GL BAJAJ
Institute of Technology & Management
FIND YOUR SPARK
Approved by AICTE & Affiliated to AKTU

DATABASE MANAGEMENT SYSTEM LAB

MINI PROJECT FILE
(BCS 551)

GL BAJAJ INSTITUTE OF TECHNOLOGY AND MANAGEMENT, GREATER NOIDA

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

Session: 2024-2025

Submitted By:

Name:
Section:
Roll no:
Semester:
Group:

Submitted To:

Acknowledgment

I would like to express my heartfelt gratitude to everyone who contributed to the successful completion of this mini-project on Database Management System (DBMS).

First and foremost, I am deeply thankful to my project guide, [Guide's Name], for their constant guidance, support, and valuable insights throughout this project. Their expertise and encouragement have been instrumental in shaping my understanding of the subject and in overcoming challenges during the project development.
I also extend my gratitude to [Professor's Name], our course instructor, for providing a comprehensive foundation in DBMS concepts and for inspiring me to explore this field further.

Additionally, I am grateful to G.L. Bajaj Institute of Technology & Management and the Department of Computer Science for providing the resources and a conducive learning environment, which enabled me to successfully complete this mini-project.
Last but not least, I would like to thank my family, friends, and classmates for their constant support and motivation during the course of this project.
This project has been a significant learning experience, and I am sincerely thankful to everyone who contributed, directly or indirectly, to its success.

Abstract

This project, titled HOSPITAL MANAGEMENT SYSTEM, focuses on the design and implementation of a Database Management System (DBMS) for Hospital Management System. The primary objective of this project is to create a robust, efficient, and user-friendly database system that manages and organizes data effectively while ensuring data integrity and accessibility.

The project involves the application of fundamental database concepts, such as Entity-Relationship (ER) modeling, normalization, and SQL queries, to design and implement a relational database. The ER diagram has been used to model the relationships between various entities, such as [some key entities, e.g., Patients, Doctors, Rooms, Bills], and the database has been normalized to minimize redundancy and improve efficiency.

The system enables functionalities such as [key functionalities, e.g., data entry, retrieval, updating, deletion, and generation of reports]. SQL queries were written for complex operations, ensuring the system meets real-world requirements. Additionally, the database system incorporates features like data validation, indexing, and constraints to enhance reliability and prevent anomalies.
This project serves as a practical implementation of the theoretical concepts learned in Database Management Systems and demonstrates the importance of databases in managing large volumes of structured data. It also provides insights into database design, optimization, and its application in solving real-world problems.

In conclusion, the project highlights the critical role of DBMS in automating and streamlining operations in Hospitals Management System, offering a foundation for future development and enhancement of database systems.

INDEX

Information related to the Hospital Management System

The organization represented in the ER diagram is a Hospital Management System that manages patients, employees, and facilities. Patients, identified by attributes like P_ID, Name, Gender, Age, DOB, and Mobile Number, consult doctors for medical care, pay bills, and have test reports associated with them. Each patient can consult multiple doctors, and each doctor, specialized in a specific department and qualification, can consult multiple patients.

Employees in the organization, identified by attributes like E_ID, Name, Sex, Salary, Mobile Number, and Address (State, City, Pin-Code), are categorized as Doctors, Nurses, and Receptionists. Doctors are associated with their department and qualifications, while Nurses manage room assignments in an M:N relationship. Receptionists are responsible for maintaining organizational records, such as appointment details, in a many-to-many relationship.

The system also tracks rooms, which have attributes such as R_ID, Type, Capacity, and Availability. Nurses coordinate with room assignments to ensure operational efficiency. Additionally, bills are issued for patient services, containing attributes like Bill ID (B_ID), Patient ID (P_ID), and Amount. Patients can pay multiple bills, while each bill corresponds to a single patient.

Test reports, containing attributes like Report ID (R_ID), Test Type, Result, and Patient ID (P_ID), are linked to patients. Each patient can have multiple test reports associated with their medical care.

Overall, the system ensures seamless management of healthcare services by integrating patient care, employee roles, billing, record maintenance, and room allocation, facilitating efficient hospital operations.

