Part 1: Key Identification Exercises

Task 1.1 – Superkeys and Candidate Keys

Relation A: Employee(EmpID, SSN, Email, Phone, Name, Department, Salary)

Assumptions grounded in common practice: EmpID, SSN, Email, and Phone are unique per employee in the sample; Name, Department, Salary are not unique.

A1) Examples of Superkeys

- {EmpID}
- {SSN}
- {Email}
- {Phone}
- {EmpID, SSN}
- {EmpID, Email}
- {EmpID, Phone}
- {SSN, Email}
- {SSN, Phone}
- {Email, Phone}
- {EmpID, SSN, Email}
- {EmpID, SSN, Email, Phone}

A2) Candidate Keys

Minimal superkeys (no proper subset is a key):

- {EmpID}
- {SSN}
- {Email}
- {Phone}

A3) Primary Key Choice & Rationale

Choose EmpID as the primary key. Because not subject to confidentiality restrictions, no changes by users

A4) Can two employees share the same phone?

From the sample rows each phone is distinct; in real systems a shared office phone can exist

Relation B: Registration(StudentID, CourseCode, Section, Semester, Year, Grade, Credits)

B1) Minimum Attributes for the Primary Key

Business rules imply uniqueness by student-course-section-semester-year. Use a natural composite key: {StudentID, CourseCode, Section, Semester, Year}.

B2) Why Each Attribute is Necessary

- StudentID – identifies the student.
- CourseCode – identifies the course.
- Section – students cannot register for the same course section in the same semester.
- Semester – the same course may be taken in different semesters.
- Year – disambiguates Fall/Spring of different years.

B3) Additional Candidate Keys

If the institution generates a RegistrationID as a surrogate, then {RegistrationID} would also be a candidate key. Otherwise, no smaller natural candidate exists under the given rules.

Task 1.2 – Foreign Key Design

```
Student(StudentID PK, Name, Email, Major, AdvisorID FK -> Professor.ProfID)

Professor(ProfID PK, Name, Department, Salary)

Course(CourseID PK, Title, Credits, DepartmentCode FK -> Department.DeptCode)

Department(DeptCode PK, DeptName, Budget, ChairID FK -> Professor.ProfID)

Enrollment(StudentID FK -> Student.StudentID,
           CourseID  FK -> Course.CourseID,
           Semester, Grade,
           PK(StudentID, CourseID, Semester))
```

Notes:

- AdvisorID in Student references Professor (optional if some students have no advisor).
- ChairID in Department references Professor (assume each department has one chair who is a professor).
- Enrollment uses a composite PK; add Year if your calendar requires it, e.g., PK(StudentID, CourseID, Semester, Year).

Part 2: ER Diagram Construction

Task 2.1 – Hospital Management System

Entities & Keys

- Patient(PatientID PK, Name, BirthDate, Address{Street, City, State, Zip}, Phone{multi}, InsuranceInfo) – Strong.

- Doctor(DoctorID PK, Name, Specialization{multi}, Phone{multi}, OfficeLocation) – Strong.
- Department(DeptCode PK, DeptName, Location) – Strong.
- Appointment(AppointmentID PK, DateTime, Purpose, Notes, PatientID FK, DoctorID FK) – Strong (tracks interactions).
- Prescription(PrescriptionID PK, Medication, Dosage, Instructions, PatientID FK, DoctorID FK, DatePrescribed) – Strong.
- Room(DeptCode FK, RoomNumber, PK(DeptCode, RoomNumber)) – Strong with composite PK.

