Part 1 — Key Identification

Part 2 — ER Diagrams

Part 4 — Normalization

Part 5 — Design Challenge

References / Tools used

*Part 1 (Key Identification):*

*Relation A: Employee  
Attributes: Employee(EmpID, SSN, Email, Phone, Name, Department, Salary)*

*1. Examples of Superkeys:*

*{EmpID}*

*{SSN}*

*{Email}*

*{EmpID, Name}*

*{SSN, Phone}*

*{Email, Phone}*

*2. Candidate Keys:*

*{EmpID}*

*{SSN}*

*{Email}*

*3. Primary Key Choice:*

*Choose EmpID as the primary key.*

*Reason: EmpID is short, system-generated, stable, and does not expose personal information.*

*4. Phone Uniqueness:*

*Two employees may share the same phone number (e.g., family or shared office line).*

*Therefore, Phone should not be treated as a unique identifier.*

*Relation B: Registration  
Attributes: Registration(StudentID, CourseCode, Section, Semester, Year, Grade, Credits)*

*1. Minimum Primary Key:*

*(StudentID, CourseCode, Section, Semester, Year)*

*2. Why each attribute is necessary:*

*StudentID - identifies the student*

*CourseCode - identifies the course*

*Section - distinguishes different sections of the same course*

*Semester, Year - distinguish the same course taken in different terms*

*3. Additional Candidate Keys:*

*A surrogate key RegistrationID could be introduced as an alternative.*

*Without it, the only candidate key is (StudentID, CourseCode, Section, Semester, Year).*

*Foreign Key Relationships:*

*Student.AdvisorID - Professor.ProfID*

*Department.ChairID - Professor.ProfID*

*Course.DepartmentCode - Department.DeptCode*

*Enrollment.StudentID - Student.StudentID*

*Enrollment.CourseID - Course.CourseID*

*Deletion/Update Behavior Recommendations:*

*If a Professor is deleted - set AdvisorID = NULL in Student*

*If a Department is deleted - restrict deletion if Courses exist*

*If a Course is deleted - cascade delete all related Enrollment records*

*Part 2.1  
Entities and Attributes:*

*Patient (PK: PatientID, Name, Birthdate, Address [Street, City, State, Zip], Insurance)*

