Laboratory work #1

Part 1

Task 1.1

Relation A:

1)

1. {EmpID} - (unique for each employee)
2. {SSN} - (unique for each employee)
3. {Email} - (unique for each employee)
4. {EmpID, Phone}
5. {Email, Department}
6. {SSN, Name}

2)

1. {EmpID} — a system unique identifier.
2. {SSN} — unique
3. {Email} — unique

3)

EmpID

- It is system-generated and stable (does not change if the employee changes name, email, or phone).
- It does not contain sensitive information (unlike SSN).
- It is compact and efficient for indexing.

4)

In the sample data, phone numbers are different, but without a unique constraint, two employees *can* have the same phone number.

phone numbers can be repeated, but the EmpID still uniquely identifies employees.

Relation B:

1)

Primary key = **{StudentID, CourseCode, Section, Semester, Year}**

2)

StudentID → identifies the student.

CourseCode → identifies which course the student is taking.

Section → distinguishes different sections of the same course in the same semester.

Semester + Year → distinguish when the course was taken (students can take the same course in different semesters/years).

3)

If the system defines a unique identifier for course offerings (e.g., OfferingID = CourseCode + Section + Semester + Year), then another candidate key is **{StudentID, OfferingID}**. Otherwise, no other candidate keys exist.

Task 1.2

1) Student(AdvisorID) → Professor(ProfID) - The student has a supervisor
2) Course(DepartmentCode) → Department(DeptCode) - The course belongs to the department
3) Department(ChairID) → Professor(ProfID) - The department is headed by a professor
4) Enrollment(StudentID) → Student(StudentID) - The course entry indicates the student
5) Enrollment(CourseID) → Course(CourseID) - The course entry indicates the course

Part 2

Task 2.1

1)

Strong entities :

1) Patient
2) Doctor
3) Department
4) Room
5) Appointment
6) Prescription
7) Medication

Weak entities :

1) Phone
2) Specialization

2)

Attributes:

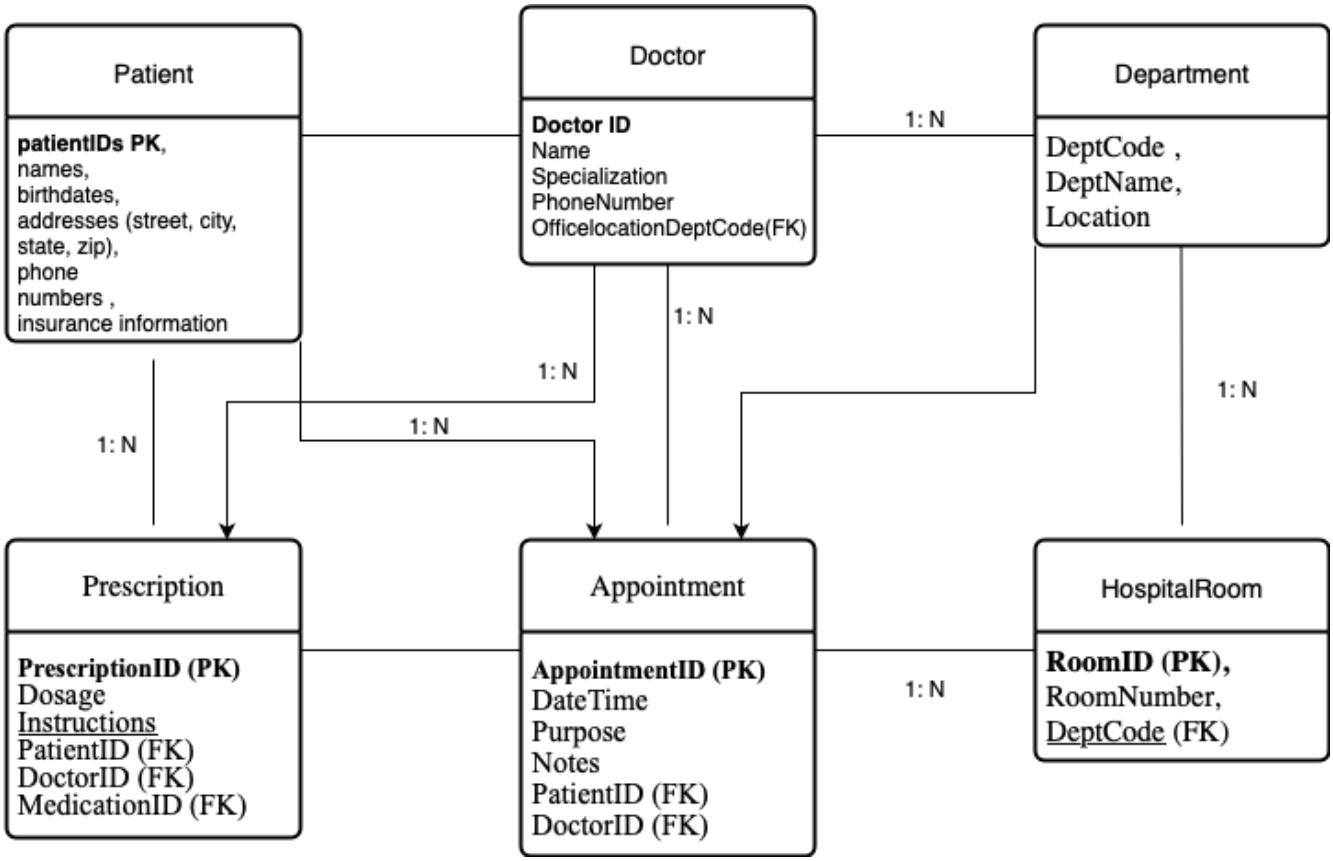
Patient:	**Doctor**:	**Appointment**:	**Prescription**:
PatientID (PK, simple)	DoctorID (PK)	AppointmentID (PK)	PrescriptionID (PK)
Name (simple)	Name (simple)	DateTime	Dosage
Birthdate (simple)	OfficeLocation (simple)	Purpose	Instructions
Address (composite: Street, City, State, Zip)	PhoneNumbers	Notes	PatientID (FK)
InsuranceInfo	Specializations	PatientID (FK)	DoctorID (FK)
PhoneNumbers		DoctorID (FK)	MedicationID (FK)

