### **Task 1.1**

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

**Task 1:**

{EmpID},{SSN},{Email},{EmpID, Name},{Phone},{EmpID, SSN, Email}

**Task 2:**

{EmpID} (unique ID assigned by company)

{SSN} (unique)

{Email} (unique)

{Phone} (unique)

**Task 3:**

**EmpID,** because it is short, numeric, and company-controlled

**Task 4:**

Based on the given data, no duplicates exist, but it is possible if employees are married or from one family.

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

**Business rules given:**

**Student can take same course in different semesters.**

**Student cannot register for the same course section in the same semester.**

**Each course section in a semester has fixed credit value.**

**Task 1:**

Min attributes for PK: {StudentID, CourseCode, Section, Semester, Year}

**Task 2:**

StudentID identifies which student.

CourseCode and Section identifies which offering of a course.

Semester and Year distinguishes between offerings across time.

**Task 3:**

{CourseCode, Section, Semester, Year}

### **Task 1.2**

Given Tables:

Student(StudentID, Name, Email, Major, AdvisorID)

Professor(ProfID, Name, Department, Salary)

Course(CourseID, Title, Credits, DepartmentCode)

Department(DeptCode, DeptName, Budget, ChairID)

Enrollment(StudentID, CourseID, Semester, Grade)

**Task 1: Foreign Keys**

**Student.AdvisorID and Professor.ProfID** (each student has one advisor)

**Course.DepartmentCode and Department.DeptCode** (course belongs to a department)

**Department.ChairID and Professor.ProfID** (department has one professor as chair)

**Enrollment.StudentID and Student.StudentID** (enrollment refers to student)

**Enrollment.CourseID and Course.CourseID** (enrollment refers to course)

## **Task 2.1**

**Task 1:**

**Patient (strong)** - PatientID (PK), Name, Birthdate, Address (Street, City, State, Zip), InsuranceInfo

**Phone (weak)** - PhoneNo, linked to PatientID

**Doctor (strong)** - DoctorID (PK), Name, OfficeLocation

**Specialization (weak)** - SpecializationName, linked to DoctorID

**Department (strong)** - DeptCode (PK), DeptName, Location

**Appointment (weak)** - AppointmentID (PK), DateTime, Purpose, Notes

**Prescription (weak)** - PrescriptionID (PK), Medication, Dosage, Instructions

**Room (strong)** - (DeptCode, RoomNo) composite PK

**Task 2:**

**Patient:**

* PatientID - simple, PK
* Name - simple
* Birthdate - simple
* Address - composite (Street, City, State, Zip)
* PhoneNo - multi-valued
* InsuranceInfo - simple

**Doctor:**

* DoctorID - simple, PK
* Name - simple
* Specialization - multi-valued
* Phone - multi-valued
* OfficeLocation - simple

**Department:**

* DeptCode - simple, PK
* DeptName - simple
* Location - simple

**Appointment:**

* AppointmentID - PK
* DateTime, Purpose, Notes - simple

**Prescription:**

* PrescriptionID - PK
* Medication, Dosage, Instructions - simple

**Room:**

* DeptCode, RoomNo - composite PK

### **Task 3**

Patient – Appointment – Doctor (M:N via Appointment)

Doctor – Prescription – Patient (M:N via Prescription)

Department – Doctor (1:N, department has many doctors)

Department – Room (1:N, department has many rooms)

### **Step 4 – ER Diagram (Description since I can’t draw here):**

* Strong entities: Patient, Doctor, Department, Room
* Weak entities: Phone (of patient), Specialization (of doctor), Appointment, Prescription
* PKs underlined, FKs indicated
* Cardinalities:
  + Patient(1) ↔ (M) Appointment (M) ↔ (1) Doctor
  + Doctor(1) ↔ (M) Prescription (M) ↔ (1) Patient
  + Department(1) ↔ (M) Doctor
  + Department(1) ↔ (M) Room

## **Task 2.2: E-commerce Platform**

### **Step 1 – Entities:**

* **Customer (strong)** → CustomerID (PK), Name, Email, BillingAddress
* **Address (weak)** → AddressID (PK), Street, City, Zip, linked to Customer (for shipping)
* **Order (strong)** → OrderID (PK), OrderDate, TotalAmount, CustomerID (FK)
* **OrderItem (weak)** → (OrderID, ProductID) as composite PK, Quantity, PriceAtOrder
* **Product (strong)** → ProductID (PK), Name, Price, StockQty
* **Category (strong)** → CategoryID (PK), CategoryName
* **Vendor (strong)** → VendorID (PK), Name, ContactInfo
* **Review (weak)** → ReviewID (PK), Rating, Comment, CustomerID (FK), ProductID (FK)

