Database Lab Work #1

Talgat Kozhakhmetov

Task 1.1

1.Superkeys: {EmpID}, {SSN}, {Email}, {SSN, Department}, {EmpID, Name}, {Email, Salary};

2.Candidate keys: EmpID, SSN, Email

3. EmpID, because It is stable (does not change), its uniqueness is controlled by the company, it is simple and convenient for database

4. No, they cant beacuse data shows only unique numbers, which implies that in this dataset, a phone number is used to identify a single employee

Relation B

1.StudentID, CourseCode, Semester, Year

2.StudentID and CourseCode are required to identify the student and course. Semester and Year are necessary to differentiate registrations for the same course in different semesters

3.StudentID, CourseCode, Section, Semester, Year. It’s unique but not minimal, since the Section attribute is redundant to ensure uniqueness

Task 1.2

1,Student table; AdvisorID is the ID of a professor who advises the student. It must be a valid ProfID that exists in the Professor table
2. Professor; Department where a professor works must be a valid department code that exists in the Department table
3. Course; Department that offers a course must be a valid department code that exists in the Department table
4. Department; ChairID is the ID of a professor who chairs the department. It must be a valid ProfID that exists in the Professor table
5. Enrollment; StudentID and CourseID; StudentID → StudentID; CourseID → CourseID