*Multi-valued: Phone*

*Doctor (PK: DoctorID, Name, OfficeLocation)*

*Multi-valued: Specialization, Phone*

*Department (PK: DeptCode, DeptName, Location)*

*Room (PK: DeptCode + RoomNumber)*

*Appointment (PK: AppointmentID, DateTime, Purpose, Notes)*

*Prescription (PK: PrescriptionID, Dosage, Instructions)*

*Medication (PK: MedicationID, Name, Description)*

*Relationships and Cardinalities:*

*Patient (1) — (N) Appointment — (1) Doctor*

*Appointment (1) — (N) Prescription — (1) Doctor*

*Prescription (N) — (1) Medication*

*Department (1) — (N) Room*

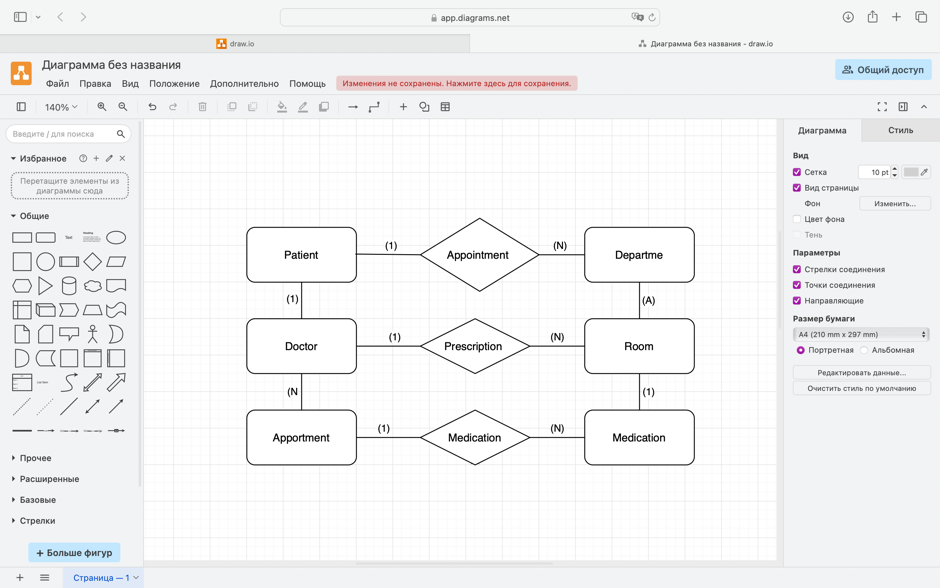
*Doctor (N) — (1) Department*

*Patient (N) — (M) Medication (through Prescription)*

*Keys:*

*Primary keys are underlined.*

*Foreign keys will be added in relational schema after ER design.*

**

*Part2.2*

*Entities and Attributes:*

*Customer (PK: CustomerID, Name, Email, Address [Street, City, State, Zip])*

*Multi-valued: Phone*

*Seller (PK: SellerID, Name, ContactInfo, BusinessName)*

*Product (PK: ProductID, Name, Description, Price, StockQuantity)*

*Order (PK: OrderID, OrderDate, Status, TotalAmount)*

*Payment (PK: PaymentID, Method, Amount, Date)*

*Review (PK: ReviewID, Rating, Comment, Date)*

*Relationships and Cardinalities:*

*Customer (1) — (N) Order*

*Order (N) — (M) Product (through OrderLine)*

*Order (1) — (1) Payment*

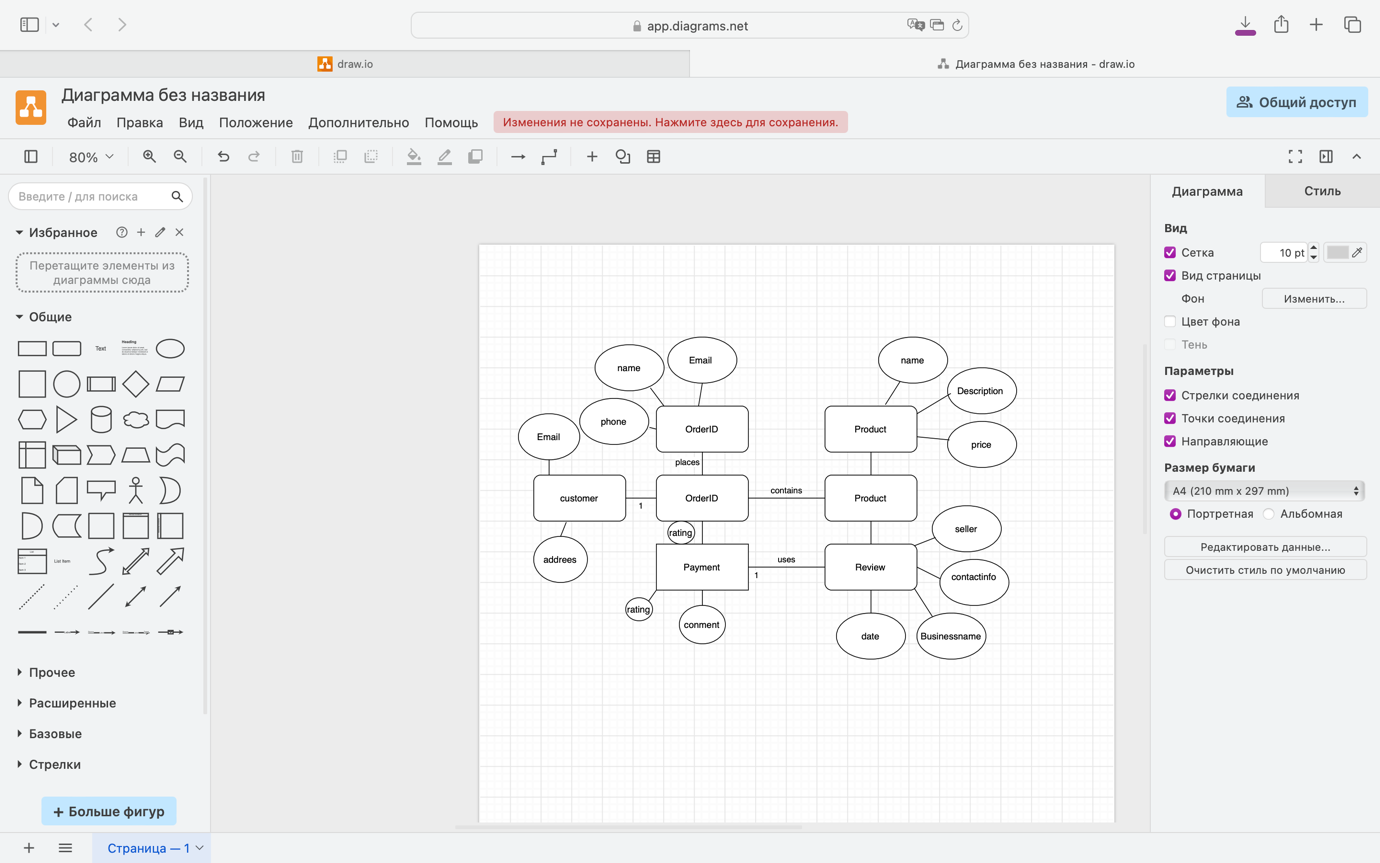
*Customer (1) — (N) Review — (1) Product*

