Databases Lab 1

1 Part 1

1.1 Task 1.1

1.1.1 Relation A

1. **Superkeys:** Since the superkey is the set of columns that recognizes string uniquely.
   - {EmpID, Name}
   - {SSN, Department}
   - {Email, Department}
   - {Phone, Name}
   - {EmpID, Department, Salary}
   - {SSN, Phone}
2. **Candidate Keys:** Candidate key is the minimal superkey, from which if you take any column, it loses its unique property.
   - {EmpID}
   - {SSN}
   - {Email}
   - {Phone}
3. **Primary Key:** Primary key is like the "main" identifier. So, I will choose EmpID.
4. **Phone Uniqueness:** In real life, yes (shared home phone, office extensions). But in this schema, Phone is treated as unique (since it qualifies as a candidate key).

1.1.2 Relation B

1. **Superkey:** {StudentID, CourseCode, Section, Semester, Year}
2. **Why each attribute is necessary:**
   - *StudentID*: Identifies which student.
   - *CourseCode*: Identifies the course.
   - *Section*: Distinguishes multiple offerings of the same course in same semester.
   - *Semester* + *Year*: Needed because same course can be repeated across different semesters.
3. **No smaller combination works**, because removing any attribute could allow duplicates.

1.2 Task 1.2

1. **Foreign Key Relationships:**
   - Student.AdvisorID $\rightarrow$ Professor.ProfID (each student is advised by a professor).

- Course.DepartmentCode → Department.DeptCode (each course belongs to a department).
- Department.ChairID → Professor.ProfID (each department has a professor as chair).
- Enrollment.StudentID → Student.StudentID (each enrollment belongs to a student).
- Enrollment.CourseID → Course.CourseID (each enrollment refers to a course).

2 Part 2

2.1 Task 2.1

1. **Entities (Strong / Weak):**
   - Patients (Strong)
   - Doctors (Strong)
   - Departments (Strong)
   - Appointments (Weak)
   - Prescriptions (Weak)
   - Hospital Rooms (Weak)
2. **Attributes:**
   - *Patients (Strong)*: PatientID (PK), Name, Birthdate
   - *Doctors (Strong)*: DoctorID (PK), Name, Specialization
   - *Departments (Strong)*: DeptCode (PK), DeptName
   - *Appointments (Weak)*: AppointmentID (PK), DateTime, PatientID (FK), DoctorID (FK)
   - *Prescriptions (Weak)*: PrescriptionID (PK), Medication, PatientID (FK), DoctorID (FK)
   - *Hospital Rooms (Weak)*: DeptCode (FK), RoomNumber (partial key)
3. **Relationships and Cardinalities:**
   - Patient — Appointment — Doctor → M:N (through Appointment)
   - Doctor — Prescription — Patient → M:N (through Prescription)
   - Department — Doctor → 1:N (one department has many doctors)
   - Department — HospitalRoom → 1:N (one department has many rooms)
   - Patient — Prescription → 1:N (patient can have many prescriptions)
4. **ER Diagram:** at the end

2.2 Task 2.2

1. **ER Diagram:** at the end
2. **OrderItems is a weak entity:**
   - It cannot exist without both an Order and a Product.

- Its primary key is composite: {OrderID, ProductID}.
- Attributes such as Quantity and PriceAtOrder belong to the relationship between Orders and Products, not to either entity alone.
- *Justification*: OrderItems is weak because its identity depends on two parent entities (Orders and Products).

3. **Relationship between Orders and Products is many-to-many (M:N):**
   - One order can include many products.
   - One product can appear in many orders.
   - To represent this properly, we use the weak entity OrderItems, which has the attributes:
     - Quantity
     - PriceAtOrder
   - *Justification*: These attributes describe the relationship between Orders and Products (e.g., how many of that product were bought, and at what price), not the entities themselves.

