**Task 1.1 Key identification exercises**

Relation A:

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

1) Superkeys: {EmpID}, {SSN}, {Email}, {Phone}, {EmpID, SSN}, {SSN, Email}, {EmpID, Email}, etc.

2) Candidate keys: {EmpID}, {SSN}, {Email}, {Phone}.

3) EmpID because: single-attribute key, automatically generated, efficient for indexing because it is numeric.

4) No, because {Phone} is a candidate key, so each phone must uniquely identify an employee.

Relation B:

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

1) {StudentID, CourseCode, Semester, Year}

2) StudentID: required to identify which student is registered; CourseCode: required to identify which course is being taken; Semester: necessary because student can take same course in different semesters; Year: needed to distinguish between different academic years

3) {StudentID, CourseCode, Section, Semester, Year} is candidate key because rules state that student cannot register for same course section in the same semester

**Task 1.2 Foreign key design**

Student.AdvisorID 🡪 Professor.ProfID

Professor.Department → Department.DeptCode

Course.DepartmentCode → Department.DeptCode

Department.ChairID → Professor.ProfID

Enrollment.StudentID → Student.StudentID

Enrollment.CourseID → Course.CourseID

**Task 2.1 Hospital Management System**

1) **Strong Entities:**

**Patient** (PatientID, Name, Birthdate, Address, InsuranceInfo); **Doctor** (DoctorID, Name, OfficeLocation); **Department** (DeptCode, DeptName, Location); **HospitalRoom** (RoomNumber, DeptCode)

**Weak Entities: Appointment** (tracks Patient-Doctor relationship); **Prescription** (tracks Doctor-Patient relationship with medications)

2) **Patient:**

**Primary key**: PatientID

**Attributes:** Name (simple), Birthdate (simple), Address (composite: Street, City, State, Zip), PhoneNumber (simple), InsuranceInfo (simple)

**Doctor:**

**Primary key**: DoctorID

**Attributes:** Name (simple), OfficeLocation (simple), PhoneNumber (simple), Specializations (multi-valued)

**Department:**

**Primary key**: DeptCode

**Attributes:** DeptName (simple), Location (simple)

**HospitalRoom:**

**Primary key**: (DeptCode, RoomNumber) [composite key]

Attributes: RoomNumber (simple)

**Appointment**

**Primary key**: AppointmentID

**Attributes:** DateTime (simple), Purpose (simple), Notes (simple)

**Foreign keys:** PatientID, DoctorID

**Prescription**

**Primary key**: PrescriptionID

**Attributes:** MedicationName (simple), Dosage (simple), Instructions (simple)

**Foreign keys:** PatientID, DoctorID

3) **Relations:**

**1)** **Patient – Appointment – Doctor**

Patient can have many appointments with doctors; doctor can see many patients.

**M:N** resolved by **Appointment** (with Date/Time, Purpose, Notes)

**2) Doctor – Prescription – Patient**

Doctor prescribes many prescriptions; patient can receive many prescriptions.

**M:N** resolved by **Prescription** (with medication details)

**3) Doctor – Department**

Each doctor belongs to **one** department; one department can have many doctors.

**1:N**

**Department – HospitalRoom**

Each department has many rooms; room numbers are unique within department (composite key).

**1:N**

**Task 2.2 E-commerce Platform**

2) Weak entity - **OrderItem** (OrderID, ProductID, Quantity, PriceAtOrder) → weak because it depends on both **Order** and **Product**

**3) Product – Review – Customer:** M:N **(**a customer can review many products, a product can be reviewed by many customers; resolved by Review entity)

**Task 4.1 Denormalized Table Analysis**

**Given table - StudentProject** (StudentID, StudentName, StudentMajor, ProjectID, ProjectTitle, ProjectType, SupervisorID, SupervisorName, SupervisorDept, Role, Hours Worked, StartDate, EndDate)

1) StudentID → StudentName, StudentMajor

SupervisorID → SupervisorName, SupervisorDept

ProjectID → ProjectTitle, ProjectType, SupervisorID, SupervisorName, SupervisorDept

{StudentID, ProjectID} → Role, HoursWorked, StartDate, EndDate, StudentName, StudentMajor, ProjectTitle, ProjectType, SupervisorID, SupervisorName, SupervisorDept

2) **Redundancy:** For every student working on the same project, all project details (ProjectTitle, ProjectType, SupervisorID, SupervisorName, SupervisorDept) are repeated. Similarly, every project a student works on repeats their StudentName and StudentMajor.

**Anomalies:**

**Update Anomaly:**Changing a supervisor's department (SupervisorDept) requires updating every row for every project they supervise. If one row is missed, the data becomes inconsistent.

