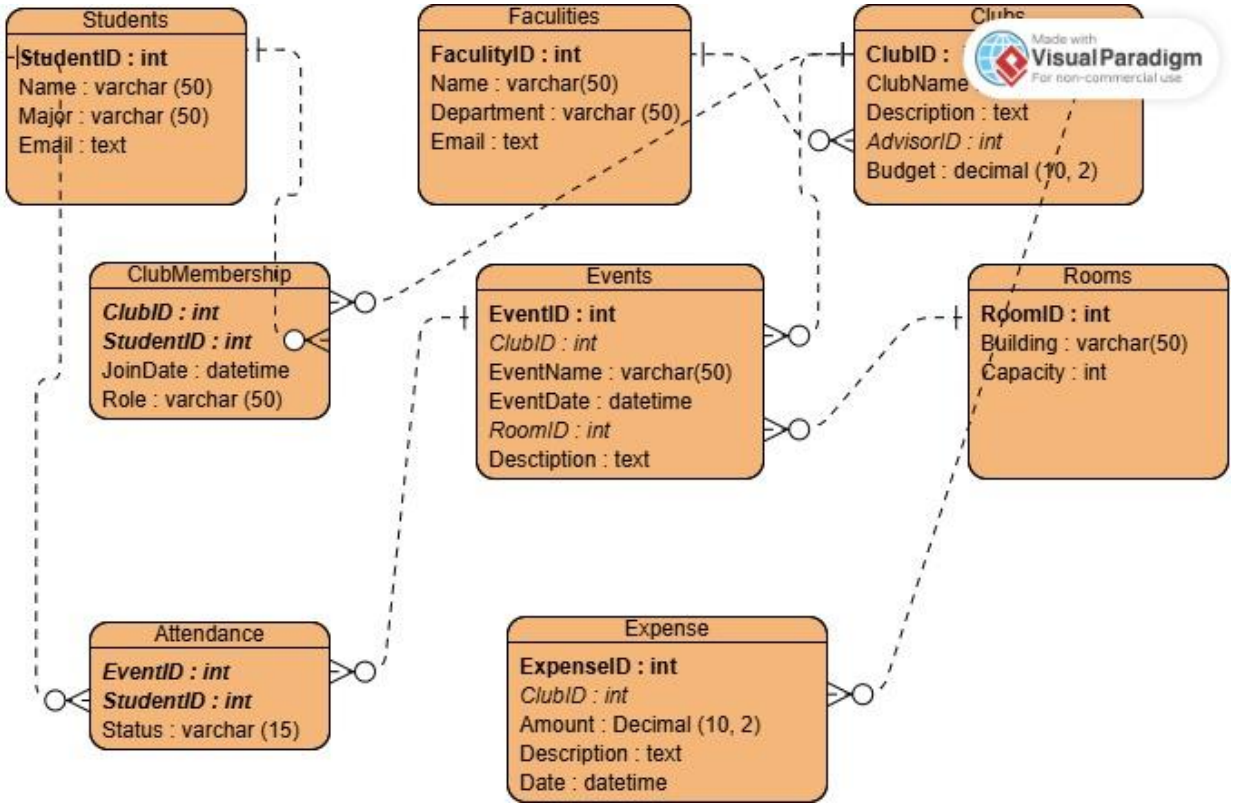
Enrollment(StudentID, CourseID)

5. If we connect these tables, there will be previous table with all the attributes. So we don't have loss.

Part 5

Task 5.1.

1.



3. Expense and Clubs could be single, just Clubs (ClubID, ClubName, Description, AdvisorID, Budget, Date). But for understanding, I decided to divide them, for writing dates for each expense, not for ech clubs. 4. "Find top 3 clubs, where Student with major "Business Analyst" came more often than other clubs", "Who is the Advisor and what is his department, who had more than 10000 $ expense for his club?", "What the role does a student with major "Programmer" or "Tester" take more than other roles?".