

Clinic Management System

Thatavarthi Sudhansh (23csb0b14)

May 04th 2025

1.INTRODUCTION: What is a clinic Management System?

In today’s fast-paced healthcare environment, the need for efficient, streamlined, and accurate management of clinical operations is more critical than ever. A **Clinic Management System (CMS)** is a comprehensive software solution designed to handle the day-to-day operations of a clinic with minimal human intervention. This system facilitates the seamless coordination of medical, administrative, and financial activities, ensuring better patient care and improved clinic performance.

The CMS integrates modules for appointment scheduling, patient records, billing and invoicing, inventory management, and reporting. By automating routine tasks and maintaining digital records, it reduces administrative burden, eliminates paperwork, and enables medical professionals to focus more on patient care. The system also enhances data accuracy, ensures regulatory compliance, and allows for data-driven decision-making.

Whether it’s a small practice or a multi-specialty clinic, a robust Clinic Management System lays the foundation for a modern, responsive, and patient-centred healthcare service.

2.PROBLEM STATEMENT

*We aim to develop a Database System for a Clinic Management Platform. The primary responsibilities of this system include:*

1. **Maintaining detailed patient records**, including personal information, contact details, age, gender, blood group, and address.
2. **Scheduling and managing appointments** between patients and doctors, including appointment type, status, date, and time.
3. **Storing doctor information**, including experience, contact details, and areas of specialization.
4. **Managing room allocations** for patients, including tracking room types, availability, floor, and date-wise occupancy history.
5. **Recording medical history and diagnoses** via medical records assigned to each patient with associated doctor notes and diagnosis details.
6. **Storing and tracking medicines** along with their manufacturers, expiry dates, stock levels, and pricing.
7. **Linking medical records with prescribed medicines**, capturing dosage, frequency, and duration of each prescribed drug.
8. **Generating and managing billing information**, including payment methods, total charges, and payment status for each patient.
9. **Organizing specialization and department data**, linking doctors with their expertise and supporting administrative categorization.

3. KEY FEATURES OF THE CLINIC MANAGEMENT SYSTEM

**Patient Information and Medical Record Management:**

The system maintains comprehensive records for each patient, including personal details like name, age, gender, phone number, address, and blood group. Medical records are linked with each patient and assigned doctor, storing diagnosis and consultation notes securely. Data integrity is ensured by assigning unique IDs and restricting unauthorized access.

**Doctor and Specialization Management:**

Doctor details such as name, experience, contact information, and associated departments are stored systematically. Each doctor can have multiple specializations, which are managed through a dedicated linking structure. Specializations are mapped to departments to support efficient categorization and clinical coordination.

**Appointment Scheduling and Tracking:**

The platform allows easy scheduling of appointments between patients and doctors, capturing type, status, date, and time. Appointment history is maintained for continuity of care and audit purposes. The system prevents double-booking and supports status updates (e.g., scheduled, completed, cancelled).

**Medicine Inventory and Prescription Linking:**

Medicines are tracked with details like name, manufacturer, expiry date, price, and available stock. Each medicine can be linked to medical records via dosage, frequency, and duration fields. Stock levels are updated in real-time to prevent shortages and support inventory control.

**Room Assignment and Occupancy Management:**

Patients can be assigned rooms based on availability, with type, floor, and status details maintained. The system tracks historical room occupancy using from-date and to-date fields, allowing administrators to manage admissions and optimize room utilization.

**Billing and Payment Record System:**

Billing data is linked to each patient, capturing payment method, amount, date, and payment status. The system supports various payment methods such as cash, card, and online payments. Billing records ensure accountability and provide transparency in financial transactions.

**Department and Staff Coordination:**

Departments are maintained as a separate entity linked to specializations. This supports administrative clarity and enables structured staff deployment. Workload distribution and departmental data can be used for planning and resource allocation.