Overview of Hospital Management System

Hospital Management System (HMS) is a comprehensive software tool devised to fast several hospital operations through a single and efficient system. It includes patient management, doctor management, room management, nurse management, test report management, personal data management, billing and secretary management.

Hospital Management System Features

To design an **Entity-Relationship Diagram** (**ER Diagram**) for a **Hospital Management System** (**HMS**) with the entities provided, let's focus on the following requirements:

- **Patient Management:** The platform should also allow for the **patients registration** and the management of their information with details such as **demographics**, **medical history**, and **current condition** in mind.
- **Doctor Management:** The system should have the functionality of a doctor directory in which details such as **doctor's specialties**, **contacts**, and **availability** can be set.
- **Room Management:** The system should be able to assist the department in assigning and inspecting rooms in the **hospital**, particularly those that are available or not currently occupied.
- **Nurse Management:** The software should contain necessary functions like the management of **nurse** data like information about their **departments**, **shifts**, and **contact** details.
- **Test Report Management:** The system provides a mechanism for the generation, **storage**, and retrieval of test reports on **patients**, which gives detailed information on the **tests**, **results**, and **dates**.
- **Record Management:** The system should keep track of hospital activities in their records. Each record should have a **unique identification** number for identification.
- **Billing:** The system will be handling billing details of services rendered to patients and this will cover all functions such as **bill generation, tracking payments and management of insurance** details.
- **Receptionist Management:** The system will have functions that will be able to handle a receptionist role in the hospital. Receptionists should be able to retrieve relevant **patient data** and do booking of **appointments schedules**.

Entities and Attributes of the Hospital Management System

A **thing** in the real world with an independent existence. It is may be an object with physical existence (ex: house, person) or with a conceptual existence (ex: course, job). The are represented by **rectangle.**
Let's Defining Entities for Hospital Management System are:

1. Patient

- **P-ID**: Unique identifier for each Patient
- **Name**: Name of the Patient.
- **DOB**: Date of borthf of Patient.
- **Gender**: Gender of Patient.
- **Mob-No**: Contact number of the Patient.
- **Age:** Age of Patient.

2. Employee

- **E-ID:** Unique identifier for each Employee.
- **Name:** Name of the Employee.
- **Salary:** Salary of Employee
- **Sex:** Gender of Employee.
- **Mob-No:** Contact number of the Employee.
- **Address:** Address of Employee.
- **State**: State of Employee
- **City**: city of Employee
- **Pin–no**: Pin no of Employee

3. Doctor

- **E-ID** (Foreign Key referencing Employee):
- **Department**: Department of doctor.
- **Qualification**: Qualification of Doctor

4. Nurse

- **E-ID:** E-ID is a foreign key linking a table to the Employee table through the **Employee ID.**

5. Room

- **R-ID:** It is an room id every room has different room number or ID.
- **Type:** It define the quality of room such as deluxe, private general etc.
- **Capacity:** It defines the number of people can stay in room.
- **Availability:** It define the duration or Availability of room.

6. Receptionist

- **E-ID** (Foreign Key referencing Employee)**:** E-ID is a foreign key in a table that references the Employee table, typically used to establish a relationship between the two tables based on the Employee ID.

7. Test Report

- **R-ID (Primary Key):** Unique identifier for each **Room.**
- **P-ID (Foreign Key referencing Patient):** P-ID is a foreign key in a table that references the Patient table, typically used to establish a relationship between the two tables based on the Patient ID.
- **Test Type:** It define the what kinf of test.
- **Result:** It shows the test result.

8. Bill

- **B-ID:** Unique identifier for each **Bill.**
- **P-ID** (Foreign Key referencing Patient): P-ID is a foreign key in a table that references the Patient table, typically used to establish a relationship between the two tables based on the Patient ID.
- **Amount:** The Amount which Patient has to pay to the Hospital.

9. Records

- **Record-no:** Every record book has some number for each **Patient**.
- **App-no:** Every app book has some number for each **Patient**.

Establishing Relationships

Entities have some relationships with each other. Relationships define how **entities are associated** with each other.