### **Step 2 – Relationships:**

* Customer – Order (1:N)
* Order – OrderItem – Product (M:N via OrderItem with attributes Quantity, PriceAtOrder)
* Product – Category (M:1)
* Product – Vendor (M:1)
* Product – Review – Customer (M:N via Review)
* Customer – Address (1:N, multiple shipping addresses possible)

### **Step 3 – Weak Entities & Justification:**

* **OrderItem** is weak → depends on both Order and Product.
* **Review** is weak → depends on both Customer and Product.
* **Address** can be weak if tied only to Customer (not standalone).

### **Step 4 – Many-to-Many with Attributes:**

* **OrderItem** is M:N between Order and Product, with attributes *Quantity, PriceAtOrder*.
* **Review** is M:N between Customer and Product, with attributes *Rating, Comment*.

## **Task 4.1**

### **Given Table:**

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

### **Task 1:**

StudentID → StudentName, StudentMajor

ProjectID → ProjectTitle, ProjectType, SupervisorID

SupervisorID → SupervisorName, SupervisorDept

{StudentID, ProjectID} → Role, HoursWorked, StartDate, EndDate

### **Task 2:**

**Redundancy:**

Student info repeated for every project

Supervisor info repeated for every project

**Update Anomaly:**

Changing a supervisor’s department requires updating multiple rows

**Insert Anomaly:**

Can’t add a new supervisor until a project is created

**Delete Anomaly:**

If last project for a supervisor is deleted, info about supervisor is lost

### **Task 3:**

Table is already atomic (no repeating groups)

### **Task 4:**

**Primary Key:** {StudentID, ProjectID} (composite key).

**Partial Dependencies:**

StudentID → StudentName, StudentMajor

ProjectID → ProjectTitle, ProjectType, SupervisorID

**Decomposition:**

Student(StudentID, StudentName, StudentMajor)

Project(ProjectID, ProjectTitle, ProjectType, SupervisorID)

Supervisor(SupervisorID, SupervisorName, SupervisorDept)

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

### **Task 5:**

**Transitive Dependency:** SupervisorID → SupervisorDept through SupervisorName.

Final 3NF decomposition:

Student(StudentID, StudentName, StudentMajor)

Supervisor(SupervisorID, SupervisorName, SupervisorDept)

Project(ProjectID, ProjectTitle, ProjectType, SupervisorID)

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

# **Task 4.2**

### **Given 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 course sections.

## **Task 1:**

Candidate PK = **{StudentID, CourseID, TimeSlot, Room}**

## **Task 2:**

StudentID → StudentMajor

CourseID → CourseName

InstructorID → InstructorName

(TimeSlot, Room) → Building

(CourseID, TimeSlot, Room) → InstructorID

(StudentID, CourseID, TimeSlot, Room) → all other attributes (by definition of PK)

## **Step 3 – Check Normal Forms**

* **1NF** - yes, attributes are atomic
* **2NF** - no, partial dependencies exist (StudentID → StudentMajor, CourseID → CourseName).
* **3NF** – no, transitive dependencies (InstructorID → InstructorName).
* **BCNF** – no, not satisfied, because some FDs have determinants that are not candidate keys (StudentID → Major).

## **Task 4:**

Break into separate tables based on FDs:

Student(StudentID, StudentMajor)

Course(CourseID, CourseName)

Instructor(InstructorID, InstructorName)

Room(Room, Building)

CourseSection(CourseID, TimeSlot, Room, InstructorID,   
 PRIMARY KEY(CourseID, TimeSlot, Room))

Enrollment(StudentID, CourseID, TimeSlot, Room,  
 PRIMARY KEY(StudentID, CourseID, TimeSlot, Room),  
 FOREIGN KEY (StudentID) REFERENCES Student(StudentID),  
 FOREIGN KEY (CourseID, TimeSlot, Room) REFERENCES CourseSection(CourseID, TimeSlot, Room))

## **Task 5:**

All original attributes are preserved across the new tables

Each FD is enforced in its own table

Decomposition is lossless and in BCNF