Department:	**Room**:	**Medication**:
DeptCode (PK)	RoomNumber (partial key, зависит от DeptCode)	MedicationID (PK)
DeptName		Name
Location	DeptCode (FK)	Description

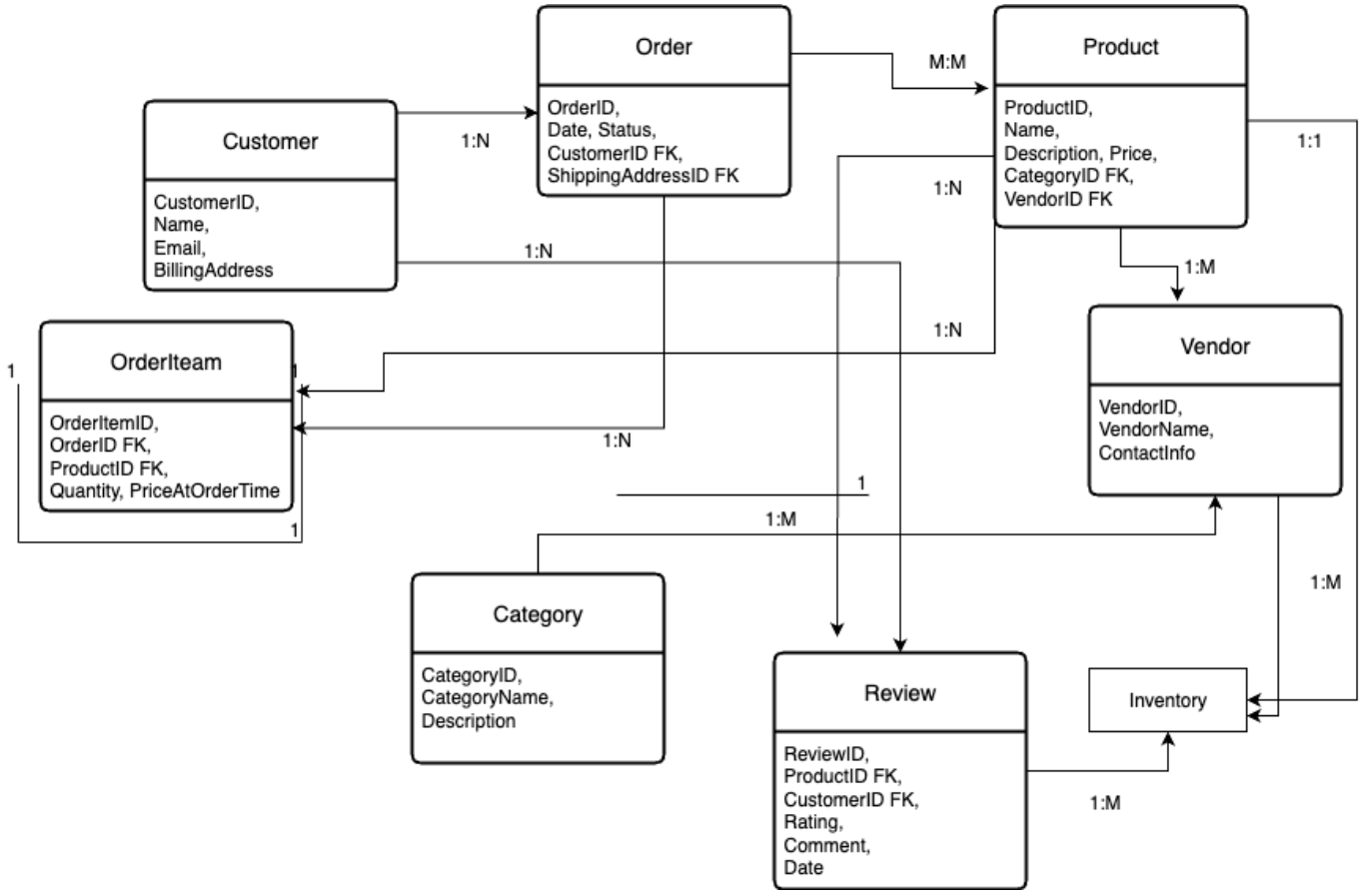
3) Relationships

- Patient — Appointment → **1:N** (A patient can have many appointments)
- Doctor — Appointment → **1:N** (The doctor has many appointments)
- Doctor — Department → **M:1** (A doctor belongs to one department, but there are many doctors in the department)
- Department — Room → **1:N** (There are many rooms in the department)
- Doctor — Prescription → **1:N** (A doctor can write many prescriptions)
- Patient — Prescription → **1:N** (A patient may have many prescriptions.)
- Prescription — Medication → **M:N** (one prescription can contain multiple medications, and one medication can occur in different prescriptions)

4)


Patient
patientIDs PK,
names,
birthdates,
addresses (street, city,
state, zip),
phone
numbers ,
insurance information
Doctor
Doctor ID
Name
Specialization
PhoneNumber
OfficelocationDeptCode(FK)
Department
DeptCode ,
DeptName,
Location
1: N
1: N
1: N
1: N
1: N
1: N
Prescription
PrescriptionID (PK)
Dosage
Instructions
PatientID (FK)
DoctorID (FK)
MedicationID (FK)
Appointment
AppointmentID (PK)
DateTime
Purpose
Notes
PatientID (FK)
DoctorID (FK)
HospitalRoom
RoomID (PK),
RoomNumber,
DeptCode (FK)

Task 2.2


Order
OrderID,
Date, Status,
CustomerID FK,
ShippingAddressID FK
M:M
Product
ProductID,
Name,
Description, Price,
CategoryID FK,
VendorID FK
Customer
CustomerID,
Name,
Email,
BillingAddress
1:N
1:N
1:N
1:1
1:M
1:N
1:N
OrderIteam
OrderItemID,
OrderID FK,
ProductID FK,
Quantity, PriceAtOrderTime
1
1
Vendor
VendorID,
VendorName,
ContactInfo
1
1:M
1:M
Category
CategoryID,
CategoryName,
Description
Review
ReviewID,
ProductID FK,
CustomerID FK,
Rating,
Comment,
Date
Inventory
1:M

Part 4

Task 4.1

1)

StudentID → StudentName, StudentMajor

ProjectID → ProjectTitle, ProjectType

SupervisorID → SupervisorName, SupervisorDept

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

2)