Let's Establishing Relationships between them are:

- Patient consults Doctor.
- Employee have roles as a nurse, doctor and receptionist within the hospital.
- Patient pays bills for medical services.
- Nurse governs rooms.
- Patient assigned rooms during their stay at hospital.
- Receptionist maintains hospital records.
- Patient has test report.

Relationships Between These Entities:

1. Patient – Doctor Relationship

- A patient can have a relationship with one or more doctors for consultations or treatments.
- A doctor can have multiple patients.
- This is a **Many-to-Many** (Patient-to-Doctor) as multiple Patient can visit multiple Doctor.

2. Nurse – Rooms Relationship

- A nurse can be assigned to one or more rooms during their shift.
- A room can have multiple nurses assigned to it over different shifts.
- This is a **many-to-many** relationship between **nurses** and **rooms**, each nurse can be assigned to multiple rooms, and each room can have multiple nurses assigned to it.

3. Receptionist – Records Relationship

- A receptionist manages records which could include patient records, appointment schedules, or other administrative documents.
- A record can be managed by one or more receptionists.
- This is a **many-to-many** relationship between **receptionists** and **records**, each receptionist can manage multiple records, and each record can be managed by multiple receptionists.

4. Patient – Bills Relationship

- One patient can have multiple bills.
- One bill is associated with only one patient.
- This is a **one-to-many** relationship between **patients** and **bills**, each patient can have multiple bills, but each bill belongs to only one patient.

5. Patient – Test Report Relationship

- One patient can have multiple test reports.
- One test report is associated with only one patient.

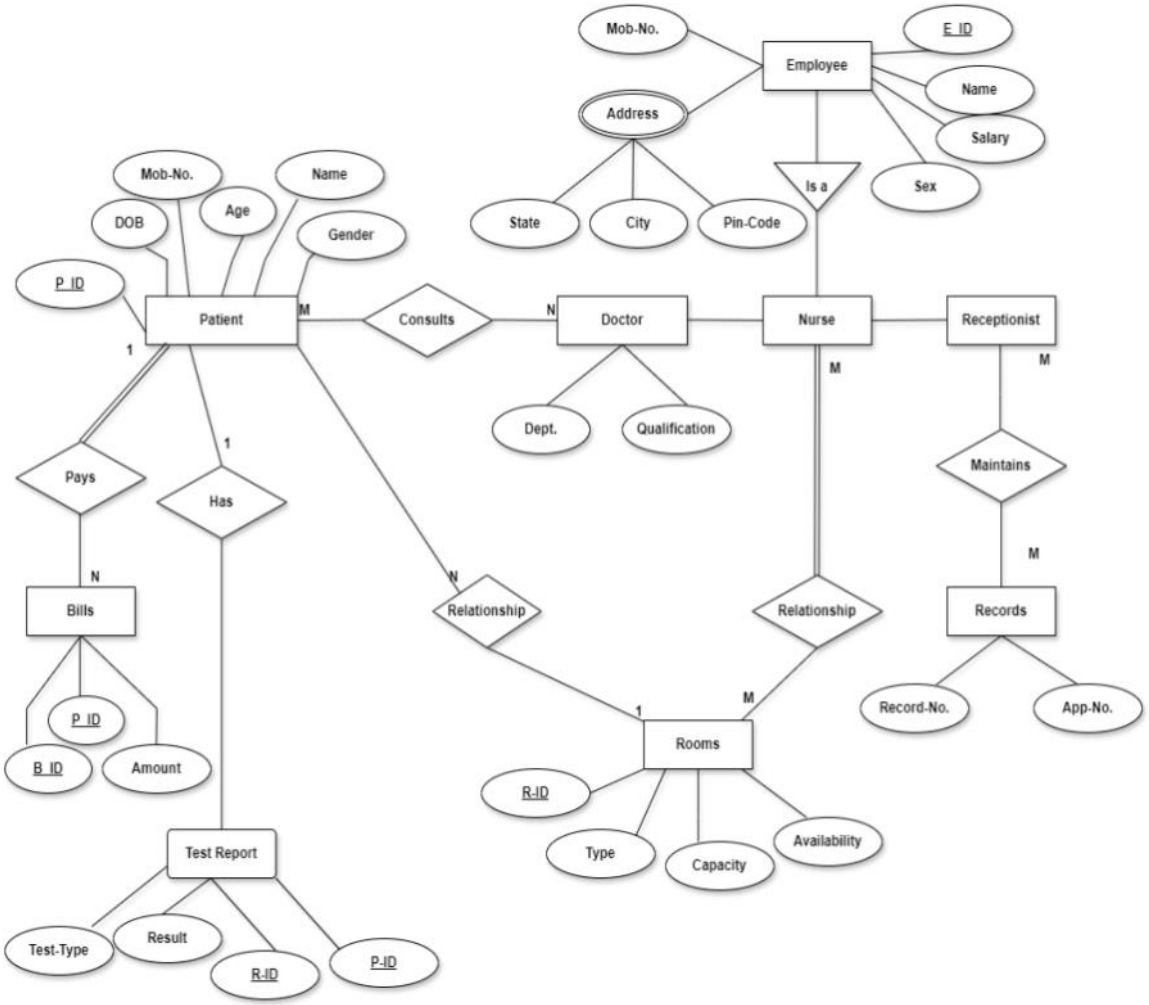
- This is a **one-to-many** relationship between **patients** and **test** reports, each patient can have multiple test reports, but each test report belongs to only one patient.

