Part 1

Task 1.1 **Relation A - Employee**

1)A superkey is any set of attributes that uniquely identifies a tuple. Based on the sample data, here are six examples:

-{EmpID, SSN, Email, Phone, Name, Department, Salary} (The entire relation)

-{EmpID}

-{SSN}

-{Email}

-{SSN, Phone}

-{EmpID, Email}

-{Email, Name} (Assuming Name is not unique, this may not be a superkey in a larger dataset. A safer superkey would be {SSN, Department})

2)A candidate key is a minimal superkey.

EmpID: Unique in the sample data.

SSN: Unique by definition (Social Security Number).

Email: Unique in the sample data (company email is typically unique per employee).

These are all single-attribute keys, so they are minimal.

3) I would choose **EmpID** as the primary key.

4) Based on the data shown, all phone numbers are unique. However, the sample size is very small. In a real-world scenario, it's entirely possible for two employees to share a phone number (e.g., a shared office line or a household with multiple employees). Therefore, **Phone cannot be considered a candidate key** based on this limited sample. The business rules would need to specify if this is allowed or not.

Task 1.1 **Relation B - Course Registration**

1)StudentID, CourseCode , Section , Semester & Year

2)

StudentID: Identifies *which* student.

CourseCode: Identifies *which* course.

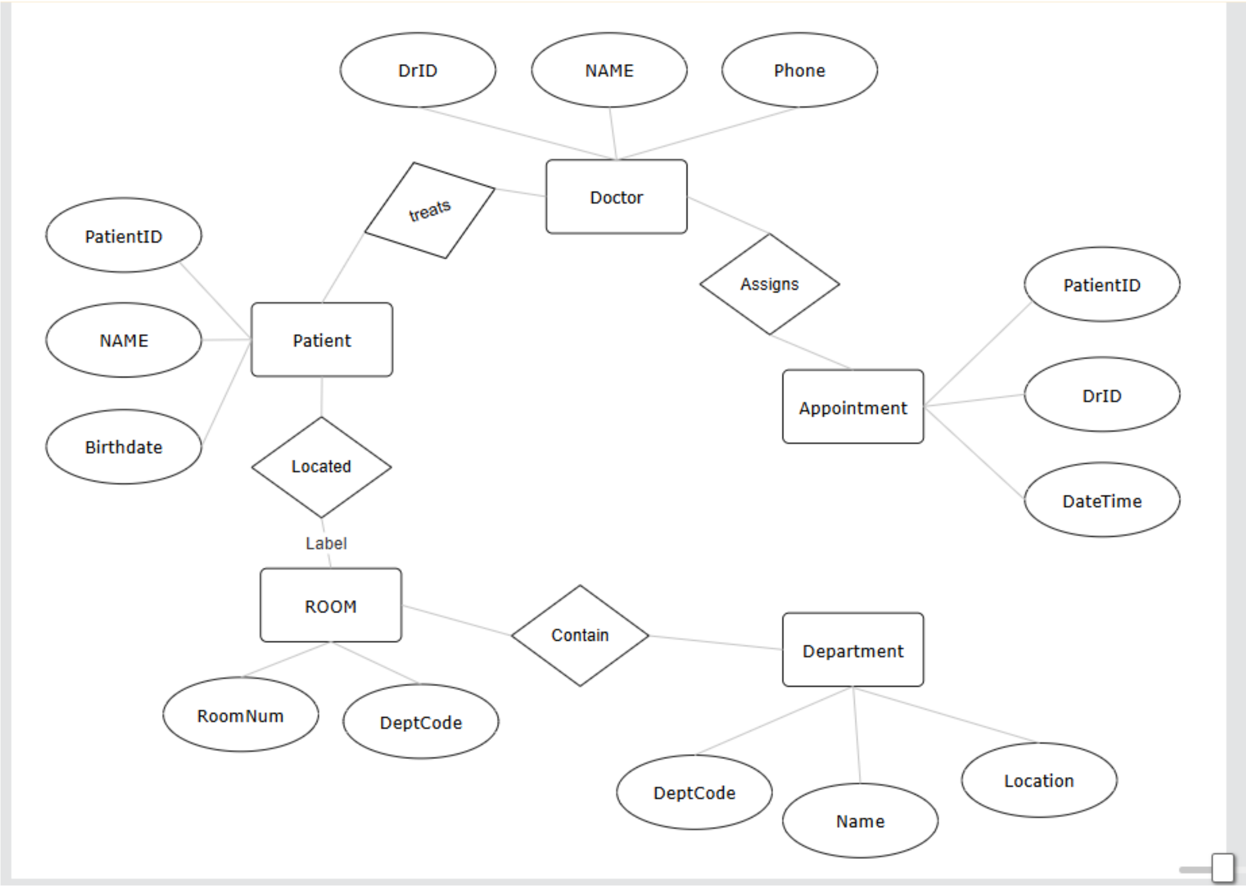
Section: Necessary because a course can have multiple sections (e.g., Lecture 1, Lab 3) in the same semester.