*Seller (1) — (N) Product*

*Keys:*

*Primary keys are underlined.*

*Foreign keys will be added in relational schema after ER design.*

**

*Part 4.1*

*Step 1: Functional Dependencies (FDs):*

*StudentID - StudentName, StudentMajor*

*ProjectID - ProjectTitle, ProjectType*

*SupervisorID - SupervisorName, SupervisorDept*

*{StudentID, ProjectID} - Role, HoursWorked, StartDate, EndDate*

*Step 2: Problems (Redundancy and Anomalies):*

*Redundancy: StudentName repeated for every project, SupervisorName repeated for each supervised project.*

*Update anomaly: If a supervisor changes department, all rows must be updated.*

*Insert anomaly: Cannot insert a new student without assigning a project.*

*Delete anomaly: If last project of a student is deleted, we lose student information.*

*Step 3: 1NF:*

*Already atomic (no repeating groups). Table is in 1NF.*

*Step 4: 2NF:*

*Primary key: {StudentID, ProjectID}*

*Partial dependencies: StudentID - StudentName, StudentMajor; ProjectID - ProjectTitle, ProjectType*

*Decomposition to 2NF:*

*Student(StudentID, StudentName, StudentMajor)*

*Project(ProjectID, ProjectTitle, ProjectType)*

*Supervisor(SupervisorID, SupervisorName, SupervisorDept)*

*StudentProject(StudentID, ProjectID, SupervisorID, Role, HoursWorked, StartDate, EndDate)*

*Step 5: 3NF:*

*Transitive dependency: SupervisorID - SupervisorDept (already isolated in Supervisor table)*

*Final 3NF tables:*

*Student(StudentID, StudentName, StudentMajor)*

*Project(ProjectID, ProjectTitle, ProjectType)*

*Supervisor(SupervisorID, SupervisorName, SupervisorDept)*

*StudentProject(StudentID, ProjectID, SupervisorID, Role, HoursWorked, StartDate, EndDate)*

*Part 4.2*

*Step 1: Identify Repetition / Problems*

*Multiple products (Product1, Product2, Product3) stored in one row → violates 1NF.*

*Customer info (Name, Address) repeated for every order → redundancy.*

*Salesperson info (Name, Phone) repeated for every order → redundancy.*

*Step 2: First Normal Form (1NF)*

*Remove repeating groups by creating a separate table for order details.*

*Customer(CustomerID PK, CustomerName, CustomerAddress)  
Salesperson(SalespersonID PK, SalespersonName, SalespersonPhone)  
Order(OrderID PK, CustomerID FK, SalespersonID FK, TotalAmount)  
OrderDetail(OrderID FK, ProductName, PRIMARY KEY(OrderID, ProductName))*