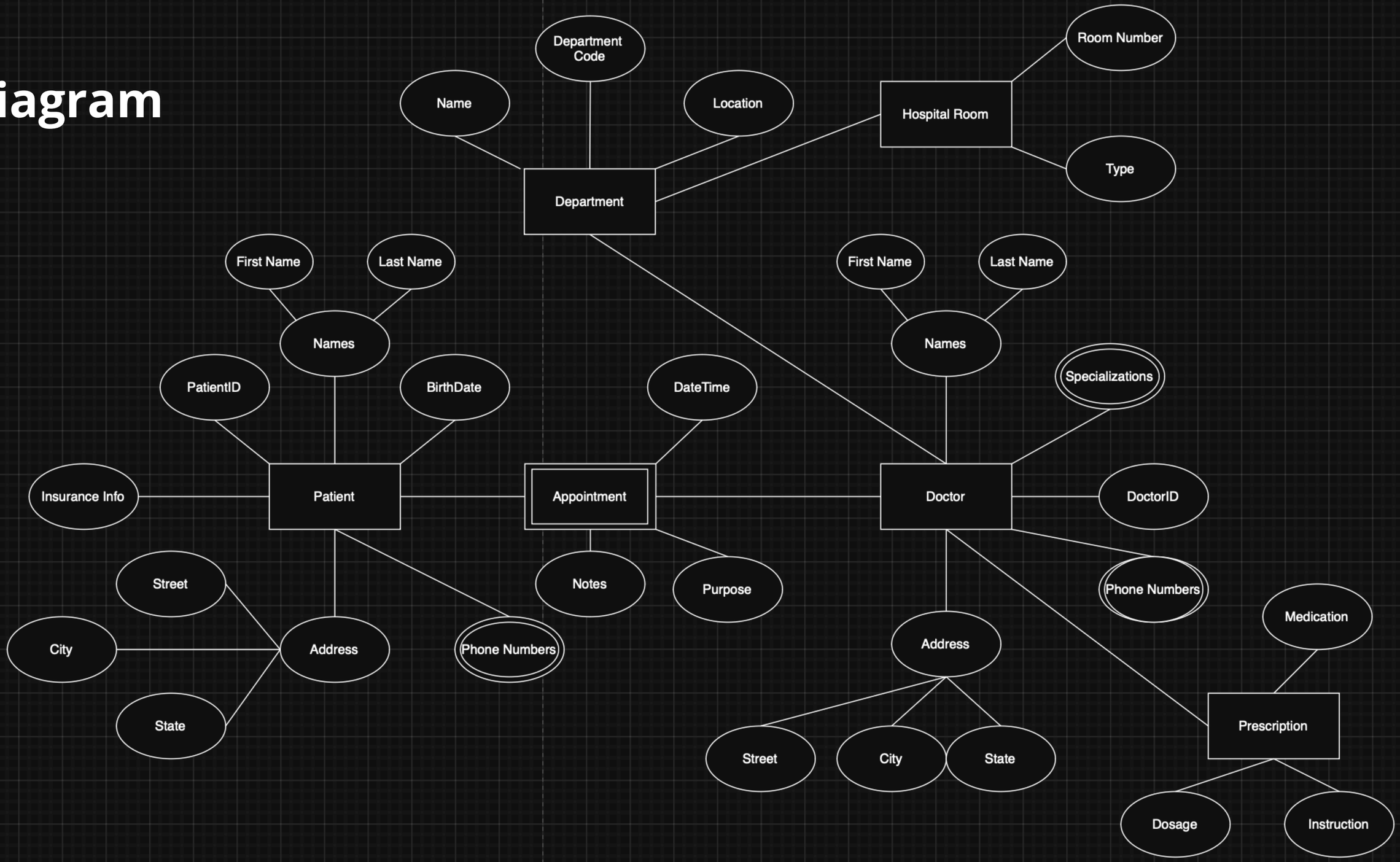
Task 2.1

1.Strong entities: Patient, Doctor, Department, Medication, Hospital Room
Weak entities: Appointment, Prescription

2. Patients: PatientID (simple), names (composite), birthdate(simple), address(composite),phone numbers(multi-valued), insurance info (simple)
Doctor: DoctorID (Simple), names (composite), specializations (multi-valued), phone numbers (multi-valued), office locations (simple)
Department: department code (simple), names(simple), location(simple)
Appointments: DateTime (simple),purpose (simple),notes (simple)
Prescription: medication, dosage and instructions (simple)
Hospital rooms: Room Number (simple), Room Category (simple)

3. Patient makes Appointments: 1:N;
Doctor has Appointments: 1:N;
Doctor in Department N:1;
Hospital Room in Department N:1;
Doctor write Prescription 1:N;
Doctor has Specialization M:N