3 Task 4.1

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

1. **Functional Dependencies:**
   - StudentID $\rightarrow$ StudentName, StudentMajor
   - ProjectID $\rightarrow$ ProjectTitle, ProjectType, SupervisorID
   - SupervisorID $\rightarrow$ SupervisorName, SupervisorDept
   - (StudentID, ProjectID) $\rightarrow$ Role, HoursWorked, StartDate, EndDate
2. **Problems (Redundancy & Anomalies):**
   - *Redundancy*: Student info, Supervisor info, and Project info are repeated.
   - *Update anomaly*: Supervisor's department change requires multiple row updates.
   - *Insert anomaly*: Cannot add a new project without a student assigned.
   - *Delete anomaly*: If last student leaves a project, project info is lost.
3. **Apply 1NF:** All attributes are atomic. Table is already in 1NF.
4. **Apply 2NF:**
   - *Primary Key*: (StudentID, ProjectID)
   - *Partial dependencies*:
     - StudentID $\rightarrow$ StudentName, StudentMajor

  - ProjectID → ProjectTitle, ProjectType, SupervisorID
  - SupervisorID → SupervisorName, SupervisorDept
- *Decomposition into 2NF*:
  (a) Student(StudentID, StudentName, StudentMajor)
  (b) Project(ProjectID, ProjectTitle, ProjectType, SupervisorID)
  (c) Supervisor(SupervisorID, SupervisorName, SupervisorDept)
  (d) StudentProject(StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate)

5. **Apply 3NF:**
   - Remove transitive dependency (ProjectID → SupervisorID → SupervisorName, SupervisorDept).
   - *Final 3NF decomposition*:
     (a) Student(StudentID, StudentName, StudentMajor)
     (b) Supervisor(SupervisorID, SupervisorName, SupervisorDept)
     (c) Project(ProjectID, ProjectTitle, ProjectType, SupervisorID)
     (d) StudentProject(StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate)

4 Task 4.2

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

1. **Primary Key:** (StudentID, CourseID, TimeSlot, Room)
2. **Functional Dependencies:**
   - StudentID → StudentMajor
   - CourseID → CourseName
   - InstructorID → InstructorName
   - Room → Building
   - (CourseID, TimeSlot, Room) → InstructorID
   - (StudentID, CourseID, TimeSlot, Room) → all attributes
3. **BCNF Check:**
   - Not in BCNF. Violations:
     - StudentID → StudentMajor
     - CourseID → CourseName
     - InstructorID → InstructorName
     - Room → Building

- (determinants are not superkeys).

4. **Decomposition to BCNF:**
   - *Decomposed tables*:
     (a) Student(StudentID, StudentMajor)
     (b) Course(CourseID, CourseName)
     (c) Instructor(InstructorID, InstructorName)
     (d) Room(Room, Building)
     (e) CourseSection(CourseID, TimeSlot, Room, InstructorID) (PK = (CourseID, TimeSlot, Room))
     (f) Enrollment(StudentID, CourseID, TimeSlot, Room) (PK = (StudentID, CourseID, TimeSlot, Room), FKs: StudentID $\rightarrow$ Student, (CourseID, TimeSlot, Room) $\rightarrow$ CourseSection)
5. **Loss of Information:** No information is lost. All dependencies are preserved. Some queries may require joins.

5 Part 5

5.1 Task 5.1

ER Diagram: at the end

5.2 Task 5.2

5.2.1 Step 1: UNF (Unnormalized Form)

All information stored in one large table with repeating groups:

StudentID, StudentName, Email, Major, Year, ClubID, ClubName, ClubDesc, Budget, FacultyID, FacultyName, FacultyEmail, Dept, EventID, EventName, EventDate, RoomID, Building, RoomNumber, Capacity, AttendanceStatus, ExpenseID, Amount, ExpenseDesc, ExpenseDate, JoinDate, Role, OfficerPosition

Problems: Repeating groups (students in multiple clubs, clubs with multiple events). Update, insert, and delete anomalies.

5.2.2 Step 2: 1NF (First Normal Form)

Remove repeating groups, ensure atomic values, and split into separate tables: Now each attribute is atomic.