**Insertion Anomaly:** A new project cannot be added to the system until at least one student is assigned to it. Similarly, a new professor (Supervisor) cannot be added until they are assigned to a project.

**Deletion Anomaly:** If the last student on a project is deleted from this table, all information about that project is also lost.

3) If an attribute like **Role** could contain multiple comma-separated values ("Coder, Tester"), it would violate 1NF. How to fix? - break any multi-valued attributes into separate atomic attributes.

4) **Primary key:** {StudentID, ProjectID}

**Partial Dependencies:**

StudentID → StudentName, StudentMajor (Part of the key determines these attributes)

ProjectID → ProjectTitle, ProjectType, SupervisorID, SupervisorName, SupervisorDept (Part of the key determines these attributes)

The attributes Role, HoursWorked, StartDate, and EndDate are fully functionally dependent on the entire composite key.

**Decomposition:**

**Student:** (StudentID, StudentName, StudentMajor)

**Project:** (ProjectID, ProjectTitle, ProjectType, SupervisorID, SupervisorName, SupervisorDept)

**Assignment:** (StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate)

5) **Transitive Dependencies:**

ProjectID → SupervisorID

SupervisorID → SupervisorName, SupervisorDept

SupervisorName and SupervisorDept are transitively dependent on ProjectID by SupervisorID.

**Decomposition:**

**Student** (StudentID, StudentName, StudentMajor)

Primary key: StudentID

**Supervisor** (SupervisorID, SupervisorName, SupervisorDept)

Primary key: SupervisorID

**Project** (ProjectID, ProjectTitle, ProjectType, SupervisorID)

Primary key: ProjectID

Foreign key: SupervisorID references Supervisor(SupervisorID)

**Assignment** (StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate)

Primary key: {StudentID, ProjectID}

Foreign key: StudentID references Student(StudentID)

Foreign key: ProjectID references Project(ProjectID)

**Task 4.2 Advanced Normalization**

**Given table – CourseSchedule** (StudentID, StudentMajor, CourseID, CourseName, InstructorID, InstructorName, TimeSlot, Room, Building)

1) **Primary key:** {StudentID, CourseID, TimeSlot}, this combination identifies a specific student's enrollment in a specific offering (section) of a course, which is defined by its time slot.

2) **Functional dependencies:**

StudentID → StudentMajor (Each student has one major)

CourseID → CourseName (Each course has a fixed name)

InstructorID → InstructorName (Each instructor has one name)

Room → Building (A room is in exactly one building)

{CourseID, TimeSlot} → InstructorID, Room, Building, CourseName, InstructorName

(A course at a specific time defines a section, which has one instructor and one room.)

{InstructorID, TimeSlot} → CourseID, Room, Building, CourseName

(An instructor can only teach one course in one room at a given time.)

{Room, TimeSlot} → CourseID, InstructorID, Building, CourseName, InstructorName

(A room can only host one course section at a given time.)

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

StudentID → StudentMajor: StudentID is not a superkey, this violates BCNF.

CourseID → CourseName: CourseID is not a superkey, this violates BCNF.

InstructorID → InstructorName: InstructorID is not a superkey, this violates BCNF.

Room → Building: Room is not a superkey, this violates BCNF.

{CourseID, TimeSlot} → ...: {CourseID, TimeSlot} is a candidate key (it determines everything else except StudentID and StudentMajor). This satisfies BCNF.

Since multiple functional dependencies have determinants that are not superkeys, table **is not in BCNF.**

4) **BCNF Decomposition:**

**Student(**StudentID**,** StudentMajor**)**

Primary key: StudentID

**Course(**CourseID**,** CourseName**)**

Primary key: CourseID

**Instructor(**InstructorID**,** InstructorName**)**

Primary key: InstructorID

**Room(**Room**,** Building**)**

Primary key: Room

**Section(**CourseID**,** TimeSlot**,** InstructorID**,** Room**)**

Primary key: {CourseID, TimeSlot}

Foreign key: CourseID references Course(CourseID)

Foreign key: InstructorID references Instructor(InstructorID)

Foreign key: Room references Room(Room)

**Enrollment(**StudentID**,** CourseID**,** TimeSlot**)**

Primary key: {StudentID, CourseID, TimeSlot}

Foreign key: StudentID references Student(StudentID)

Foreign key: {CourseID, TimeSlot} references Section(CourseID, TimeSlot)

5) The decomposition is lossless because each step of the decomposition was based on a functional dependency where the determinant became the primary key of a new table and was left behind in the original table as a foreign key.