4. ENTITY-RELATIONSHIP MODEL



5.ENTITIES

**PATIENT**: Represents a person who receives medical care at the clinic.  
*Attributes*: patient\_id, patient\_name, gender, age, bloodgroup, phone\_number, address  
*Primary Key*: patient\_id

**DOCTOR**: Represents a doctor working in the clinic.  
*Attributes*: doctor\_id, doctor\_name, phone\_number, doctor\_email, experience, license\_number  
*Primary Key*: doctor\_id

**SPECIALIZATION**: Represents a medical department or specialty linked to doctors.  
*Attributes*: specialization\_name, department\_name, description  
*Primary Key*: specialization\_name

**APPOINTMENT**: Represents a scheduled visit between a patient and a doctor.  
*Attributes*: appointment\_id, appointment\_date, appointment\_time, appointment\_type, appointment\_status  
*Primary Key*: appointment\_id

**MEDICALRECORD**: Represents medical information recorded for a patient.  
*Attributes*: record\_id, record\_date, diagnosis, notes  
*Primary Key*: record\_id

**MEDICINE**: Represents a medicine prescribed or stored in the clinic.  
*Attributes*: medicine\_id, medicine\_name, manufacturer, expiry\_date, price, stock  
*Primary Key*: medicine\_id

**ROOM**: Represents a room that can be assigned to patients.  
*Attributes*: room\_id, room\_number, room\_type, room\_floor, status  
*Primary Key*: room\_id

**BILLS**: Represents billing information for a patient.  
*Attributes*: bill\_id, bill\_date, amount, payment\_method, payment\_status  
*Primary Key*: bill\_id

6.RELATIONSHIPS

**BOOKS**  
Between: *PATIENT* (totally participates) and *APPOINTMENT* (totally participates)  
Cardinality: one PATIENT to many APPOINTMENT

**Assumption**: A patient can book multiple appointments, and each appointment is linked to one patient.  
As it is one-to-many, **Primary Key of PATIENT** is added to the APPOINTMENT table.

**ATTENDS**  
Between: *DOCTOR* (totally participates) and *APPOINTMENT* (partially participates)  
Cardinality: one DOCTOR to many APPOINTMENT

**Assumption**: A doctor can attend many appointments, but an appointment may exist without a doctor assigned yet.  
As it is one-to-many, **Primary Key of DOCTOR** is added to the APPOINTMENT table.

**HAS**  
Between: *PATIENT* (totally participates) and *MEDICALRECORD* (totally participates)  
Cardinality: one PATIENT to many MEDICALRECORD

**Assumption**: A patient has at least one medical record; each medical record belongs to one patient.  
As it is one-to-many, **Primary Key of PATIENT** is added to the MEDICALRECORD table.

**PRESCRIBES**  
Between: *MEDICALRECORD* (partially participates) and *MEDICINE* (partially participates)  
Cardinality: many-to-many

**Assumption**: A medical record can include multiple medicines, and a medicine can be prescribed in many medical records.  
As it is many-to-many, a **bridge table** is created with **Primary Keys of MEDICALRECORD and MEDICINE**.

**ASSIGNED**  
Between: *ROOM* (partially participates) and *PATIENT* (partially participates)  
Cardinality: many ROOM to many PATIENT

**Assumption**: A room can be assigned to many Patient, but some rooms may be unassigned.  
As it is one-to-many, **Primary Key of ROOM** is added to the Patient table.

