**Part 1: Key Identification Exercises**

**Task 1.1: Superkey and Candidate Key Analysis**

**Relation A: Employee**

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

**1. List at least 6 different superkeys** *(Superkey: Any combination of attributes that uniquely identifies each tuple)*

* {EmpID}
* {SSN}
* {Email}
* {EmpID, Email}
* {EmpID, Phone}
* {EmpID, Name}
* {EmpID, Name, Department}
* Etc;

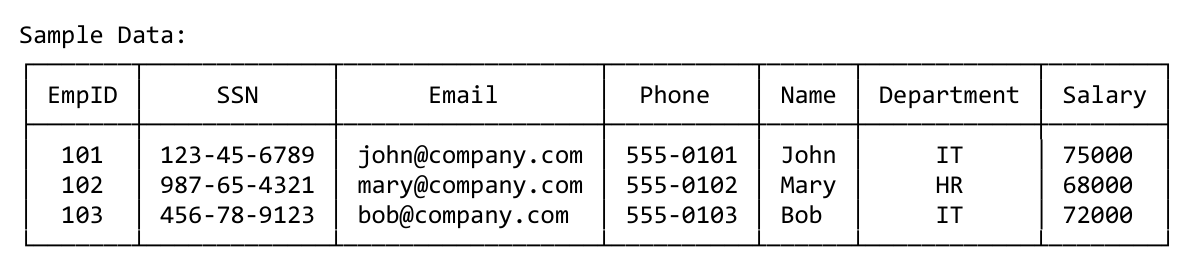
**2. Identify all candidate keys** *(Candidate Key: Minimal superkey (no proper subset is also a superkey)*

* {EmpID}
* {SSN}
* {Email}

**3. Which candidate key would you choose as primary key and why?**

* {EmpID} Cause its stable and unqiue, only one ID per employee. Email and phone can be changed, so it doesnt suit. SSN is unique and stable, but its sensitive and about privacy

**4. Can two employees have the same phone number? Justify your answer based on the data shown.**

****

* From the sample data all phone numbers are dictinct. We cannot conclude phone number as unique, unless its specified. Many big companies have shared office numbers. If its specified as only one unque phone number per employee, we can declare phone number as unique constraint and key.

**Relation B: Course Registration**

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

Business Rules:

- A student can take the same course in different semesters

- A student cannot register for the same course section in the same semester

- Each course section in a semester has a fixed credit value

**1. Determine the minimum attributes needed for the primary key**

* {StudentID, CourseCode, Section, Semester, Year}

**2. Explain why each attribute in your primary key is necessary**

* {StudentID} – identifies student who takes course
* {CourseCode} – identifies the course
* {Section} – prevent registring for the same course in the same semester
* {Semester, Year} – identifies the semester

**3. Identify any additional candidate keys (if they exist)**

* {StudentID, CourseCode, Semester, Year}

**Task 1.2: Foreign Key Design**

**Design the foreign key relationships for this university system:**

**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)**

**1. Identify all foreign key relationships**

* Course{DepartmentCode} – Department{DeptCode}
* Course{CourseID} – Enrollment{CourseID}
* Enrollment{StudentID} – Student{StudentID}
* Student{AdvisorID} – Professor{ProfID}
* Department{ChairID} – Professor{ProfID}
* Professor{Department} – Department{DeptName}

**Part 2: ER Diagram Construction**

**Task 2.1: Hospital Management System**

**Scenario: Design a database for a hospital management system.**

**Requirements:**

**• Patients have unique patient IDs, names, birthdates, addresses (street, city, state, zip), phone numbers (multiple allowed), and insurance information**

