1.1

A

1. EmpID; SSN; Email; (EmpID Name); (SSN phone); (Email Department)

2. EmpID; SSN; Email

3. EmpID is numerical (convenient for indexing), not change

4. Yes, because 2 employees can have 1 work phone

B

1. StudentID, CourseCode, Section, Semester, Year

2. StudentID – defines student, CourseCode – defines course, Section – different groups of same course, semester + year – defines time

3.No because if we remove 1 it will not unique for students

1.2

1. Student(AdvisorID) – Professor(ProfID)

Course(DepartmentCode) - Department(DeptCode)

Department(ChairID) - Professor(ProfID)

Enrollment(StudentID) - Student(StudentID)

Enrollment(CourseID) - Course(CourseID)

2.1

1. Patient (strong)

Doctor (strong)

Department (strong)

Appointment (strong, as it's a separate event)

Prescription (weak, depends on Patient + Doctor)

Room (weak, depends on Department)

2. Patient: PatientID, Name, Birthdate, Address (composite: Street, City, State, Zip), Phone (multi-valued), Insurance.

Doctor: DoctorID , Name, Specializations (multi-valued), Phone (multi-valued), Office.

Department: DeptCode, DeptName, Location.

Appointment: ApptID (PK), DateTime, Purpose, Notes

Prescription: PrescriptionID , MedicationName, Dosage, Instructions.

Room: (DeptCode + RoomNumber) (composite).

3.

Patient - (1:N) - Appointment

Doctor - (1:N)- Appointment

Doctor - (1:N)- Prescription.

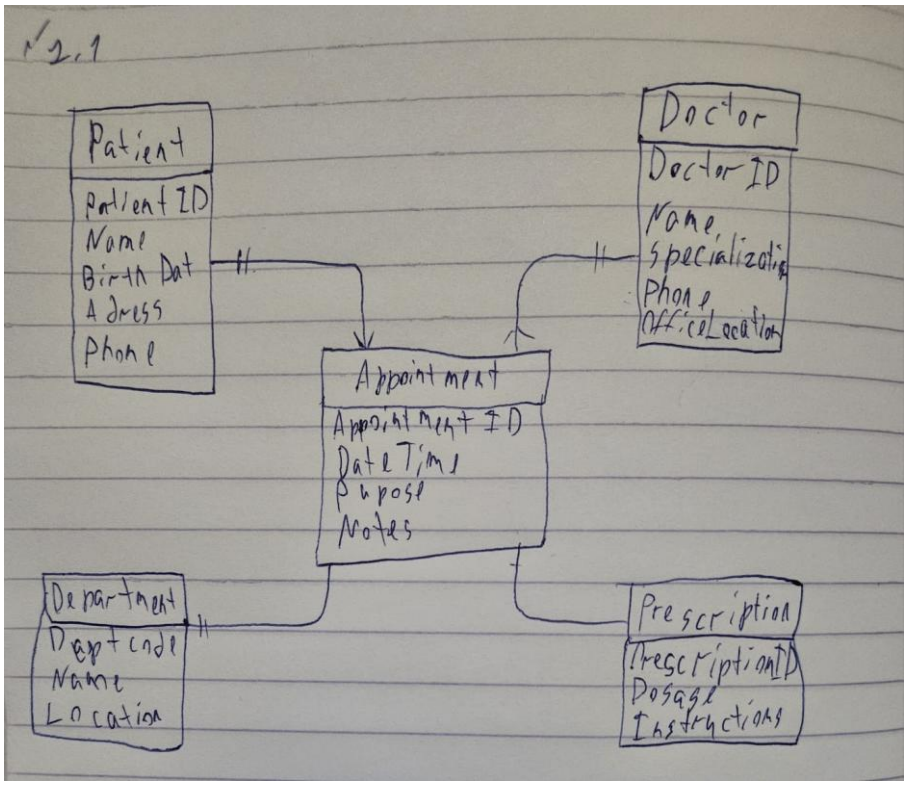
Patient - (1:N)- Prescription.

Department - (1:N)- Doctor.