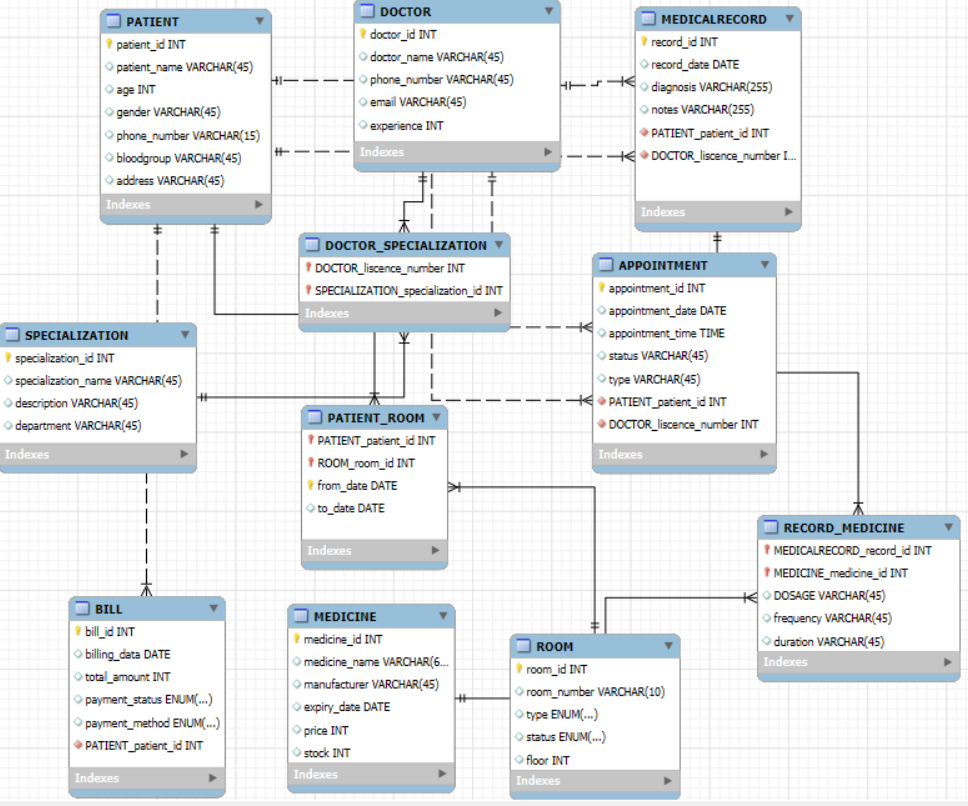
**RECEIVES**  
Between: *PATIENT* (totally participates) and *BILLS* (totally participates)  
Cardinality: one PATIENT to many BILLS

**Assumption**: Every patient receives at least one bill, and each bill belongs to a specific patient.  
As it is one-to-many, **Primary Key of PATIENT** is added to the BILLS table.

**SPECIALIZES\_IN**  
Between: *DOCTOR* (totally participates) and *SPECIALIZATION* (totally participates)  
Cardinality: many-to-many

**Assumption**: A doctor can have multiple specializations, and a specialization may apply to multiple doctors.  
As it is many-to-many, a **bridge table** is created with **Primary Keys of DOCTOR and SPECIALIZATION**.

7.RELATION SCHEMA AND NORMALIZATION



**1. PATIENT**

**FDs:**  
Patient\_ID → Name, Age, Gender, Address, Blood\_Group  
**Primary Key:** Patient\_ID  
**Normal Form:** Relation is in **BCNF**

**2. PATIENT\_PHONE**

**FDs:**  
Patient\_ID, Phone\_Number → (No other attributes)  
**Primary Key:** Patient\_ID, Phone\_Number  
**Normal Form:** Relation is in **BCNF**

**3. DOCTOR**

**FDs:**  
Doctor\_ID → Name, Email, Experience  
Email → Doctor\_ID (since it's unique)  
**Primary Key:** Doctor\_ID  
**Normal Form:** Relation is in **BCNF**

**4. DOCTOR\_PHONE**

**FDs:**  
Doctor\_ID, Phone\_Number → (No other attributes)  
**Primary Key:** Doctor\_ID, Phone\_Number  
**Normal Form:** Relation is in **BCNF**

**5. DEPARTMENT**

**FDs:**  
Department\_ID → Department\_Name, Description  
Department\_Name → Department\_ID (since name is unique)  
**Primary Key:** Department\_ID  
**Normal Form:** Relation is in **BCNF**