6. Rooms – Patient Relationship

- One room can accommodate multiple patients over time.
- One patient occupies one room at a time.

This is a **one-to-many** relationship between rooms and patients, each room can accommodate multiple patients, but each patient occupies only one room at a time.

ER Diagram


Mob-No.
E_ID
Employee
Name
Address
Salary
Is a
Sex
State
City
Pin-Code
Mob-No.
Name
DOB
Age
Gender
P_ID
Patient
M
Consults
N
Doctor
Nurse
Receptionist
1
M
M
Dept.
Qualification
1
Pays
Has
Maintains
N
M
Bills
N
Relationship
Relationship
Records
P_ID
M
1
M
Rooms
Record-No.
App-No.
B_ID
Amount
R-ID
Type
Capacity
Availability
Test Report
Test-Type
Result
R-ID
P-ID

Reduce ER Diagram into Tables

1. Patient Table

Attribute	Data Type	Description
P_ID	INT (PK)	Unique identifier for patient
Name	VARCHAR(100)	Name of the patient
Gender	CHAR(1)	Gender of the patient (M/F)
DOB	DATE	Date of birth of the patient
Age	INT	Age of the patient
Mob-No	VARCHAR(15)	Mobile number of the patient

2. Employee Table

Attribute	Data Type	Description
E_ID	INT (PK)	Unique identifier for employee
Name	VARCHAR(100)	Name of the employee
Salary	DECIMAL(10, 2)	Salary of the employee
Sex	CHAR(1)	Gender of the employee (M/F)
Mob-No	VARCHAR(15)	Mobile number of the employee
Address	VARCHAR(200)	Address of the employee
State	VARCHAR(50)	State of residence
City	VARCHAR(50)	City of residence
Pin-Code	VARCHAR(10)	Postal code

3. Doctor Table

Attribute	Data Type	Description
E_ID	INT (PK, FK)	Linked to Employee table
Dept	VARCHAR(100)	Department of the doctor
Qualification	VARCHAR(100)	Doctor's qualifications

4. Nurse Table

Attribute	Data Type	Description
E_ID	INT (PK, FK)	Linked to Employee table
Relationship	VARCHAR(50)	Nurse-patient relationship type

5. Receptionist Table

Attribute	Data Type	Description
E_ID	INT (PK, FK)	Linked to Employee table

6. Rooms Table

Attribute	Data Type	Description
R_ID	INT (PK)	Unique identifier for a room
Type	VARCHAR(50)	Room type (e.g., General, ICU)
Capacity	INT	Number of patients the room can hold
Availability	BOOLEAN	Availability status of the room

7. Test Report Table

Attribute	Data Type	Description
R_ID	INT (PK, FK)	Linked to Rooms table
P_ID	INT (PK, FK)	Linked to Patient table
Test-Type	VARCHAR(50)	Type of test conducted
Result	TEXT	Test result details