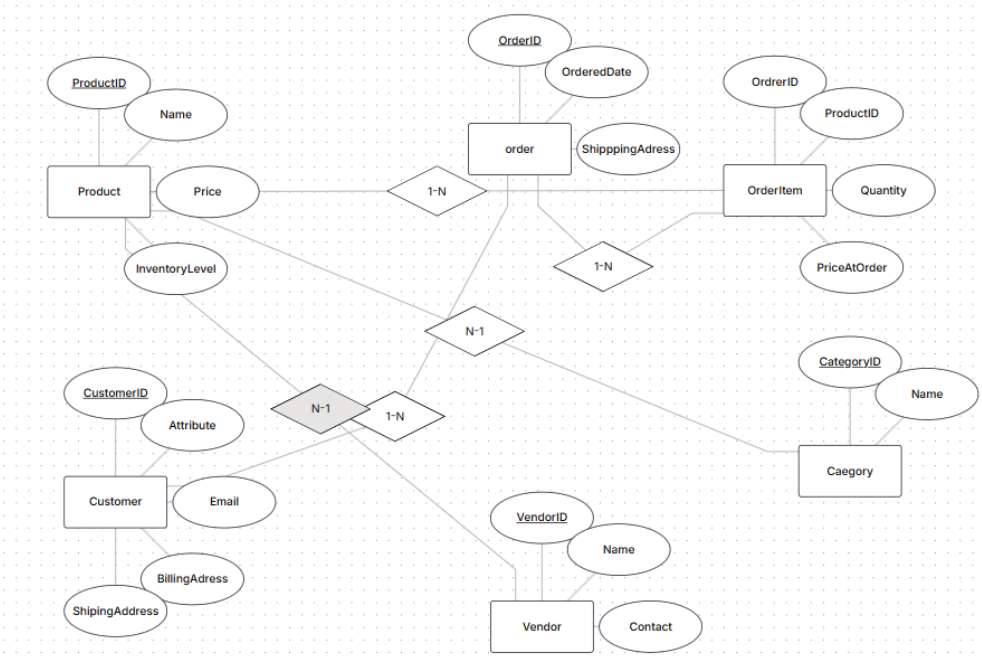
Department - (1:N)- Room.

Prescription – (M:N)-Medication

4.



2.2
1.



2. OrderItem – is weak because it does not have its own PK and exists only as a link between the Order and the Product.
3. Order – Product . One order may contain multiple items. One item may appear in multiple orders.

4.1

1. Functional Dependencies (FDs)

StudentID - StudentName, StudentMajor

ProjectID - ProjectTitle, ProjectType

SupervisorID - SupervisorName, SupervisorDept

(StudentID, ProjectID) - Role, HoursWorked, StartDate, EndDate

2. Problems (Redundancy and Anomalies)

Redundancy:

StudentName and StudentMajor are repeated for every project a student participates in.

ProjectTitle and ProjectType are duplicated for each student assigned to the same project.

SupervisorName and SupervisorDept are repeated for each project supervised by the same person.

Update Anomaly:

If a project's title changes, it must be updated in multiple rows.

Insert Anomaly:

It is impossible to insert a new project without assigning at least one student to it.

Delete Anomaly:

If the last student of a project is removed, the information about the project and its supervisor is lost.

3. 1NF

All attributes are atomic → the table already satisfies 1NF.

4. 2NF

Primary Key: (StudentID, ProjectID).

Partial dependencies exist:

StudentID - StudentName, StudentMajor

ProjectID - ProjectTitle, ProjectType

SupervisorID - SupervisorName, SupervisorDept

Decomposition into 2NF:

1. Student(StudentID, StudentName, StudentMajor)

2. Project(ProjectID, ProjectTitle, ProjectType, SupervisorID)

3. Supervisor(SupervisorID, SupervisorName, SupervisorDept)

4. StudentProject(StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate)

5. 3NF

Transitive dependency: ProjectID → SupervisorID → (SupervisorName, SupervisorDept).

By separating Supervisor into its own table, transitivity is removed.

Final 3NF schema:

1. Student(StudentID, StudentName, StudentMajor)

2. Supervisor(SupervisorID, SupervisorName, SupervisorDept)

3. Project(ProjectID, ProjectTitle, ProjectType, SupervisorID)

4. StudentProject(StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate)

4.2

1. Primary Key

The primary key must uniquely identify an enrollment.

Composite key: (StudentID, CourseID, TimeSlot, Room).

