**Database Systems Lab — Week 2: Relational Model & Keys**  
**Part 1: Key Identification Exercises  
Task 1.1**

**Relation A: Employee**  
1. Six example superkeys:

* EmpID
* SSN
* Email
* EmpID + Email
* SSN + Phone
* EmpID + SSN + Phone

2. Each of EmpID, SSN, and Email uniquely identify rows in the sample and are minimal. Phone may not be unique in general; Name and Department are not unique.  
  
3. EmpID is a compact, system-assigned integer-like. SSN is sensitive, and Email may change; therefore, EmpID is preferred for PK.  
  
4. Can two employees have the same phone number?

Based on sample data no duplicates are shown; however business rules are required. Phones often are not guaranteed unique. Without a unique constraint, two employees could share a number. If phone uniqueness is required, add UNIQUE constraint on Phone.

**Relation B: Course Registration**

1. Minimum attributes for primary key: (StudentID, CourseCode, Section, Semester, Year)

2. A student can take the same CourseCode in different semesters; the business rule forbids registering the same section in the same semester twice, so Section+Semester+Year distinguishes sections; including StudentID ensures the registration row is per student. Credits and Grade are dependent attributes and are not part of the key.  
  
3. If the institution uses a composite SectionID that already encodes semester & year & course (e.g., SectionInstanceID), then (StudentID, SectionInstanceID) could be a candidate key.

If there is a RegistrationID surrogate, it can also be a candidate key (system-generated).

**Task 1.2**   
Foreign keys:

* Student.AdvisorID -> Professor.ProfID (a student references their advisor)
* Course.DepartmentCode -> Department.DeptCode (course belongs to a department)
* Department.ChairID -> Professor.ProfID (department chair is a professor)
* Enrollment.StudentID -> Student.StudentID (enrollment references student)
* Enrollment.CourseID -> Course.CourseID (enrollment references course)

**Part 2: ER Diagram Construction  
Task 2.1**

1-2.

Patient (strong): PatientID (PK), Name, Birthdate, Address {Street, City, State, Zip} (composite), Phone (multi-valued), InsuranceInfo

Doctor (strong): DoctorID (PK), Name, Specializations (multi-valued), Phone, OfficeLocation

Department (strong): DeptCode (PK), DeptName, Location

Appointment (associative / weak): AppointmentID (PK) or composite (PatientID, DoctorID, DateTime); tracks purpose and notes

Prescription (associative): PrescriptionID (PK) or composite (PatientID, DoctorID, MedID, Date); Medication, Dosage, Instructions

Room (weak, owned by Department): DeptCode (FK, part of PK), RoomNumber (partial key), Type, Capacity → Composite PK: (DeptCode, RoomNumber)

3.

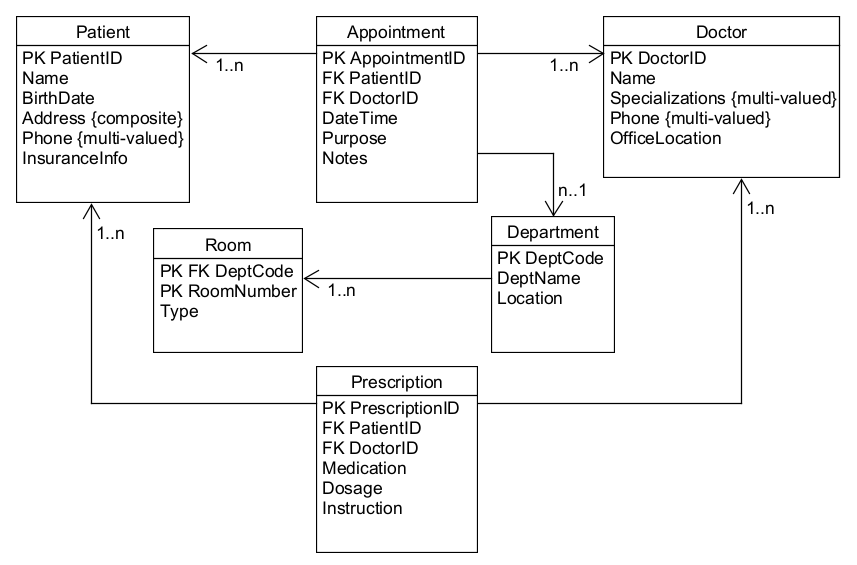
Patient 1..N Appointment N..1 Doctor

Doctor N..1 Department

Department 1..N Room (weak entity)