8. Bills Table

Attribute	Data Type	Description
B_ID	INT (PK)	Unique identifier for a bill
P_ID	INT (FK)	Linked to Patient table
Amount	DECIMAL(10, 2)	Total amount billed

9. Records Table

Attribute	Data Type	Description
Record-No	INT (PK)	Unique identifier for a record
App-No	INT	Appointment number linked to the record

10. Consults Table (Relationship between Patient and Doctor)

Attribute	Data Type	Description
P_ID	INT (PK, FK)	Linked to Patient table
E_ID	INT (PK, FK)	Linked to Employee table for doctors

11. Maintains Table (Relationship between Receptionist and Records)

Attribute	Data Type	Description
E_ID	INT (PK, FK)	Linked to Employee table for receptionists
Record-No	INT (PK, FK)	Linked to Records table

Relational Algebra, and SQL Queries

Query 1: Retrieve all patient details.

Relational Algebra: $\pi_{(P\_ID, Name, Gender, Age, DOB, Mob\text{-}No)}$ (Patient)

SQL: SELECT P_ID, Name, Gender, Age, DOB, Mob_No

FROM Patient;

Query 2: Find all doctors working in a specific department (e.g., "Cardiology").

Relational Algebra: $\sigma_{(Dept = 'Cardiology')}$(Doctor)

SQL: SELECT *

FROM Doctor

WHERE Dept = 'Cardiology';

Query 3: List all patients who have consulted a specific doctor.

Relational Algebra: $\pi_{(P\_ID, Name)}$(Patient⋈Patient.P_ID = Consults.P_ID Consults⋈Consults.E_ID = Doctor.E_IDDoctor)

SQL: SELECT Patient.P_ID, Patient.Name

FROM Patient

JOIN Consults ON Patient.P_ID = Consults.P_ID

JOIN Doctor ON Consults.E_ID = Doctor.E_ID;

Query 4: Find the total number of patients assigned to each doctor.

Relational Algebra: γE_ID,count(P_ID) \rightarrowPatient_Count(Consults)

SQL: SELECT E_ID, COUNT(P_ID) AS Patient_Count

FROM Consults

GROUP BY E_ID;

Query 5: Get details of all nurses who maintain rooms.

Relational Algebra: $\pi_{(E\_ID, Name, Qualification)}$(Nurse⋈Relationship⋈Rooms)

SQL: SELECT Nurse.E_ID, Employee.Name, Nurse.Qualification

FROM Nurse

JOIN Relationship ON Nurse.E_ID = Relationship.E_ID

JOIN Rooms ON Relationship.R_ID = Rooms.R_ID;

Query 6: Find patients who have unpaid bills.

Relational Algebra: $\sigma_{(Amount > 0)}(Bills)$

SQL: SELECT P_ID, B_ID, Amount

FROM Bills

WHERE Amount > 0;

Query 7: Retrieve room details and their availability status.

Relational Algebra: $\pi_{(R\_ID, Type, Capacity, Availability)}(Rooms)$

SQL: SELECT R_ID, Type, Capacity, Availability

FROM Rooms;

Query 8: Find all patients who have undergone a specific test type (e.g., "Blood Test").

Relational Algebra: $\sigma_{(Test\text{-}Type = 'Blood\ Test')}(Test\ Report)$

SQL: SELECT P_ID, Test_Type, Result

FROM Test_Report

WHERE Test_Type = 'Blood Test';

Query 9: Get details of all employees working in the hospital.

Relational Algebra: $\pi_{(E\_ID, Name, Salary, Sex)}(Employee)$

SQL: SELECT E_ID, Name, Salary, Sex

FROM Employee;

Query 10: List all appointments maintained by receptionists.

Relational Algebra: $\pi_{(Record\text{-}No, App\text{-}No)}(Records)$

SQL: SELECT Record_No, App_No

FROM Records;

Query 11: Find the details of all bills paid by a specific patient (e.g., P_ID = 101).

Relational Algebra: $\sigma_{(P\_ID = 101)}(Bills)$

SQL: SELECT B_ID, Amount

FROM Bills

WHERE P_ID = 101;

Query 12: Retrieve names and contact details of all employees working in a specific city (e.g., "New York").