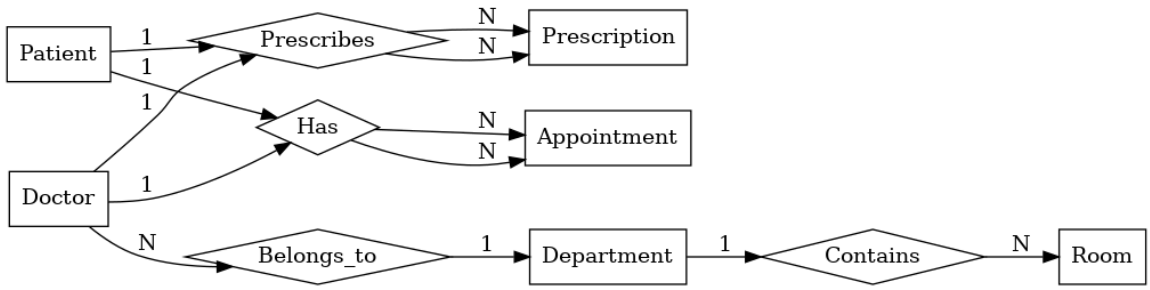
Attributes Classification

- Simple: PatientID, Name, BirthDate, DoctorID, DeptCode, DeptName, Location, AppointmentID, PrescriptionID.
- Composite: Address(Street, City, State, Zip).
- Multi-valued: Patient.Phone, Doctor.Phone, Doctor.Specialization.
- Derived (optional): PatientAge (from BirthDate).

Relationships & Cardinalities

- Patient–Appointment: 1:N (one patient has many appointments).
- Doctor–Appointment: 1:N (one doctor has many appointments).
- Doctor–Department: N:1 (many doctors belong to one department).
- Patient–Prescription: 1:N; Doctor–Prescription: 1:N.
- Department–Room: 1:N (rooms numbered per department).

ER Notation



Task 2.2 – E- commerce Platform

Entities & Relationships

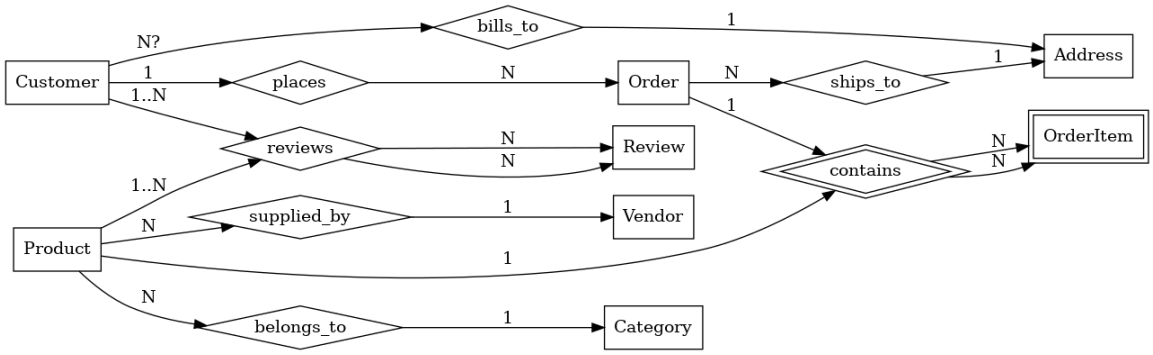
- Customer(CustomerID PK, Name, Email, Phone, BillingAddressID FK -> Address.AddressID).
- Address(AddressID PK, Street, City, State, Zip, Country).

- Order(OrderID PK, CustomerID FK, OrderDate, ShippingAddressID FK -> Address.AddressID, Status, TotalAtOrder).
- OrderItem(OrderID FK, ProductID FK, Quantity, PriceAtOrder, PK(OrderID, ProductID)).
- Product(ProductID PK, Name, SKU, UnitPrice, CategoryID FK, VendorID FK, InventoryLevel).
- Category(CategoryID PK, CategoryName, ParentCategoryID FK -> Category.CategoryID NULL).
- Vendor(VendorID PK, VendorName, ContactEmail, Phone).
- Review(ReviewID PK, ProductID FK, CustomerID FK, Rating, Comment, ReviewDate).

Weak entity: OrderItem is weak relative to Order and Product; it has a composite PK (OrderID, ProductID) and depends on its parent order for existence.

M:N needing attributes: Customer–Product through Review (attributes: Rating, Comment, ReviewDate).