Patient 1..N Prescription N..1 Doctor

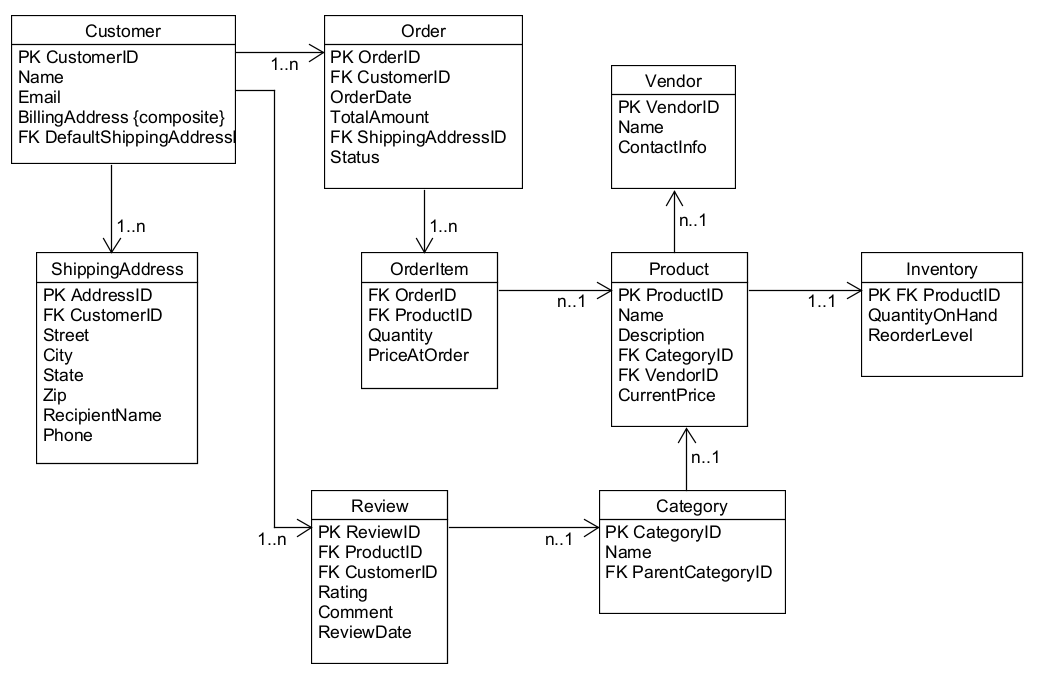
4.

****

5. Patient.PatientID, Doctor.DoctorID, Department.DeptCode, Appointment.AppointmentID, Prescription.PrescriptionID, Room composite key (DeptCode, RoomNumber).

**Task 2.2 — E-commerce Platform**

1.



2. OrderItem is weak (depends on Order and Product) — its PK is composite (OrderID, ProductID) and it stores quantity and price at order time.

3. Many-to-many needing attributes: Order and Product have M:N via OrderItem; OrderItem stores Quantity and PriceAtOrder which are attributes of the relationship.

**Part 4: Normalization Workshop  
Task 4.1**

1. Functional Dependencies (FDs):

* StudentID → StudentName, StudentMajor
* ProjectID → ProjectTitle, ProjectType, SupervisorID
* SupervisorID → SupervisorName, SupervisorDept
* (StudentID, ProjectID) → Role, HoursWorked, StartDate, EndDate

2. Redundancy and anomalies:

* Redundancy: StudentName and StudentMajor repeated for each project of a student; SupervisorName and SupervisorDept repeated for each project supervised by same supervisor.
* Update anomaly example: Changing SupervisorName requires updating many rows.
* Insert anomaly example: To add a new supervisor with no project yet, ProjectID required (depending on PK design).
* Delete anomaly example: Deleting last project by a student may remove supervisor info if stored only in that row.

3. 1NF: No repeating groups apparent; ensure multi-valued attributes (if any) are moved to separate tables (e.g., multiple supervisors or roles). Assume table is in 1NF.

4. 2NF: Primary key = (StudentID, ProjectID) assuming each student may work on multiple projects. Partial dependencies: StudentID → StudentName, StudentMajor (depends only on StudentID) and ProjectID → ProjectTitle, ProjectType, SupervisorID (depends only on ProjectID).

2NF decomposition:

* + Student(StudentID PK, StudentName, StudentMajor)
  + Project(ProjectID PK, ProjectTitle, ProjectType, SupervisorID)
  + StudentProject(StudentID FK, ProjectID FK, Role, HoursWorked, StartDate, EndDate)

5. 3NF: Transitive dependency: SupervisorID → SupervisorName, SupervisorDept in Project table; to remove transitive dependency, create Supervisor/Professor table:

* Supervisor(SupervisorID PK, SupervisorName, SupervisorDept)

Final 3NF tables: Student, Supervisor, Project (with SupervisorID FK), StudentProject (associative).