**• Doctors have unique doctor IDs, names, specializations (can have multiple), phone numbers, and office locations**

**• Departments have department codes, names, and locations**

**• Appointments track which patient sees which doctor at what date/time, the purpose of visit, and any notes**

**• Prescriptions track medications prescribed by doctors to patients, including dosage and instructions**

**• Hospital Rooms are numbered within departments (room 101 in Cardiology is different from room 101 in Neurology)**

**1. Identify all entities (specify which are strong and which are weak)**

**Strong**

* Patient
* Doctor
* Department

**Weak**

* Hospital Room
* Presciptions
* Appoinment

**2. Identify all attributes for each entity (classify as simple, composite, multi-valued, or derived)**

* Patient
  + PatientID – simple, PK
  + Name – composite(FirstName, LastName)
  + BirthDate – simple
  + Adress – composite(Street ,City, State, ZIP)
  + Phone – multi-valued
  + Insurance – composite(InsurerName, InsurerNumber)
* Doctor
  + DoctorID – simple. PK
  + Name – composite(FirstName, LastName)
  + Specializations – multi-valued
  + Phone – multi-valued
  + OfficeLoc – simple Hospital Room
* Department
  + DeptCode – simple, PK
  + DeptName – simple
  + Location – simple or composite
* Appoinment
  + AppointmentID – simple, PK, or PatientID, DoctorID, AppDateTime – composite
  + DateTime – simple
  + Purpose – simple
  + Notes – simple
* Prescription
  + PrescriptionID - simple, PK
  + DoctorID,PatientID, MedicationID – FK
  + DateIssued – simple
  + Dosage – simple
  + Instructions – simple
* HospitalRoom
  + DeptCode – simple, FK
  + RoomNumber – simple
  + DeptCode,RoomNumber – PK

**3. Identify all relationships with their cardinalities (1:1, 1:N, M:N)**

* Department 1 : N Doctor
* Doctor 1 : N Appointment
* Patient 1 : N Appointment
* Doctor 1 : N Prescription
* Patient 1 : N Prescription
* Department 1 : N HospitalRoom

**4. Draw the complete ER diagram using proper notation**

**5. Mark primary keys**

**Task 2.2: E-commerce Platform**

**Scenario: Design a simplified e-commerce database.**

**Requirements: • Customers place Orders for Products • Products belong to Categories and are supplied by Vendors • Orders contain multiple Order Items (quantity and price at time of order) • Products have reviews and ratings from customers • Track Inventory levels for each product • Shipping addresses can be different from customer billing addresses**

**1. Create a complete ER diagram**

**2. Identify at least one weak entity and justify why it’s weak**

**3. Identify at least one many-to-many relationship that needs attributes**

**Part 4: Normalization Workshop**

**Task 4.1: Denormalized Table Analysis**

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

**1. Identify functional dependencies: List all FDs in the format A → B**

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

**2. Identify problems: - What redundancy exists in this table? - Give specific examples of update, insert, and delete anomalies**

Redundancy

* Student info repeated for each project row for that student.
* Project info repeated for each student working on the same project.
* Supervisor info repeated for each project supervised by the same supervisor.

Examples

* Update anomaly: If supervisor’s name changes, many rows must be updated.
* Insert anomaly: To add a new project without any students yet, you may not be able to insert a row unless you allow NULLs for Student fields.
* Delete anomaly: Deleting the only student on a project will remove the only row containing the project’s data.

**3. Apply 1NF: Are there any 1NF violations? How would you fix them?**

* If all attributes are atomic (no repeating groups), the table is in 1NF.

Fix if violations exist

* Move multi-valued attributes to separate relation.

1NF

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

**4. Apply 2NF: - What is the primary key of this table? - Identify any partial dependencies - Show the 2NF decomposition**

Partial dependencies (attributes depending on part of composite key):

* StudentName, StudentMajor depend only on StudentID.
* ProjectTitle, ProjectType, SupervisorID depend only on ProjectID.

**2NF decomposition**  
Decompose into:

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

**5. Apply 3NF: - Identify any transitive dependencies - Show the final 3NF decomposition with all table schemas**

Transitive dependencies found:

* ProjectID → SupervisorID and SupervisorID → SupervisorName, SupervisorDept — so SupervisorName transitively depends on ProjectID.

**3NF decomposition (final)**

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

**Task 4.2: Advanced Normalization**

**Business Rules: • 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 be enrolled in multiple course sections**

**1. Determine the primary key of this table (hint: this is tricky!)**

**2. List all functional dependencies**

**3. Check if the table is in BCNF**

**4. If not in BCNF, decompose it to BCNF showing your work**

**5. Explain any potential loss of information in your decomposition**