**6. SPECIALIZATION**

**FDs:**  
Name → Description, Department\_ID  
**Primary Key:** Name  
**Normal Form:** Relation is in **BCNF**

**7. DOCTOR\_SPECIALIZATION**

**FDs:**  
Doctor\_ID, Specialization\_Name → (No other attributes)  
**Primary Key:** Doctor\_ID, Specialization\_Name  
**Normal Form:** Relation is in **BCNF**

**8. APPOINTMENT**

**FDs:**  
Appointment\_ID → Patient\_ID, Doctor\_ID, Date, Time, Status, Type  
**Primary Key:** Appointment\_ID  
**Normal Form:** Relation is in **BCNF**

**9. MEDICAL\_RECORD**

**FDs:**  
Record\_ID → Patient\_ID, Doctor\_ID, Date, Diagnosis, Notes  
**Primary Key:** Record\_ID  
**Normal Form:** Relation is in **BCNF**

**10. RECORD\_MEDICINE**

**FDs:**  
Record\_ID, Medicine\_ID → Dosage, Frequency, Duration  
**Primary Key:** Record\_ID, Medicine\_ID  
**Normal Form:** Relation is in **BCNF**

**11. MEDICINE**

**FDs:**  
Medicine\_ID → Name, Manufacturer, Expiry\_Date, Price, Stock  
**Primary Key:** Medicine\_ID  
**Normal Form:** Relation is in **BCNF**

**12. ROOM**

**FDs:**  
Room\_ID → Room\_Number, Type, Status, Floor  
**Primary Key:** Room\_ID  
**Normal Form:** Relation is in **BCNF**

**13. PATIENT\_ROOM**

**FDs:**  
Patient\_ID, Room\_ID, From\_Date → To\_Date  
**Primary Key:** Patient\_ID, Room\_ID, From\_Date  
**Normal Form:** Relation is in **BCNF**

**14. BILL**

**FDs:**  
Bill\_ID → Patient\_ID, Date, Total\_Amount, Payment\_Status, Payment\_Method  
**Primary Key:** Bill\_ID  
**Normal Form:** Relation is in **BCNF**

Why I created 3 new tables during normalization?

Solution:

1. **PATIENT\_PHONE**
   * In the original PATIENT table, a patient could have more than one phone number.
   * But according to 1NF, each field should have atomic (single) values.
   * So, I separated the phone numbers into a new table to avoid storing multiple numbers in one cell.
   * Now, each phone number is stored as a separate row with the corresponding Patient\_ID.
2. **DOCTOR\_PHONE**
   * Just like patients, doctors can also have multiple contact numbers.
   * Keeping all phone numbers in one column violates 1NF.
   * So, I made a separate table to handle doctor phone numbers properly.
   * This way, one doctor can have any number of phone numbers, and the table stays normalized.
3. **DOCTOR\_SPECIALIZATION**
   * A single doctor can have multiple specializations, and the same specialization can belong to multiple doctors.
   * That’s a many-to-many relationship, which can't be stored properly in a single table.
   * So, I created a separate table to map each doctor to their specializations.
   * This keeps things clean and avoids redundancy, satisfying normalization rules.

8. SQL queries to create tables

CREATE TABLE PATIENT (

Patient\_ID INT PRIMARY KEY NOT NULL,

patient\_name VARCHAR(100),

Age INT,

Gender CHAR(1),

Address TEXT,

Blood\_Group VARCHAR(5)

);

CREATE TABLE PATIENT\_PHONE (

Patient\_ID INT NOT NULL,

Phone\_Number VARCHAR(15),

PRIMARY KEY (Patient\_ID, Phone\_Number),

FOREIGN KEY (Patient\_ID) REFERENCES PATIENT(Patient\_ID)

);

CREATE TABLE DOCTOR (

Doctor\_ID INT PRIMARY KEY NOT NULL,

doctor\_name VARCHAR(100),

Email VARCHAR(100) UNIQUE,

Experience INT

);

CREATE TABLE DOCTOR\_PHONE (

Doctor\_ID INT not null,

Phone\_Number VARCHAR(15),

PRIMARY KEY (Doctor\_ID, Phone\_Number),

FOREIGN KEY (Doctor\_ID) REFERENCES DOCTOR(Doctor\_ID)