**Task 4.2**

1. Primary key: (StudentID, CourseID, TimeSlot) or more precisely (StudentID, CourseSectionID) if a CourseSectionID exists. Reason: student can take multiple course sections; a course section is uniquely defined by CourseID+TimeSlot+Room (or a SectionID).

2. Functional dependencies:

* + StudentID → StudentMajor
  + CourseID → CourseName
  + InstructorID → InstructorName
  + Room → Building (rooms unique across campus)
  + (CourseID, TimeSlot, Room) → InstructorID (each course section taught by one instructor at one time in one room)

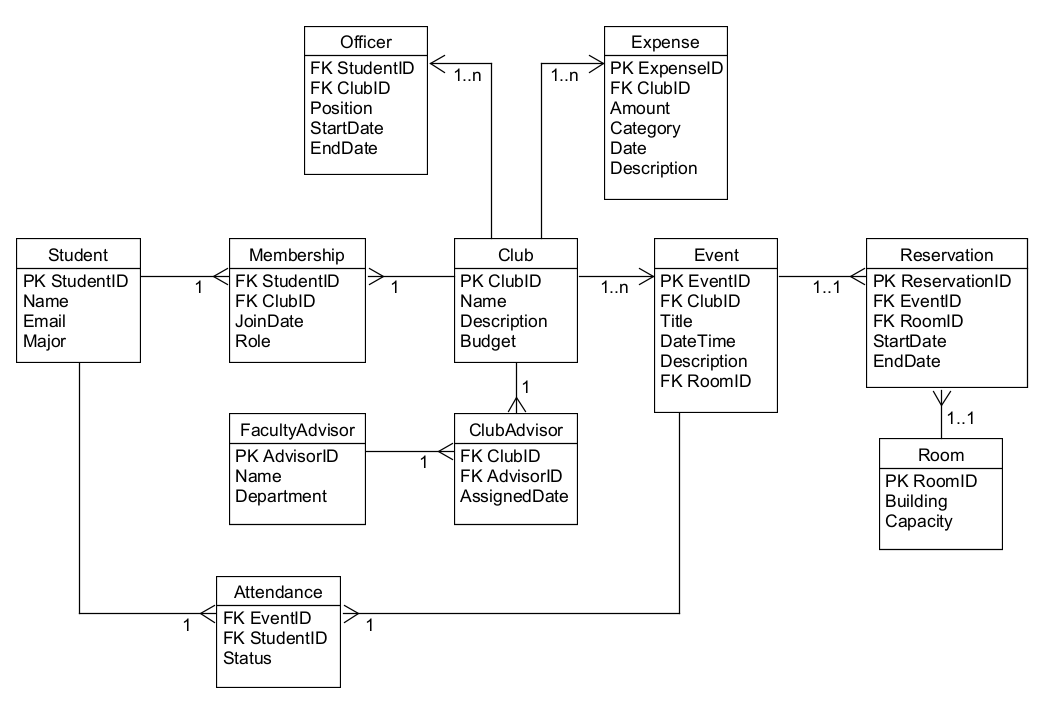
3. BCNF check: The table is not in BCNF because StudentID → StudentMajor (non-key → attribute) and Room → Building are FDs violating BCNF if keys include StudentID+CourseID+TimeSlot.

4. BCNF decomposition:

* + Student(StudentID PK, StudentMajor)
  + Course(CourseID PK, CourseName)
  + Instructor(InstructorID PK, InstructorName)
  + Room(Room PK, Building)
  + CourseSection(CourseSectionID PK, CourseID FK, InstructorID FK, TimeSlot, Room FK)
  + Enrollment(StudentID FK, CourseSectionID FK, PRIMARY KEY(StudentID, CourseSectionID))

5. Decomposition is lossless because CourseSection references original course+time+room combination; Enrollment links students to sections. No information loss if FKs are maintained; ensure data migration preserves CourseSection identity.

**Part 5: Design Challenge — University Clubs System  
Task 5.1**1-2.

  
  
3. Officer positions could be stored as attributes on Membership (e.g., isPresident flag) versus a separate Officer table. I chose a separate Officer table because a student can hold multiple positions over time and historical records must be kept.

4.

* "Find all students who are officers in the Computer Science Club."
* "List all events scheduled for next week with their room reservations."
* "Show all clubs with current budget totals and total expenses for the current semester."