Semester & Year: Necessary because a student can re-take the same course (and section) in a future term. Without these, the combination of (StudentID, CourseCode, Section) would not be unique over time.

3) Another potential candidate key could be a synthetic key like RegistrationID. However, based on the given attributes, the composite key **{StudentID, CourseCode, Section, Semester, Year}** is the only natural candidate key.

Part 2

Task 2.1



Patient (Strong)
Doctor (Strong)
Department (Strong)
Appointment (Weak)
Prescription (Weak)
Room (Strong)
Phone (Weak, Multi-valued)

Attribute Classification:

Composite: Patient Address -> {Street, City, State, Zip}
Multi-valued: Doctor.Specialization. This would be modeled as a separate weak entity Specialization(DoctorID, Specialization).
Derived: (Possible) Patient.Age (derived from Birthdate).

Relationships & Cardinalities:

Patient *makes* Appointment (1:N) *(A patient can have many appointments, an appointment is for one patient)*

Doctor *has* Appointment (1:N) *(A doctor can have many appointments, an appointment is with one doctor)*

Doctor *works_in* Department (N:1) *(A doctor works in one department, a department has many doctors)*

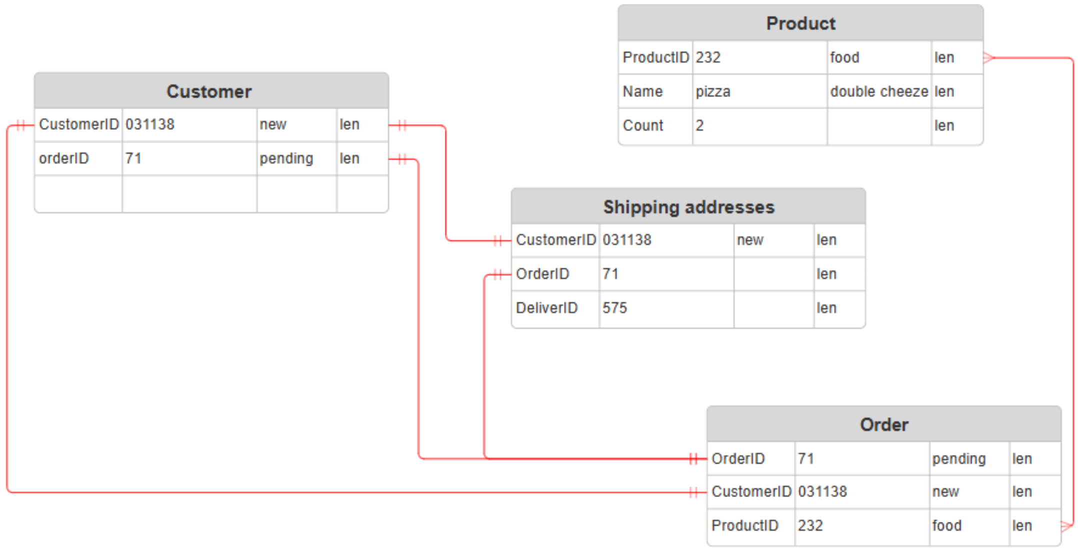
Doctor *prescribes* Prescription (1:N)