ER Notation



Part 4: Normalization Workshop

Task 4.1 – Denormalized Table Analysis

Table: StudentProject(StudentID, StudentName, StudentMajor, ProjectID, ProjectTitle, ProjectType, SupervisorID, SupervisorName, SupervisorDept, Role, HoursWorked, StartDate, EndDate)

1) Functional Dependencies (typical university rules)

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

2) Redundancy & Anomalies

- Redundancy: repeating StudentName/Major for each project row; repeating SupervisorName/Dept per project.
- Update anomaly: changing a supervisor's department requires updating many rows.
- Insert anomaly: cannot store a new project (with supervisor) until at least one student is assigned.
- Delete anomaly: removing the last student on a project loses the project and supervisor link.

3) 1NF Check

Assume atomic attributes (no repeating groups). If Role or Phone were multi-valued, split into separate rows.

Result: keep as-is for 1NF if all attributes are atomic.

4) 2NF (identify PK and remove partial dependencies)

Natural PK: (StudentID, ProjectID). Partial dependencies exist: StudentID → StudentName, StudentMajor; ProjectID → ProjectTitle, ProjectType, SupervisorID.

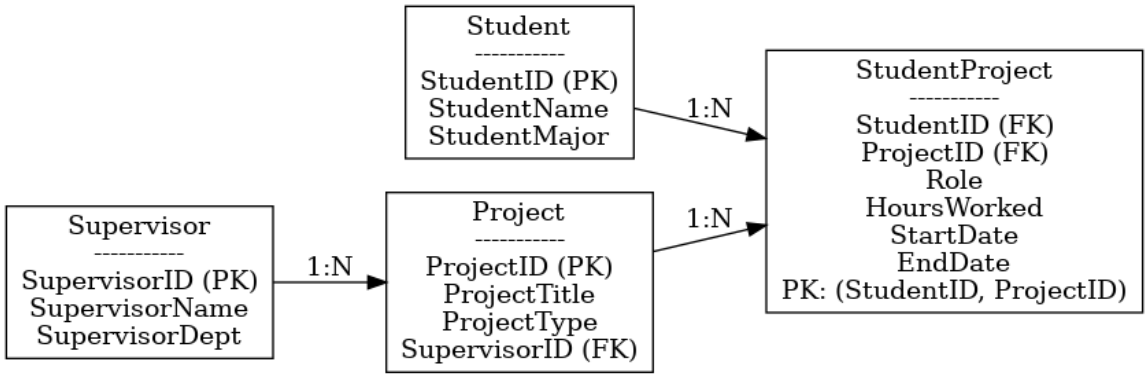
Decomposition to 2NF:

```
Student(StudentID PK, StudentName, StudentMajor)

Project(ProjectID PK, ProjectTitle, ProjectType, SupervisorID FK ->
Supervisor.SupervisorID)

Supervisor(SupervisorID PK, SupervisorName, SupervisorDept)

StudentProject(StudentID FK -> Student, ProjectID FK -> Project,

               Role, HoursWorked, StartDate, EndDate,

               PK(StudentID, ProjectID))
```

5) 3NF (remove transitive dependencies)

Transitive: ProjectID → SupervisorID and SupervisorID → SupervisorName, SupervisorDept. By isolating Supervisor in its own table and referencing from Project, we reach 3NF.

Final 3NF schema is exactly



Task 4.2 – Advanced Normalization (toward BCNF)

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

Business Rules Recap

- Each student has exactly one major.
- Each course has a fixed name.
- Each instructor has exactly one name.
- Each time slot in a room determines the building (rooms are unique across campus).
- Each course section is taught by one instructor at one time in one room.
- A student can enroll in multiple sections.

1) Primary Key

Enrollment grain is (StudentID, CourseID, TimeSlot) or (StudentID, SectionID). Because rules bind Instructor and Room to a specific section/time, the minimal determinant for a row including TimeSlot is: PK = (StudentID, CourseID, TimeSlot).

2) Functional Dependencies

- StudentID → StudentMajor
- CourseID → CourseName
- InstructorID → InstructorName
- (Room, TimeSlot) → Building (or equivalently Room → Building if rooms are globally unique)
- (CourseID, TimeSlot) → InstructorID, Room (defines a section mapping)
- (StudentID, CourseID, TimeSlot) → (all non-key enrollment attributes, e.g., derived Instructor/Room/Building)

3) BCNF Check

Violations: non-key determinants exist (StudentID, CourseID, InstructorID, Room).