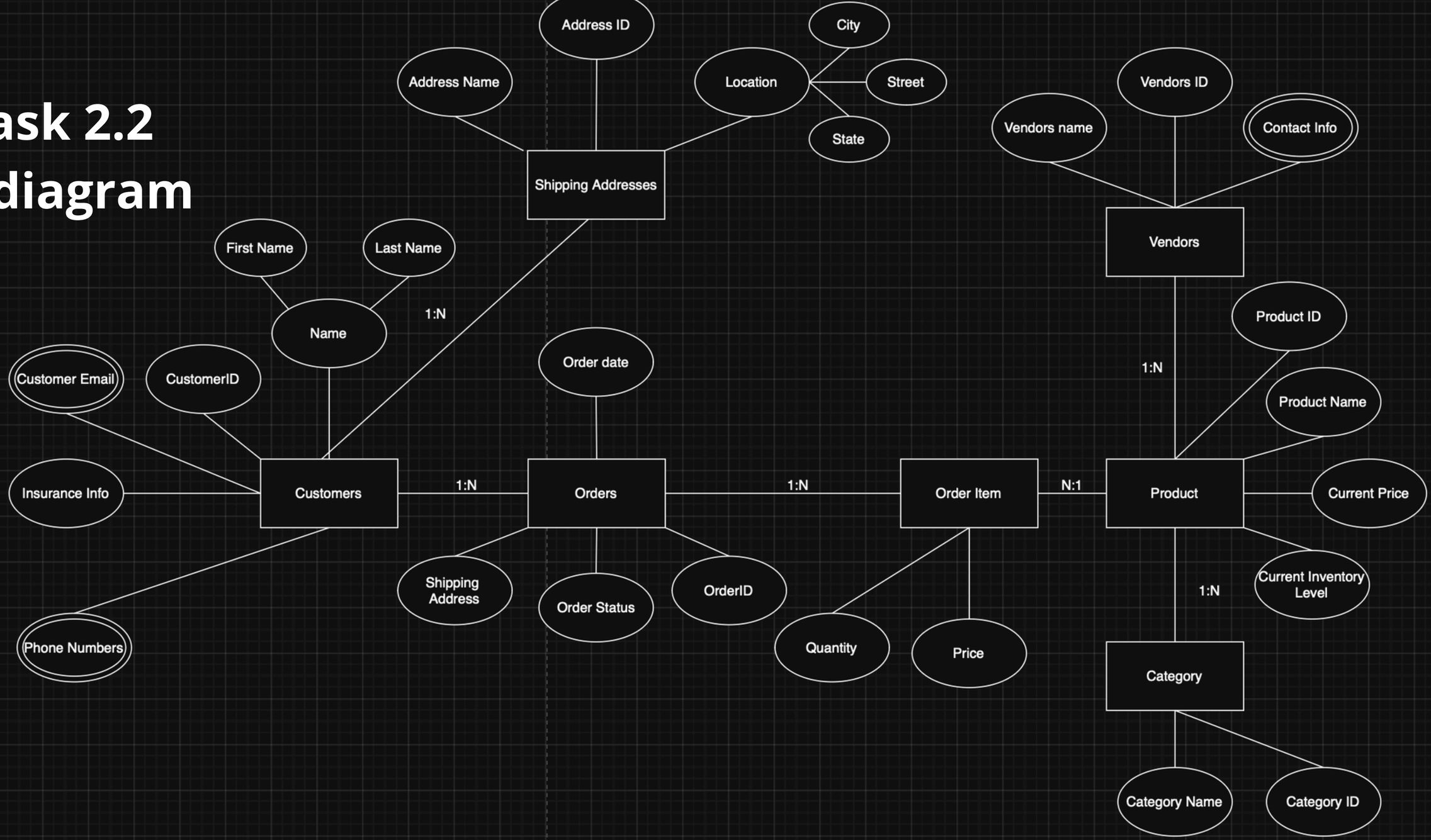
ER diagram


Department Code
Name
Location
Room Number
Hospital Room
Type
Department
First Name
Last Name
Names
PatientID
BirthDate
First Name
Last Name
Names
Specializations
DateTime
Insurance Info
Patient
Appointment
Doctor
DoctorID
Notes
Purpose
Street
City
State
Address
Phone Numbers
Phone Numbers
Medication
Address
Prescription
Street
City
State
Dosage
Instruction

Task 2.1

5. Primary keys: Patient: patient_id;
Doctor: doctor_id; Department: dept_code;
Medication: medication_id;
Appointment: appointment_id;
Prescription: prescription_id;
HospitalRoom: (room_number, dept_code);
DoctorSpecialization: (doctor_id, specialization_name);
PatientPhone: (patient_id, phone_number)

Task 2.2
ER diagram



Task 2.2

Weak Entity: OrderItem
Justification: An OrderItem cannot exist independently without an Order. The line items on an invoice (quantity, price paid) have no meaning outside the context of the specific order they belong to. Their existence is existence-dependent on the Order entity. Furthermore, its primary key is partially formed by the primary key of the Order entity (order_id), which is the definition of a weak entity.

Relationship: The relationship between Customer and Product for Reviews and Ratings.
Why it's M:N: One Customer can review many Products. One Product can be reviewed by many Customers.
The Need for Attributes: This relationship is not just a simple link; it has its own attributes: rating, review_text, and review_date. These attributes describe the relationship itself, not just the customer or the product.

Task 4.1

1.StudentID → Student Name, Student Major;
ProjectID → ProjectTitle, ProjectType, SupervisorID
SupervisorID → SupervisorName, SupervisorDept
{StudentID, ProjectID} → Role, HoursWorked, StartDate, EndDate;

2.Redundancy: Student, project, and supervisor data are duplicated for each Student Project relationship.
Update anomaly: Changing the lead student or department manager requires editing all rows.
Add anomaly: It is possible to add a project without a student or a student without a project.
Delete anomaly: Deleting the last student in a project results in the loss of project data.

3.The Role attribute is not atomic. A student can have multiple roles on a project, which is a repeating group.
We should create a new table StudentProjectRoles with a composite key (StudentID, ProjectID, Role). Move the multi-valued Role data there. The original StudentProject table keeps a single Role per student-project or is eliminated.