2. Functional Dependencies (FDs)

1. StudentID → StudentMajor

2. CourseID → CourseName

3. InstructorID → InstructorName

4. Room → Building

5. (CourseID, TimeSlot, Room) → InstructorID

3.Not in BCNF because there are dependencies where the determinant is not a superkey:

StudentID → StudentMajor

CourseID → CourseName

InstructorID → InstructorName

Room → Building

4. Decomposition into BCNF

1. Student(StudentID, StudentMajor)

2. Course(CourseID, CourseName)

3. Instructor(InstructorID, InstructorName)

4. Room(Room, Building)

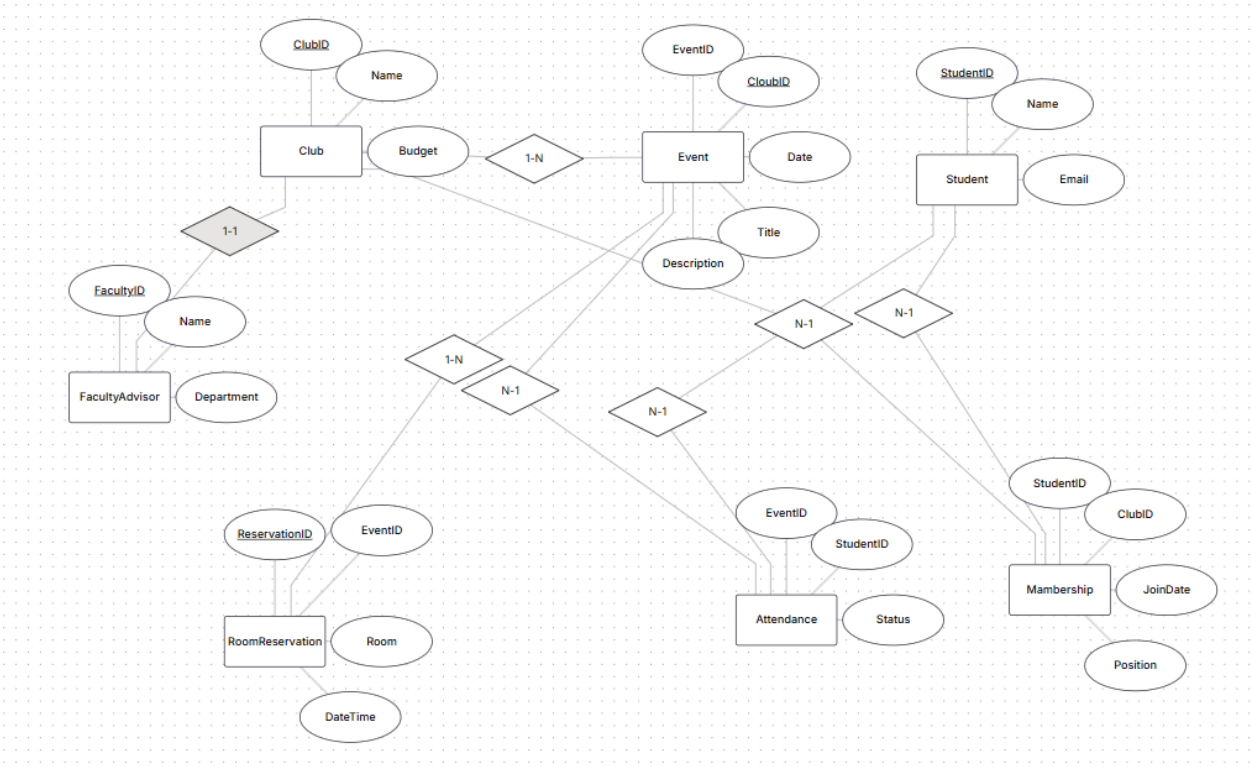
5. CourseSection(CourseID, TimeSlot, Room, InstructorID)

6. Enrollment(StudentID, CourseID, TimeSlot, Room)

5. No information is lost: every original attribute is preserved across the decomposed tables.

5.1

1.



2. Student(StudentID PK, Name, Email)

Club(ClubID PK, Name, Budget, FacultyID FK)

Membership(StudentID PK FK→Student, ClubID PK FK→Club, JoinDate, Position)

Event(EventID PK, ClubID FK→Club, Date, Title, Description)

Attendance(EventID PK FK→Event, StudentID PK FK→Student, Status)

FacultyAdvisor(FacultyID PK, Name, Department)

RoomReservation(ReservationID PK, EventID FK→Event, Room, DateTime)

3. I could model events and room reservations as one table or as two tables. I decided to separate them into Event and RoomReservation because a single event may have multiple room bookings. This design reduces redundancy and makes it easier to track changes.

4. 1 Find all students who are officers in the Computer Science Club

2 List all events scheduled for next week with their room reservations

3 Show the budget and advisor name for each club