Relational Algebra: $\sigma_{(City = 'New York')}$(Employee)

SQL: SELECT Name, Mob_No, Address

FROM Employee

WHERE City = 'New York';

Query 13: List all rooms currently maintained by a specific nurse (E_ID = 201).

Relational Algebra: πR_ID, Type, Capacity, Availability(σE_ID = 201(Relationship⋈Rooms))

SQL: SELECT Rooms.R_ID, Rooms.Type, Rooms.Capacity, Rooms.Availability

FROM Rooms

JOIN Relationship ON Rooms.R_ID = Relationship.R_ID

WHERE Relationship.E_ID = 201;

Query 14: Retrieve all test reports of patients who have undergone tests in a specific room (R_ID = 301).

Relational Algebra: πP_ID, Test-Type, Result(σR_ID = 301(Test Report))

SQL: SELECT P_ID, Test_Type, Result

FROM Test_Report

WHERE R_ID = 301;

Query 15: Find the name and age of the youngest patient

Relational Algebra: πName, Age(σAge=MIN(Age)(Patient))

SQL: SELECT Name, Age

FROM Patient

WHERE Age = (SELECT MIN(Age) FROM Patient);

Database

sql queries

```
-- Create Database
CREATE DATABASE hospital_db;

USE hospital_db;

-- Patients Table
CREATE TABLE Patients (
    P_ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100),
    Gender ENUM('Male', 'Female', 'Other'),
    Age INT,
    DOB DATE,
    Mob_No VARCHAR(15)
);

-- Employees Table
CREATE TABLE Employees (
    E_ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100),
    Mob_No VARCHAR(15),
    Attribute VARCHAR(50),
    State VARCHAR(50),
    City VARCHAR(50),
    Pin_Code VARCHAR(10),
    Salary DECIMAL(10, 2),
    Sex ENUM('Male', 'Female')
);
```

```
-- Doctors Table
CREATE TABLE Doctors (
    E_ID INT PRIMARY KEY,
    Dept VARCHAR(50),
    FOREIGN KEY (E_ID) REFERENCES Employees(E_ID)
);

-- Nurses Table
CREATE TABLE Nurses (
    E_ID INT PRIMARY KEY,
    Qualification VARCHAR(50),
    FOREIGN KEY (E_ID) REFERENCES Employees(E_ID)
);

-- Rooms Table
CREATE TABLE Rooms (
    R_ID INT AUTO_INCREMENT PRIMARY KEY,
    Type VARCHAR(50),
    Capacity INT,
    Availability ENUM('Available', 'Occupied')
);

-- Test Reports Table
CREATE TABLE TestReports (
    R_ID INT AUTO_INCREMENT PRIMARY KEY,
    P_ID INT,
    Test_Type VARCHAR(100),
    Result VARCHAR(255),
    FOREIGN KEY (P_ID) REFERENCES Patients(P_ID)
```

```
);

-- Bills Table
CREATE TABLE Bills (
    B_ID INT AUTO_INCREMENT PRIMARY KEY,
    P_ID INT,
    Amount DECIMAL(10, 2),
    FOREIGN KEY (P_ID) REFERENCES Patients(P_ID)
);

-- Records Table
CREATE TABLE Records (
    Record_No INT AUTO_INCREMENT PRIMARY KEY,
    App_No VARCHAR(50)
);
```

Normlization

1. Patients Table

The Patients table seems to already meet 1NF, but we can break down Mob_No and DOB to ensure full compliance with the rules of 2NF.

```
CREATE TABLE Patients (
    P_ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100),
    Gender ENUM('Male', 'Female', 'Other'),
    Age INT,
    DOB DATE
);
```

- Mob_No can be in a separate table because it may not be unique for each patient (for example, a patient might have multiple emergency contacts). We will create a new table for contact information.

