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**

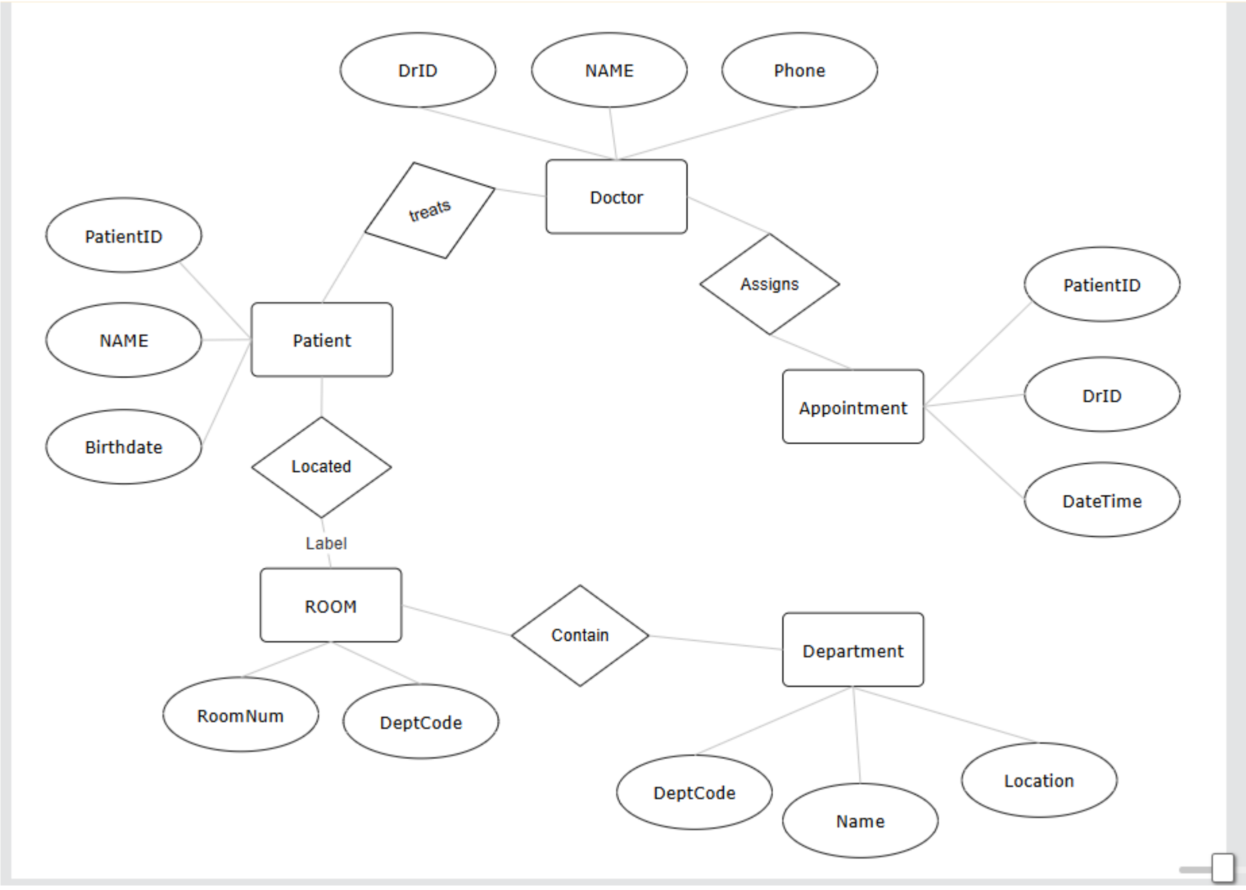
1)StudentID, CourseCode , Section , Semester & Year

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)*

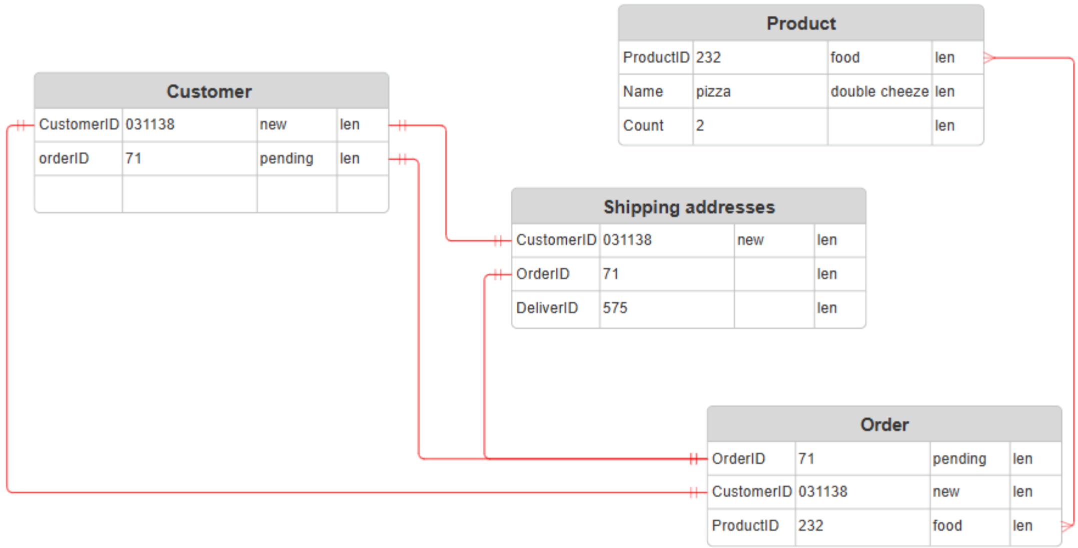
Doctor *prescribes* Prescription (1:N)

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.