5.2.3 Step 3: 2NF (Second Normal Form)

Remove partial dependencies (attributes depending on part of a composite key). Membership $\rightarrow$ composite key (StudentID, ClubID). JoinDate and Role depend on full key. Attendance $\rightarrow$ composite key (StudentID, EventID). Status depends on full key. Already satisfied.

5.2.4 Step 4: 3NF (Third Normal Form)

Remove transitive dependencies. Example: Faculty details are separate (not inside CLUB). Officer positions standardized in a separate table. Now schema is in 3NF.

Table	Attributes
STUDENT	StudentID (PK), Name, Email, Major, Year
FACULTY	FacultyID (PK), Name, Email, Department
CLUB	ClubID (PK), ClubName, Description, Budget, FacultyID (FK)
ROOM	RoomID (PK), Building, RoomNumber, Capacity
EVENT	EventID (PK), EventName, EventDate, ClubID (FK), RoomID (FK)
EXPENSE	ExpenseID (PK), Amount, Description, Date, ClubID (FK)
MEMBERSHIP	StudentID (FK), ClubID (FK), JoinDate, Role
ATTENDANCE	StudentID (FK), EventID (FK), Status
OFFICERPOSITION	PositionID (PK), PositionName

5.2.5 Final Normalized Schema (3NF)

- STUDENT(StudentID, Name, Email, Major, Year)
- FACULTY(FacultyID, Name, Email, Department)
- CLUB(ClubID, ClubName, Description, Budget, FacultyID FK)
- MEMBERSHIP(StudentID, ClubID, JoinDate, Role)
- OFFICERPOSITION(PositionID, PositionName)
- MEMBERSHIP_OFFICER(StudentID, ClubID, PositionID) (optional, if strict officer role enforcement)
- ROOM(RoomID, Building, RoomNumber, Capacity)
- EVENT(EventID, ClubID, EventName, EventDate, RoomID FK)
- ATTENDANCE(StudentID, EventID, Status)
- EXPENSE(ExpenseID, ClubID, Amount, Description, Date)

5.3 Task 5.3

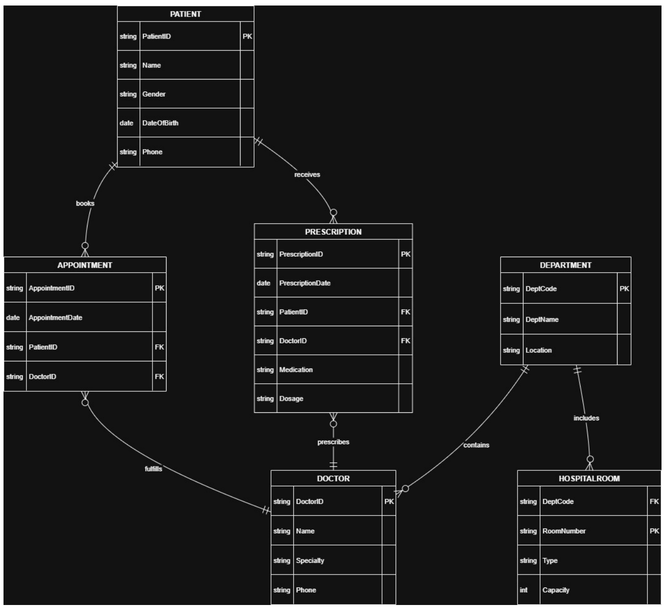
Create a separate OFFICERPOSITION table and link it to membership using a junction table MEMBERSHIP_OFFICER.