Patient *is_prescribed* Prescription (1:N)

Room *is_located_in* Department (N:1) *(A room belongs to one department, a department has many rooms)*

Patient *has* Phone (1:N) *(Modeling the multi-valued attribute as an entity)*

Task 2.2



Weak Entity: OrderItem. It is weak because its existence is dependent on the Order entity. An OrderItem cannot exist without an Order. Its primary key would be a composite of OrderID (from its owner, Order) and ProductID or a line item number.

Many-to-Many with Attributes: The relationship between Order and Product is M:N. This relationship itself has attributes Quantity and PriceAtTimeOfOrder. This is precisely why we create the associative entity OrderItem to hold these attributes.

Part4 : **Normalization Workshop**

Task 4.1

1. Functional Dependencies (FDs):

- StudentID → StudentName, StudentMajor
- ProjectID → ProjectTitle, ProjectType, SupervisorID
- SupervisorID → SupervisorName, SupervisorDept

- (StudentID, ProjectID) → Role, HoursWorked, StartDate, EndDate

2. Redundancy and anomalies:

- Redundancy: StudentName and StudentMajor repeated for each project of a student; SupervisorName and SupervisorDept repeated for each project supervised by same supervisor.
- Update anomaly example: Changing SupervisorName requires updating many rows.
- Insert anomaly example: To add a new supervisor with no project yet, ProjectID required (depending on PK design).
- Delete anomaly example: Deleting last project by a student may remove supervisor info if stored only in that row.

3. 1NF: No repeating groups apparent; ensure multi-valued attributes (if any) are moved to separate tables (e.g., multiple supervisors or roles). Assume table is in 1NF.
4. 2NF: Primary key = (StudentID, ProjectID) assuming each student may work on multiple projects. Partial dependencies: StudentID → StudentName, StudentMajor (depends only on StudentID) and ProjectID → ProjectTitle, ProjectType, SupervisorID (depends only on ProjectID).

2NF decomposition:

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

5. 3NF: Transitive dependency: SupervisorID → SupervisorName, SupervisorDept in Project table; to remove transitive dependency, create Supervisor/Professor table:

- Supervisor(SupervisorID PK, SupervisorName, SupervisorDept)

Final 3NF tables: Student, Supervisor, Project (with SupervisorID FK), StudentProject (associative).

Task 4.2

1. Primary key: (StudentID, CourseID, TimeSlot) or more precisely (StudentID, CourseSectionID) if a CourseSectionID exists. Reason: student can take multiple course sections; a course section is uniquely defined by CourseID+TimeSlot+Room (or a SectionID).

2. Functional dependencies:

- StudentID → StudentMajor
- CourseID → CourseName
- InstructorID → InstructorName
- Room → Building (rooms unique across campus)
- (CourseID, TimeSlot, Room) → InstructorID (each course section taught by one instructor at one time in one room)

3. BCNF check: The table is not in BCNF because StudentID → StudentMajor (non-key → attribute) and Room → Building are FDs violating BCNF if keys include StudentID+CourseID+TimeSlot.

4. BCNF decomposition:

- Student(StudentID PK, StudentMajor)
- Course(CourseID PK, CourseName)
- Instructor(InstructorID PK, InstructorName)
- Room(Room PK, Building)
- CourseSection(CourseSectionID PK, CourseID FK, InstructorID FK, TimeSlot, Room FK)
- Enrollment(StudentID FK, CourseSectionID FK, PRIMARY KEY(StudentID, CourseSectionID))

5. Decomposition is lossless because CourseSection references original course+time+room combination; Enrollment links students to sections. No information loss if FKs are maintained; ensure data migration preserves CourseSection identity.