4.PK:{StudentID, ProjectID}
StudentID → Student Name, Student Major;
ProjectID → ProjectTitle, ProjectType, SupervisorID, SupervisorName, SupervisorDept;
2NF:
Decompose into three tables:

1. **Students(StudentID, StudentName, StudentMajor)**
2. **Projects(ProjectID, ProjectTitle, ProjectType, SupervisorID, SupervisorName, SupervisorDept)**
3. **Work(StudentID, ProjectID, Role, HoursWorked, StartDate, EndDate)**

Task 4.1

5.SupervisorID → SupervisorName, SupervisorDept

3NF:

1. **Students(StudentID (PK), StudentName, StudentMajor)**
2. **Supervisors(SupervisorID (PK), SupervisorName, SupervisorDept)**
3. **Projects(ProjectID (PK), ProjectTitle, ProjectType, SupervisorID` (FK to Supervisors))**
4. **Work(StudentID (FK to Students), ProjectID(FK to Projects), Role, HoursWorked, StartDate, EndDate)**
   PK: {StudentID, ProjectID}

Task 4.2

1.PK: {StudentID, CourseID, TimeSlot} or {StudentID, CourseID, Room}. A student can be in multiple courses, a course has multiple students, and a room/time is unique per course section;

2.

- **StudentID → Student Major**
- **CourseID → CourseName**
- **InstructorID → InstructorName**
- **{TimeSlot, Room} → Building**
- **{CourseID, TimeSlot} → InstructorID, Room**
- **{CourseID, Room} → InstructorID, TimeSlot**

3. No, it is not in BCNF. The determinants like StudentID, CourseID, InstructorID, {TimeSlot, Room} are not superkeys in this table.