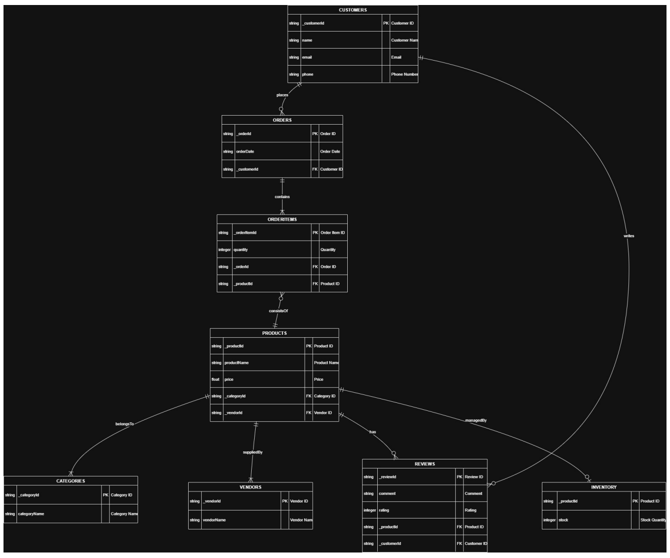
- Ensures consistency (all roles predefined).
- Supports students holding multiple positions.
- Easier to update officer role names globally.
- Slightly more complex design.

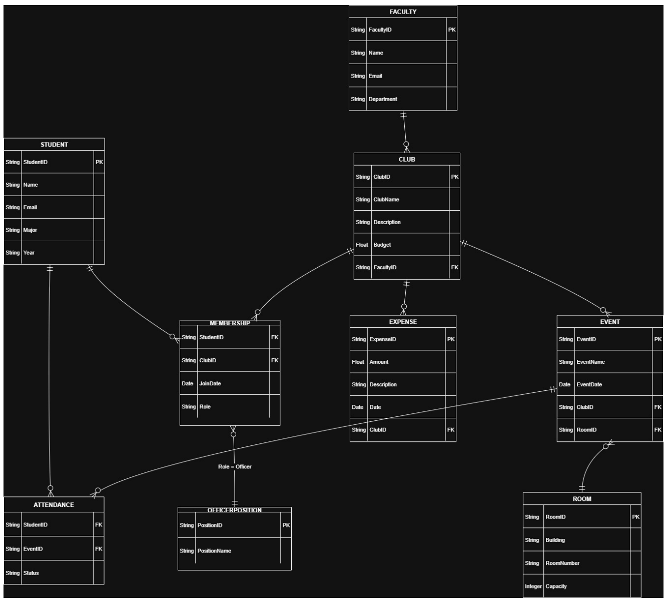
5.4 Task 5.4

1. List all students who are members of the Photography Club, along with their roles and join dates.
2. Show all upcoming events, including the club name, event date, and the reserved room.

3. Find the total expenses for each club this semester, and compare it to the allocated budget.




CUSTOMERS
string _customerId PK Customer ID
string name Customer Name
string email Email
string phone Phone Number
places
ORDERS
string _orderId PK Order ID
string orderDate Order Date
string _customerId FK Customer ID
contains
ORDERITEMS
string _orderItemId PK Order Item ID
integer quantity Quantity
string _orderId FK Order ID
string _productId FK Product ID
writes
consistsOf
PRODUCTS
string _productId PK Product ID
string productName Product Name
float price Price
string _categoryId FK Category ID
string _vendorId FK Vendor ID
belongsTo
suppliedBy
has
managedBy
CATEGORIES
string _categoryId PK Category ID
string categoryName Category Name
VENDORS
string _vendorId PK Vendor ID
string vendorName Vendor Name
REVIEWS
string _reviewId PK Review ID
string comment Comment
integer rating Rating
string _productId FK Product ID
string _customerId FK Customer ID
INVENTORY
string _productId PK Product ID
integer stock Stock Quantity


FACULTY
String FacultyID PK
String Name
String Email
String Department
STUDENT
String StudentID PK
String Name
String Email
String Major
String Year
CLUB
String ClubID PK
String ClubName
String Description
Float Budget
String FacultyID FK
MEMBERSHIP
String StudentID FK
String ClubID FK
Date JoinDate
String Role
EXPENSE
String ExpenseID PK
Float Amount
String Description
Date Date
String ClubID FK
EVENT
String EventID PK
String EventName
Date EventDate
String ClubID FK
String RoomID FK
Role = Officer
ATTENDANCE
String StudentID FK
String EventID FK
String Status
OFFICERPOSITION
String PositionID PK
String PositionName
ROOM
String RoomID PK
String Building
String RoomNumber
Integer Capacity