**Surname, First Name**

**Member 1:Dullon, Kurt Jansen**

**Member 2:Montes, John Michael**

**SCENARIO: MANAGING PATIENTS, DOCTORS, AND APPOINTMENTS**

A hospital needs a **database system** to manage **patients, doctors, appointments, and medical records**. The goal is to efficiently store and retrieve **patient details, doctor schedules, and appointment history**.

**Step 1: Identify Entities (Core Objects in the System)**

Entities represent the **main objects** in the system that need to store data.

**Entities in our Hospital Management System:**

1. **Patients –** Stores patient details such as name, age, gender, contact information, and medical history.
2. **Doctors –** Contains information about doctors, their specialization, contact details, and availability.
3. **Appointments –** Manages appointment scheduling between patients and doctors, including date, time, and status.
4. **Medical Records –** Stores patient medical history, diagnoses, prescriptions, and treatment plans.

**Step 2: Define Attributes (Characteristics of Each Entity)**

Each entity consists of **attributes** that store relevant data.

* **Patient** (PatientID (PK), Name, DateOfBirth, Gender, Contact, Address)
* **Doctor** (DoctorID (PK), Name, Specialization, Contact, Availability)
* **Appointment** (AppointmentID (PK), PatientID (FK), DoctorID (FK), AppointmentDate, Status)
* **Medical\_Record** (RecordID (PK), PatientID (FK), DoctorID (FK), Diagnosis, Prescription, RecordDate)

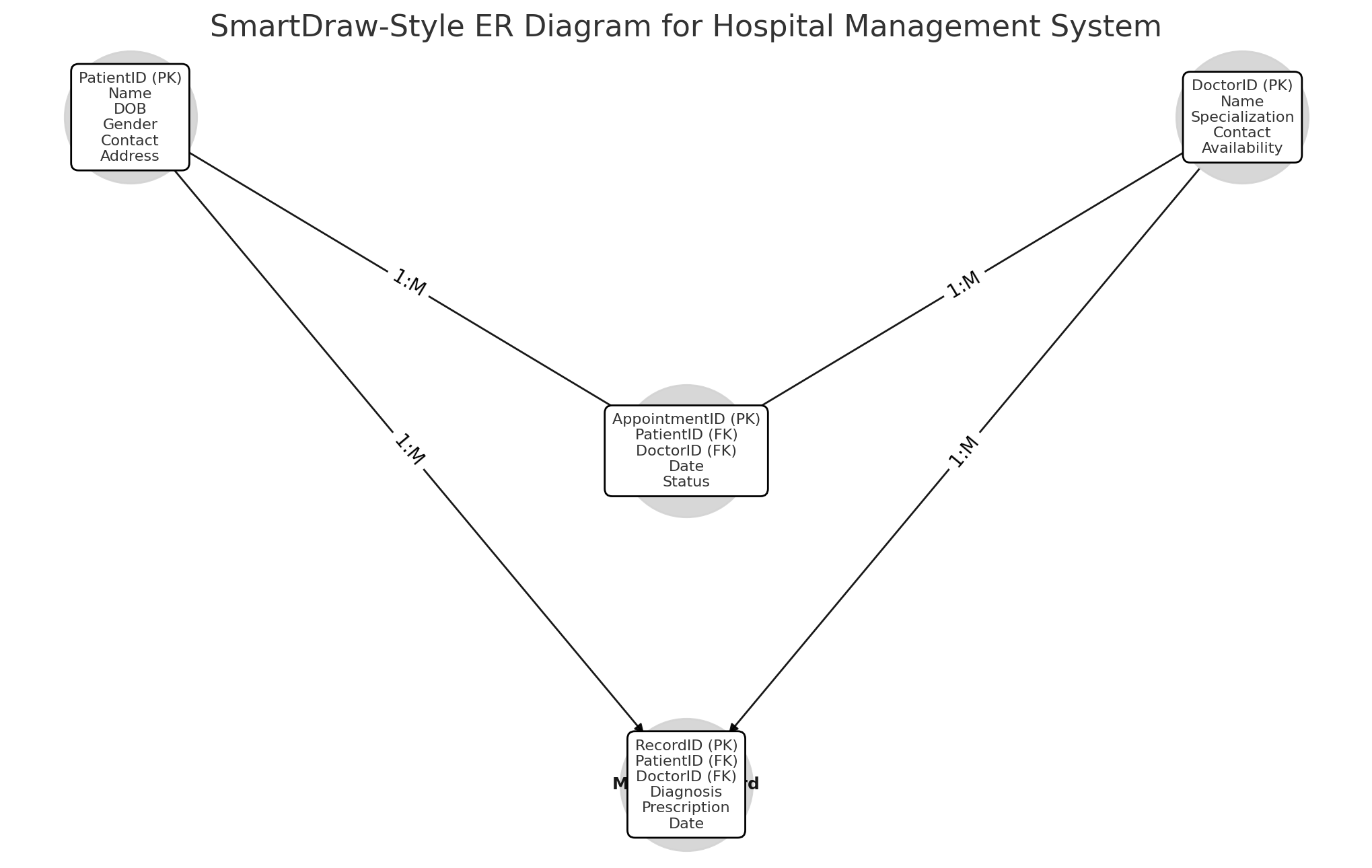
**Primary Key (PK)** → Unique identifier for each record.  
**Foreign Key (FK)** → Creates relationships between tables.

**Step 3: Establish Relationships**

Now, we define **how these entities relate** to one another.

* **Patient & Appointment → One-to-Many (1:M) → One patient can have multiple appointments, but each appointment belongs to only one patient.**
* **Doctor & Appointment → One-to-Many (1:M) → One doctor can have multiple appointments, but each appointment is assigned to only one doctor.**
* **Patient & Medical\_Record → One-to-Many (1:M) → One patient can have multiple medical records, but each medical record belongs to only one patient.**
* **Doctor & Medical\_Record → One-to-Many (1:M) → One doctor can create multiple medical records, but each medical record is associated with only one doctor.**

**Step 4: Draw the ERD (Entity-Relationship Diagram)**

****

**(Not final)**

**Step 5: Normalize the Data Model**

To ensure **data consistency**, we apply **database normalization** principles:

**1st Normal Form (1NF):** No duplicate columns or repeating groups.  
**2nd Normal Form (2NF):** Every non-key column depends on the whole primary key.  
**3rd Normal Form (3NF):** No transitive dependencies (no unnecessary dependencies on non-key attributes).

This ensures **data integrity** and avoids redundancy.

**Step 6: Implement the Database Schema (SQL Code)**

Once the data model is finalized, we can create the **database tables** using **SQL**:

CREATE TABLE

CREATE TABLE

CREATE TABLE

CREATE TABLE

**Step 7: Validate & Optimize the Model**

* **Indexing:** Create indexes on PatientID, DoctorID, and AppointmentID for fast retrieval.
* **Constraints:** Use **NOT NULL** constraints to ensure required fields are filled.
* **Stored Procedures:** Automate common queries, such as retrieving **patient history**.