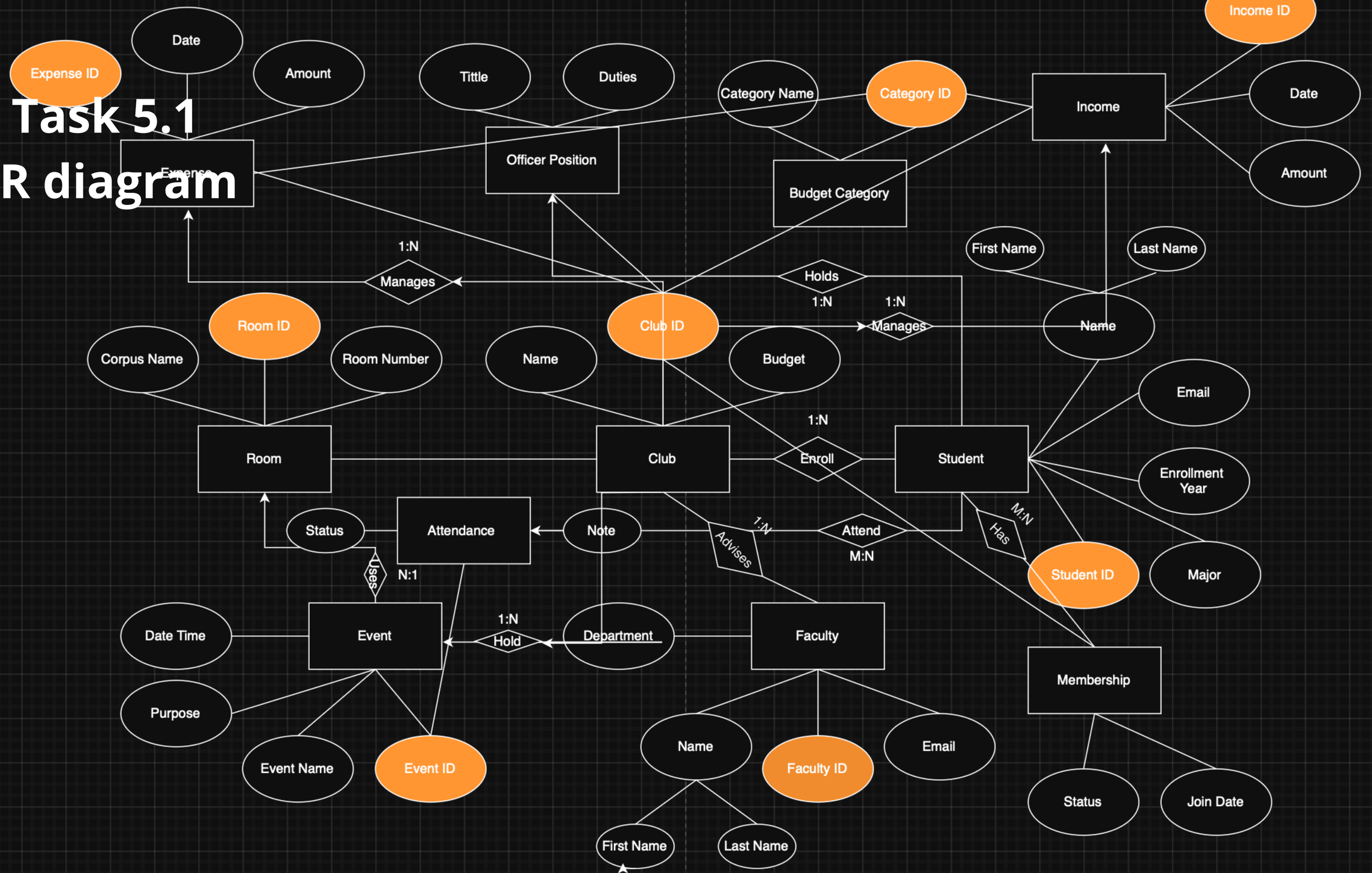
Task 4.2

BCNF Decomposition:

1. **Students(StudentID (PK), StudentMajor)**
2. **Courses(CourseID (PK), CourseName)**
3. **Instructors(InstructorID (PK), InstructorName)**
4. **Rooms(Room (PK), Building)**
5. **Sections(CourseID (FK to Courses), TimeSlot, InstructorID(FK to Instructors), Room (FK to Rooms))**
   - **PK: {CourseID, TimeSlot} (or {CourseID, Room} based on business rule)**
6. **Enrollment(StudentID (FK to Students), CourseID(FK to Courses), TimeSlot)**
   - **PK: {StudentID, CourseID, TimeSlot}**

5. No, the decomposition is lossless. All original information can be recovered by joining the tables.

Task 5.1
ER diagram



Task 5.1

1. **Student (student_id PK, first_name, last_name, email, major, enrollment_year)**
2. **Faculty (faculty_id PK, first_name, last_name, email, department)**
3. **Club (club_id PK, club_name, description, founding_date, budget, advisor_id FK to Faculty.faculty_id)**
4. **Room (room_id PK, building_name, room_number, capacity)**
5. **BudgetCategory (category_id PK, category_name)**
6. **Event (event_id PK, event_name, description, date_time, purpose, club_id FK to Club.club_id,room_id FK to Room.room_id)**
7. **Membership (student_id PK/FK to Student.student_id, club_id PK/FK to Club.club_id, join_date, membership_status)**
8. **OfficerPosition (position_id PK, title, duties, start_date, end_date, club_id FK to Club.club_id,student_id FK to Student.student_id)**
9. **Expense (expense_id PK, amount, description, date, club_id FK to Club.club_id, category_idFK to BudgetCategory.category_id)**
10. **Income (income_id PK, amount, description, date, club_id FK to Club.club_id, category_id FK to BudgetCategory.category_id)**
11. **Attendance (student_id PK/FK to Student.student_id, event_id PK/FK to Event.event_id, attended, notes)**

3.
Design Decision: Choice of Primary Key for OfficerPosition
Option 1: Composite natural key (club_id, title). Valid because a club can't have two presidents.
Option 2: Surrogate key (position_id). Simplifies relations and is more stable.
My Choice: Option 2 (position_id).
Reasoning: A surrogate key is more robust. It remains unchanged if the job title (title) is ever updated, preventing unnecessary complexity and potential future foreign key management issues.

4. Example Queries (English)
Membership Roster: "Retrieve a list of all active members of the 'Data Science Club', showing their full names, email addresses, and their join dates."
Event Attendance Tracking: "List all students who attended the event 'Annual Hackathon 2024', along with the name of their major."
Budget Overview: "Show the total expenses and total income for the 'Photography Club' in the year 2024, broken down by budget category."