4) Decomposition to BCNF

```
Student(StudentID PK, StudentMajor)

Course(CourseID PK, CourseName)

Instructor(InstructorID PK, InstructorName)

Room(Room PK, Building)            -- if Room uniquely determines Building

Section(CourseID FK, TimeSlot, InstructorID FK, Room FK,

        PK(CourseID, TimeSlot))    -- each course-time defines one section

Enrollment(StudentID FK, CourseID FK, TimeSlot,

           PK(StudentID, CourseID, TimeSlot),

           FK (CourseID, TimeSlot) -> Section)
```

All determinants are candidate keys in their own relations (BCNF).

5) Losslessness

The decomposition is lossless because Enrollment keeps the composite foreign key to Section and joins on keys. No spurious tuples are introduced, and dependencies are preserved across the schema set.

Part 5: Design Challenge – Student Clubs

Task 5.1 – ER & Normalized Schema

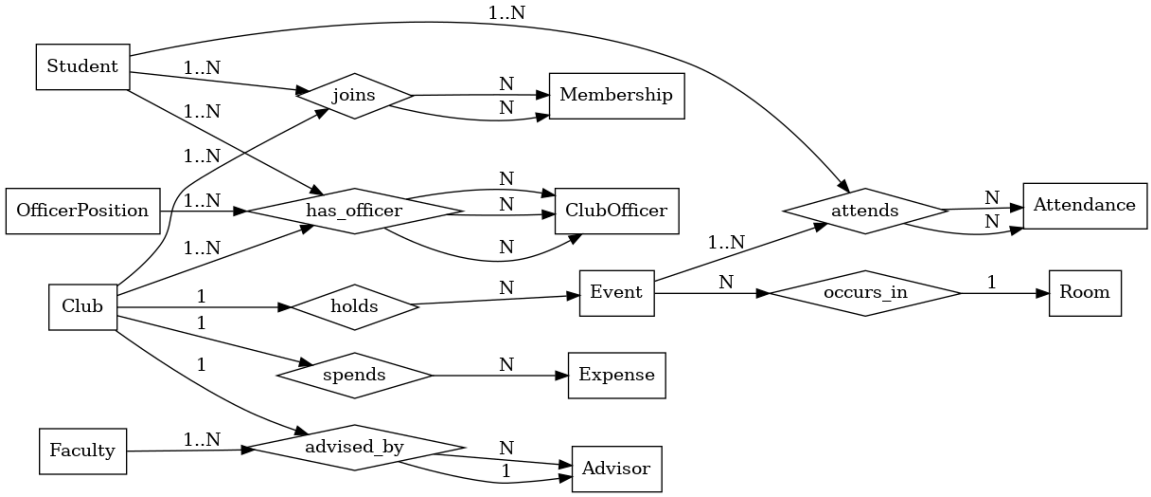
Entities

- Student(StudentID PK, Name, Email, Major, YearOfStudy)
- Club(ClubID PK, ClubName, Category, Budget)
- Membership(StudentID FK, ClubID FK, JoinDate, Role, PK(StudentID, ClubID))
- Event(EventID PK, ClubID FK, Title, EventDateTime, RoomID FK, Description)
- Attendance(EventID FK, StudentID FK, Status, PK(EventID, StudentID))
- OfficerPosition(PositionID PK, PositionName)
- ClubOfficer(ClubID FK, StudentID FK, PositionID FK, StartDate, EndDate, PK(ClubID, StudentID, PositionID))
- Faculty(FacultyID PK, Name, Email, Department)
- Advisor(ClubID PK, FacultyID FK) -- one advisor per club
- Room(RoomID PK, Building, Number)
- Expense(ExpenseID PK, ClubID FK, Amount, Purpose, ExpenseDate)