);

drop table department;

CREATE TABLE DEPARTMENT (

Department\_ID INT PRIMARY KEY not null,

Department\_Name VARCHAR(100) UNIQUE,

Description\_ TEXT

);

CREATE TABLE SPECIALIZATION (

specialization\_name VARCHAR(100) PRIMARY KEY,

Description\_ TEXT,

Department\_ID INT,

FOREIGN KEY (Department\_ID) REFERENCES DEPARTMENT(Department\_ID)

);

CREATE TABLE DOCTOR\_SPECIALIZATION (

Doctor\_ID INT,

Specialization\_Name VARCHAR(100),

PRIMARY KEY (Doctor\_ID, Specialization\_Name),

FOREIGN KEY (Doctor\_ID) REFERENCES DOCTOR(Doctor\_ID),

FOREIGN KEY (Specialization\_Name) REFERENCES SPECIALIZATION(specialization\_Name)

);

CREATE TABLE ROOM (

Room\_ID INT PRIMARY KEY,

Room\_Number VARCHAR(10),

room\_type VARCHAR(50),

room\_status VARCHAR(20),

Floor INT

);

CREATE TABLE PATIENT\_ROOM (

Patient\_ID INT,

Room\_ID INT,

From\_Date DATE,

To\_Date DATE,

PRIMARY KEY (Patient\_ID, Room\_ID, From\_Date),

FOREIGN KEY (Patient\_ID) REFERENCES PATIENT(Patient\_ID),

FOREIGN KEY (Room\_ID) REFERENCES ROOM(Room\_ID)

);

CREATE TABLE APPOINTMENT (

Appointment\_ID INT PRIMARY KEY,

Patient\_ID INT,

Doctor\_ID INT,

appoint\_Date DATE,

appoint\_time TIME,

appoint\_status VARCHAR(20),

appoint\_type VARCHAR(50),

FOREIGN KEY (Patient\_ID) REFERENCES PATIENT(Patient\_ID),

FOREIGN KEY (Doctor\_ID) REFERENCES DOCTOR(Doctor\_ID)

);

CREATE TABLE MEDICAL\_RECORD (

Record\_ID INT PRIMARY KEY,

Patient\_ID INT,

Doctor\_ID INT,

record\_date DATE,

Diagnosis TEXT,

Prescription TEXT,

Notes TEXT,

FOREIGN KEY (Patient\_ID) REFERENCES PATIENT(Patient\_ID),

FOREIGN KEY (Doctor\_ID) REFERENCES DOCTOR(Doctor\_ID)

);

CREATE TABLE MEDICINE (

Medicine\_ID INT PRIMARY KEY,

medicine\_name VARCHAR(100),

Manufacturer VARCHAR(100),

Expiry\_Date DATE,

Price int,

Stock INT

);

CREATE TABLE RECORD\_MEDICINE (

Record\_ID INT,

Medicine\_ID INT,

Dosage VARCHAR(50),

Frequency VARCHAR(50),

Duration VARCHAR(50),

PRIMARY KEY (Record\_ID, Medicine\_ID),

FOREIGN KEY (Record\_ID) REFERENCES MEDICAL\_RECORD(Record\_ID),

FOREIGN KEY (Medicine\_ID) REFERENCES MEDICINE(Medicine\_ID)

);

;

CREATE TABLE BILL (

Bill\_ID INT PRIMARY KEY,

Patient\_ID INT,

bill\_date DATE,

Total\_Amount int,

Payment\_Status VARCHAR(20),

Payment\_Method VARCHAR(30),

FOREIGN KEY (Patient\_ID) REFERENCES PATIENT(Patient\_ID)

);

9.SQL queries to insert data

INSERT INTO PATIENT VALUES

(1, 'Ravi Kumar', 34, 'M', 'Hyderabad', 'B+'),

(2, 'Anita Rao', 28, 'F', 'Bangalore', 'O+');