2. Employees Table

The Employees table appears mostly normalized, but we can make minor changes. Since Salary and Attribute seem to depend on the employee role (which is more appropriately stored in a **Roles** table), we will separate them out.

```
CREATE TABLE Employees (
    E_ID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100),
    Mob_No VARCHAR(15),
    Sex ENUM('Male', 'Female')
);
CREATE TABLE EmployeeRoles (
    E_ID INT PRIMARY KEY,
    Attribute VARCHAR(50),
    State VARCHAR(50),
    City VARCHAR(50),
```

```
    Pin_Code VARCHAR(10),
    Salary DECIMAL(10, 2),
    FOREIGN KEY (E_ID) REFERENCES Employees(E_ID)
);
```

- EmployeeRoles stores information specific to the role (e.g., address, salary), ensuring it is separated from general employee details.

3. Doctors Table

The Doctors table is fine for 2NF, as it only contains information directly related to the doctor, but we can normalize it further by separating the Dept attribute into its own table (if departments are likely to be reused across many doctors).

```
CREATE TABLE Doctors (
    E_ID INT PRIMARY KEY,
    Dept_ID INT,
    FOREIGN KEY (E_ID) REFERENCES Employees(E_ID),
    FOREIGN KEY (Dept_ID) REFERENCES Departments(Dept_ID)
);
CREATE TABLE Departments (
    Dept_ID INT AUTO_INCREMENT PRIMARY KEY,
    Dept_Name VARCHAR(50)
);
```

- This allows us to normalize the Dept column into its own table, ensuring that the department is consistent and can be reused.

4. Nurses Table

Similarly, the Nurses table is normalized, but we can break down qualifications into a separate table if there are many qualifications.

```
CREATE TABLE Nurses (
    E_ID INT PRIMARY KEY,
    Qualification_ID INT,
    FOREIGN KEY (E_ID) REFERENCES Employees(E_ID),
    FOREIGN KEY (Qualification_ID) REFERENCES Qualifications(Qualification_ID)
```

```
);
CREATE TABLE Qualifications (
    Qualification_ID INT AUTO_INCREMENT PRIMARY KEY,
    Qualification VARCHAR(50)
);
```

5. Rooms Table

The Rooms table looks fine, but we can break it down into a RoomTypes table if Type is reused in multiple rooms, ensuring no redundancy.

```
CREATE TABLE Rooms (
    R_ID INT AUTO_INCREMENT PRIMARY KEY,
    Room_Type_ID INT,
    Capacity INT,
    Availability ENUM('Available', 'Occupied'),
    FOREIGN KEY (Room_Type_ID) REFERENCES RoomTypes(Room_Type_ID)
);
CREATE TABLE RoomTypes (
    Room_Type_ID INT AUTO_INCREMENT PRIMARY KEY,
    Type VARCHAR(50)
);
```

6. Test Reports Table

The TestReports table appears to be fine for normalization. However, if there are many different types of tests, we can split the Test_Type into a new table.

```
CREATE TABLE TestReports (
    R_ID INT AUTO_INCREMENT PRIMARY KEY,
    P_ID INT,
    Test_Type_ID INT,
    Result VARCHAR(255),
    FOREIGN KEY (P_ID) REFERENCES Patients(P_ID),
    FOREIGN KEY (Test_Type_ID) REFERENCES TestTypes(Test_Type_ID)
```

```
);
CREATE TABLE TestTypes (
    Test_Type_ID INT AUTO_INCREMENT PRIMARY KEY,
    Test_Type VARCHAR(100)
);
```

7. Bills Table

The Bills table is already normalized in terms of its relationships. It is directly related to the Patients table, and since Amount is directly related to the bill, no further normalization is needed here.

8. Records Table

The Records table seems fine, but if the App_No is a unique appointment number, it should be a primary key. If App_No is simply an identifier for appointments, it could reference another Appointments table.

```
CREATE TABLE Records (
    Record_No INT AUTO_INCREMENT PRIMARY KEY,
    App_No VARCHAR(50),
    FOREIGN KEY (App_No) REFERENCES Appointments(App_No)
);
CREATE TABLE Appointments (
    App_No VARCHAR(50) PRIMARY KEY,
    P_ID INT,
    Date DATE,
    Time TIME,
    Doctor_E_ID INT,
    FOREIGN KEY (P_ID) REFERENCES Patients(P_ID),
    FOREIGN KEY (Doctor_E_ID) REFERENCES Doctors(E_ID)
);
```