# Part 1. Key Identification

## 1.1 Employee. Relation A: Employee

**supercase:**

{EmpID}, {SSN}, {Email}, {EmpID, Phone}, {SSN, Email}, {FirstName, LastName, SSN}

**Candidate keys:**

{EmpID}, {SSN}, {Email}

**Primary keys:**

EmpID (unique)

What if two employees have the same phone number?

This is acceptable because Phone is not a key.

## Relation B: Course Registration

**1) Minimum attributes for a primary key:**

- Option A: RegID (created) — minimal and convenient.
- Option B: (StudentID, CourseID, Section, Semester, Year)

**2) Why is each attribute necessary (for option B):**

- StudentID — identify student; without it, the registration is not related to the student
- CourseID — identify course.

**3) any additional candidate keys(if their exist):**

- If there is RegID — it is candidate key
- Otherwise, the candidate key is option B

### 1.2 Foreign Keys

**foreign keys:**

- Enrollment.StudentID → Student(StudentID)
- Enrollment.CourseID → Course(CourseID)
- Course.ProfessorID → Professor(ProfessorID)
- Course.DeptID → Department(DeptID)
- Professor.DeptID → Department(DeptID) (if)
- Student.AdvisorID → Professor(ProfessorID)

# Part 2. ER Diagram Construction

**2.1 Hospital System**

**entities (strong/weak):**

1. Patient — strong
2. Doctor — strong
3. Appointment — strong
4. Prescription — weak
5. Medicine — strong

**2) Identify all attributes for each entity**

- Patient
    - PatientID (PK) — simple
    - Name — simple (FirstName, LastName)
    - Address — composite (Street, City, PostalCode)
    - Phone — multi-valued (maybe many phones)
- Doctor
    - DoctorID (PK) — simple
    - Name — simple
    - Specialty — simplr
    - Phone — simple or multi-valued (if work's phone)
- Appointment
    - AppID (PK) — simple
    - DateTime — simple
    - Reason/Diagnosis — simple
    - Status — simple
- Prescription
    - (PresID) — PK
    - Dosage — simple

    - Instructions — simple
- Medicine
    - MedID (PK) — simple
    - MedName — simple
    - Manufacturer — simple

**3) connections**

- Patient (1) — (M) Appointment — one patient can have many appointments. (1:M)
- Doctor (1) — (M) Appointment — one doctor can have many patients. (1:M)
- Appointment (1) — (M) Prescription — several prescriptions can be written for one appointment. (1:M)
- Prescription (M) — (N) Medicine — one prescription can contain several medications, one medication can appear in different prescriptions. (M:N)

**2.2 E-commerce System**

**attributes:**

- Customer(CustID, Name, Email)
- Order(OrderID, Date)
- Product(ProdID, Name, Price)
- Vendor(VendorID, Name)
- OrderItem(OrderID, ProdID, Quantity, PriceAtPurchase) (weak entity)

**connections:**

- Customer — Order (1:M)
- Order — Product (M:N  in OrderItem)
- Product — Vendor (M:N in Supply)

**Weak entity & justification**

**Weak entity:** OrderItem

Рассматриваю как слабую, потому что сама по себе не имеет смысла без (Order).
 PK для OrderItem часто составной: (OrderID, ProdID)
**Почему слабая:** OrderItem не существует независимо от Order, и её идентичность определяется (Order).

**M:N relationship that needs attributes**

Order ↔ Product (M:N) реализуется через OrderItem.

# Part 4. Normalization

**4.1 StudentProject**

**relation:** (StudentID, StudentName, Major, ProjectID, ProjectTitle, SupervisorID, SupervisorName)

**functional dependencies:**

- StudentID → StudentName, Major
- ProjectID → ProjectTitle, SupervisorID
- SupervisorID → SupervisorName

**Problems**

> Redundancy:  The student's (name, major) data is repeated for each project he/she participates in. The supervisor's (name) data is repeated for each project he/she supervises.

**1NF**

1NF требует атомарности атрибутов. В нашей таблице все атомарны (StudentName — строка, Major — строка, ProjectTitle — строка.

**2NF**

**Определение первичного ключа исходной таблицы:**
Если одна строка описывает участие одного студента в одном проекте, то PK = (StudentID, ProjectID)

- StudentName, Major зависят от StudentID — частичная зависимость
- ProjectTitle, SupervisorID зависят от ProjectID — частичная зависимость

разделим таблицу, устраняя частичные зависимости:

- Student(StudentID PK, StudentName, Major)
- Project(ProjectID PK, ProjectTitle, SupervisorID FK)

**3NF**

SupervisorName зависит от SupervisorID, а SupervisorID присутствует в Project. То есть в исходной таблице SupervisorName — зависит от ProjectID через SupervisorID.

- Student(StudentID PK, StudentName, Major)
- Supervisor(SupervisorID PK, SupervisorName)
- Project(ProjectID PK, ProjectTitle, SupervisorID FK)

**4.2 CourseSchedule**

**relation:** (StudentID, StudentMajor, CourseID, CourseName, TimeSlot, Room, Capacity)

**Primary key:**

- StudentID identify student (StudentMajor depends).
- CourseID identify course (CourseName depends).

**functional dependencies:**

- StudentID → StudentMajor
- CourseID → CourseName
- (CourseID, TimeSlot) → Room

**BCNF(created other tables):**

1. Student(StudentID PK, StudentMajor)
2. Course(CourseID PK, CourseName)
3. Room(RoomID PK, Capacity)

# Part 5. Clubs System

**requirements:**

- Student(StudentID, Name, Major)
- Club(ClubID, Name)
- Membership(StudentID, ClubID, Role) ← связь M:N
- Event(EventID, ClubID, Date, Title)

**relations:**

- Student ↔ Club (M:N in Membership)

- Club → Event (1:M)

Examples:

1) “Find all students who are officers in the Computer Science Club.”
(Club.Name = 'Computer Science'  Membership.Role IN ('officer','president','vice-president'))
2) “List all events scheduled for next week with their room reservations.”
(SELECT Event.Title, Event.Date, Room.RoomName, Room.Capacity BETWEEN 7 AND 14)
3) “Find students who are members of more than 3 clubs.”
(SELECT StudentID, COUNT() FROM Membership GROUP BY StudentID HAVING COUNT() > 3)