INSERT INTO PATIENT\_PHONE VALUES

(1, '9876543210'),

(1, '9123456789'),

(2, '9988776655');

INSERT INTO DOCTOR VALUES

(101, 'Dr. Meena Sharma', 'meena.sharma@gmail.com', 12),

(102, 'Dr. Arjun Verma', 'arjun.verma@gmail.com', 8);

INSERT INTO DOCTOR\_PHONE VALUES

(101, '9876512345'),

(102, '9123412345');

INSERT INTO DEPARTMENT VALUES

(1, 'Cardiology', 'Heart-related treatments'),

(2, 'Neurology', 'Brain and nervous system');

INSERT INTO SPECIALIZATION VALUES

('Cardiologist', 'Heart specialist', 1),

('Neurologist', 'Brain specialist', 2);

INSERT INTO DOCTOR\_SPECIALIZATION VALUES

(101, 'Cardiologist'),

(102, 'Neurologist');

INSERT INTO ROOM VALUES

(201, 'A101', 'General', 'Available', 1),

(202, 'A102', 'Private', 'Occupied', 2);

INSERT INTO PATIENT\_ROOM VALUES

(1, 201, '2025-06-01', '2025-06-05'),

(2, 202, '2025-06-02', '2025-06-04');

INSERT INTO APPOINTMENT VALUES

(301, 1, 101, '2025-06-03', '10:00:00', 'Completed', 'Regular Checkup'),

(302, 2, 102, '2025-06-04', '11:30:00', 'Pending', 'Consultation');

INSERT INTO MEDICAL\_RECORD VALUES

(401, 1, 101, '2025-06-03', 'Hypertension', 'Amlodipine 5mg', 'Take BP daily'),

(402, 2, 102, '2025-06-04', 'Migraine', 'Paracetamol 650', 'Avoid stress');

INSERT INTO MEDICINE VALUES

(501, 'Amlodipine', 'Sun Pharma', '2026-01-01', 15, 100),

(502, 'Paracetamol', 'Cipla', '2025-12-01', 5, 200);

INSERT INTO RECORD\_MEDICINE VALUES

(401, 501, '5mg', 'Once daily', '30 days'),

(402, 502, '650mg', 'Twice daily', '5 days');

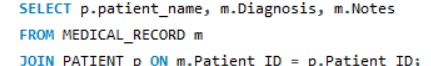
INSERT INTO BILL VALUES

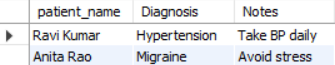
(601, 1, '2025-06-05', 800, 'Paid', 'Cash'),

(602, 2, '2025-06-04', 1000, 'Unpaid', 'UPI');

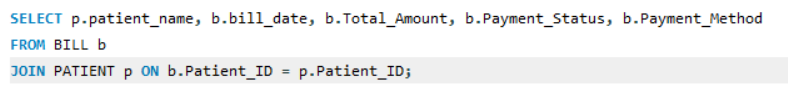
1. DATABASE IN ACTION

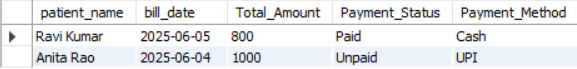
Q1) Show medical record details for each patient.



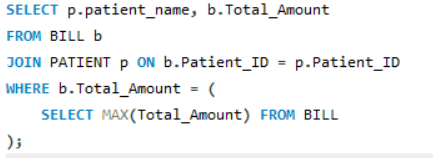


Q2) Find the billing details of all patients.



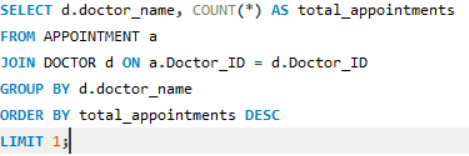


Q3) Who has paid the highest bill amount?





Q4) Which doctor is assigned to the most appointments





Q5) List the top 2 patients with the highest total bill amounts.