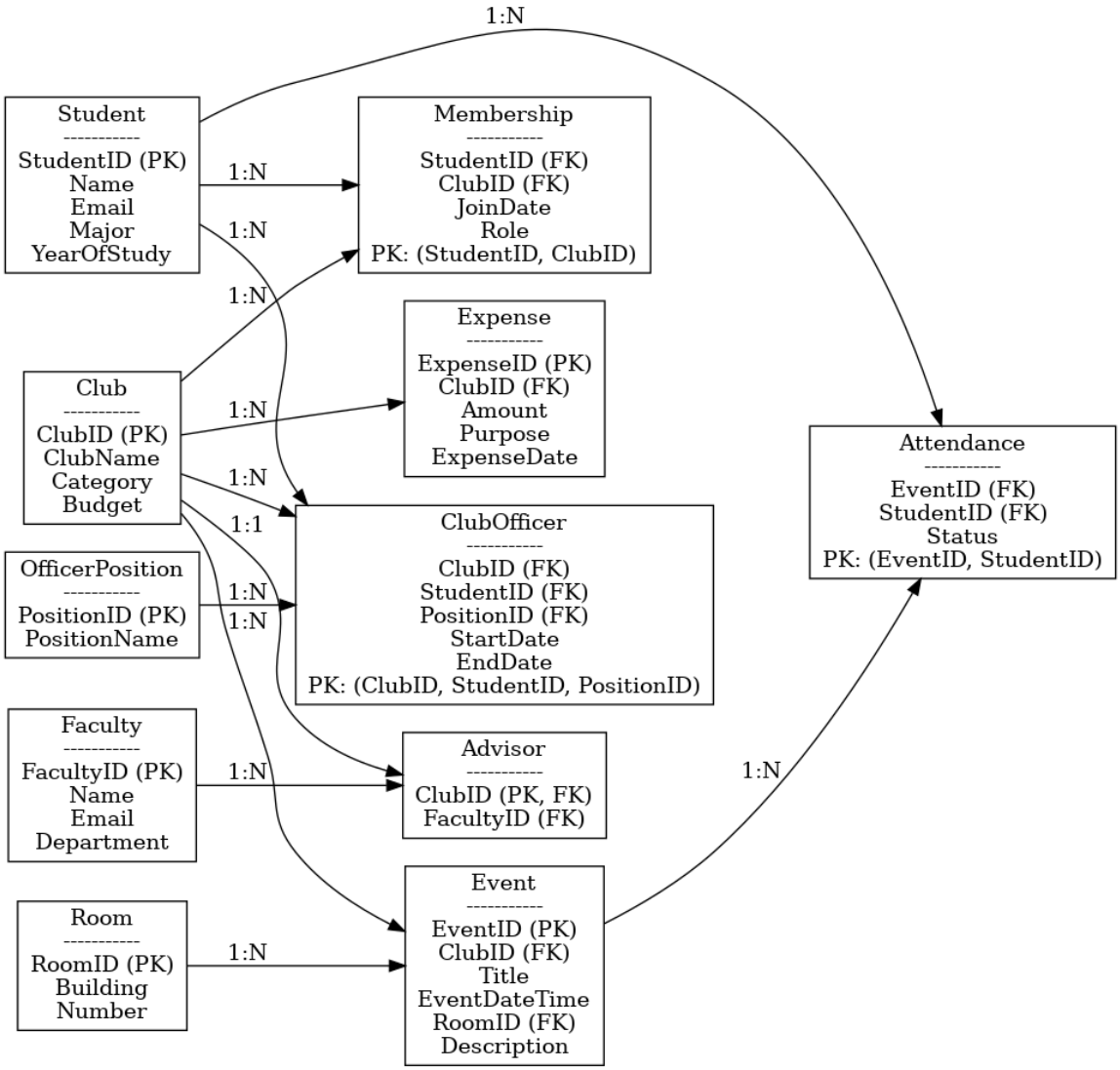
Relationships & Notes

- Student–Club: M:N via Membership (Role optional here for member-level role).
- Club–Event: 1:N; Event–Attendance: M:N (Attendance as bridge).
- Club officers: ClubOfficer ties a Student to a Position within a Club over a time window.
- Faculty advisors: Advisor ensures 1:1 club→advisor, while one faculty can advise many clubs.
- Room reservations: Event references Room.
- Budget tracking: Expense references Club.

ER diagram



Normalized Text Schema



Example Queries (English no SQL)

1. Find all students who are officers in the Computer Science Club.
2. List all events scheduled for next week with their room reservations and attending headcount.
3. Show each club's total expenses in the current semester compared to its budget.