*Step 3: Second Normal Form (2NF)*

*Ensure no partial dependencies.*

*Since OrderDetail uses composite key (OrderID, ProductName), all attributes depend on the full key. Already in 2NF.*

*Step 4: Third Normal Form (3NF)*

*Remove transitive dependencies.*

*No attribute depends on non-key attributes.*

*Final schema remains same.*

*Final Normalized Tables (3NF):*

*Customer(CustomerID PK, CustomerName, CustomerAddress)  
Salesperson(SalespersonID PK, SalespersonName, SalespersonPhone)  
Order(OrderID PK, CustomerID FK, SalespersonID FK, TotalAmount)  
OrderDetail(OrderID FK, ProductName, PRIMARY KEY(OrderID, ProductName))*

*Part 5.1*

*Step 1: Entities and Attributes*

*Student*

*StudentID (PK)*

*Name*

*Email*

*Major*

*Club*

*ClubID (PK)*

*ClubName*

*Budget*

*FacultyAdvisor*

*FacultyID (PK)*

*Name*

*Department*

*Membership (associative entity for Student–Club many-to-many)*

*StudentID (FK - Student.StudentID)*

*ClubID (FK - Club.ClubID)*

*Role (officer position: president, treasurer, etc.)*

*PRIMARY KEY (StudentID, ClubID)*

*Event*

*EventID (PK)*

*ClubID (FK - Club.ClubID)*

*EventName*

*EventDate*

*Attendance (associative entity for Student–Event many-to-many)*

*StudentID (FK -Student.StudentID)*

*EventID (FK -Event.EventID)*

*PRIMARY KEY (StudentID, EventID)*

*Room*

*RoomID (PK)*

*Location*

*Reservation*

*ReservationID (PK)*

*EventID (FK - Event.EventID)*

*RoomID (FK - Room.RoomID)*

*DateTime*

*Expense*

*ExpenseID (PK)*

*ClubID (FK - Club.ClubID)*

*Amount*

*Description*

*Date*

*Step 2: Relationships and Cardinalities*

*Student–Club: M:N (resolved by Membership)*

*Club–Advisor: 1:N (one faculty can advise many clubs, one club has one advisor)*

*Club–Event: 1:N (each club organizes many events)*

*Student–Event: M:N (resolved by Attendance)*

*Event–Room: M:N (resolved by Reservation)*

*Club–Expense: 1:N (each club can have many expenses)*

*Step 3: Normalized Relational Schema*

*Student(StudentID PK, Name, Email, Major)  
Club(ClubID PK, ClubName, Budget, FacultyID FK -FacultyAdvisor.FacultyID)  
FacultyAdvisor(FacultyID PK, Name, Department)  
Membership(StudentID FK -Student.StudentID, ClubID FK -Club.ClubID, Role, PRIMARY KEY(StudentID, ClubID))  
Event(EventID PK, ClubID FK - Club.ClubID, EventName, EventDate)  
Attendance(StudentID FK - Student.StudentID, EventID FK - Event.EventID, PRIMARY KEY(StudentID, EventID))  
Room(RoomID PK, Location)  
Reservation(ReservationID PK, EventID FK - Event.EventID, RoomID FK -Room.RoomID, DateTime)  
Expense(ExpenseID PK, ClubID FK - Club.ClubID, Amount, Description, Date)*

*Step 4: Design Decision Example*

*For Room reservations, I could have stored RoomID directly in the Event table.*

*Instead, I created a separate Reservation entity because one event may require multiple rooms or multiple reservations at different times.*

*This design provides more flexibility.*

*Step 5: Example Queries (in English, not SQL)*

1. *“Find all students who are officers in the Computer Science Club.”*
2. *“List all events scheduled for next week with their room reservations.”*
3. *“Show the total expenses for each club this semester.”*