Redundancy exists because StudentName and StudentMajor repeat for the same student, ProjectTitle and ProjectType repeat for the same project, and SupervisorName and SupervisorDept repeat for the same supervisor.

Update anomaly: if a student changes major, it must be updated in every row where that student appears.
Insert anomaly: a new student cannot be added without a project.
Delete anomaly: if the last student in a project is removed, the project information is lost.

3)

Yes, the table breaks 1NF because some fields can repeat (like student info, supervisor info). To fix it, we keep only atomic values and split repeating groups into separate rows.

4)

Primary key = (StudentID, ProjectID).
Partial dependencies: StudentID → StudentName, StudentMajor; ProjectID → ProjectTitle, ProjectType, SupervisorID; SupervisorID → SupervisorName, SupervisorDept.
After 2NF:

- Student(StudentID, StudentName, StudentMajor)
- Project(ProjectID, ProjectTitle, ProjectType, SupervisorID)
- Supervisor(SupervisorID, SupervisorName, SupervisorDept)

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

5)

Transitive dependency: SupervisorID → SupervisorDept (through SupervisorName).
Final tables in 3NF are the same as in 2NF, because now each attribute depends only on the key.

Task 4.2

1) The primary key is (StudentID, CourseID, TimeSlot), because a student can take multiple courses, and each course section is uniquely identified by its course and time slot.

2)

StudentID → StudentMajor

CourseID → CourseName

InstructorID → InstructorName

Room → Building

(CourseID, TimeSlot) → InstructorID, Room

(StudentID, CourseID, TimeSlot) → Grade

3)

The table is not in BCNF. Some functional dependencies, such as StudentID → StudentMajor or CourseID → CourseName, have determinants that are not superkeys.

4)

The table is decomposed into:

- Student(StudentID, StudentMajor)

- Course(CourseID, CourseName)
- Instructor(InstructorID, InstructorName)
- Room(Room, Building)
- Section(CourseID, TimeSlot, InstructorID, Room)
- Enrollment(StudentID, CourseID, TimeSlot, Grade)

5)

There is no actual loss of information, but queries require more joins to reconstruct the original data.

Part 5

Task 5.1