**[HOSPITAL MANAGEMENT SYSTEM]**

Project submitted for the partial fulfillment of the requirements for the course

**CSE 209: DBMS**

Offered by the

**Department Computer Science and Engineering**

**School of Engineering and Science**

Submitted by

1) M.Mithesh(AP22110010195)



**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[APRIL, 2024]**

**Certificate**

Date: 02-05-2024

This is to certify that the work present in this Project entitled “HOSPITAL MANAGEMENT SYSTEM” has been carried out by [MITHESH MALINENI] under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the

award of Bachelor of Technology/Master of Technology in School of Engineering

and Sciences.

**Supervisor**

(Signature)

Prof. [ Karnena Kavitha Rani]

Designation-Coding Trainer

Affiliation-CSC

**Hospital Management System**

**BACKGROUND:**

* A hospital database management system (DBMS) is a powerful tool for managing hospital data efficiently and effectively.
* It provides a centralized repository for storing all hospital data, including patient information, medical records, doctor and staff details, medical equipment, and financial transactions.
* The hospital DBMS is designed to streamline hospital operations, improve patient outcomes, and enhance the overall patient experience.
* It consists of various components, including the database, database management software, and database applications.
* The database is the core of the system, storing all hospital data in a structured and organized manner.
* The database management software is responsible for managing the database, including creating, modifying, and deleting data, as well as managing user access and permissions.
* The database applications are the user interfaces that allow authorized users to interact with the system, such as entering and retrieving data.
* A hospital DBMS can manage large volumes of data from various sources, including electronic health records (EHRs), medical devices, and financial systems.
* It provides robust security features to ensure the confidentiality, integrity, and availability of patient data.
* It can improve hospital operations by automating various administrative tasks, such as scheduling appointments, managing patient billing, and tracking medical equipment.
* The system provides a user-friendly interface for authorized users to interact with the system, as well as robust security features to protect patient data.
* Overall, the hospital DBMS is a critical tool for managing hospital data and improving patient care.

**DESCRIPTION OF PROJECT:**

* The project on Hospital Management System is designed to optimize healthcare administration and enhance patient care delivery. The system comprises modules for Doctor Management, Patient Management, Patient Treatment, Room Assignments. It encompasses various entities crucial to the hospital managing domain.

Doctor Management facilitates efficient tracking and record-keeping through and essential attributes. Similarly, Patient Management ensures streamlined handling of patient data.

The Billing System transparently generates invoices for medical services and accommodation, enhancing financial transparency within the healthcare facility.

Room Assignments are made based on medical needs, ensuring seamless coordination and enhancing patient comfort throughout their stay. The system also includes comprehensive patient records, appointment scheduling.

In this management system we have:

Each ***DOCTOR*** has a d\_id, doc\_name, area\_of\_treatment, doctor\_charges.

Each ***PATIENT*** has a p\_id, name, phone\_number, gender,age,email,area\_of\_treatment,slot,doctor\_id .

Each ***BILL*** has a bill\_num, room\_charges,p\_id,name,area\_of\_treatment,age,doctor\_ charges, total\_charges.

Each ***ROOM*** has a room\_id,room\_charges.

Each doctor can treat multiple patients, but a patient is primarily treated by one doctor.

Each patient can incur multiple bills for their treatment.

Each bill is associated with one patient.

Each patient is assigned to one room during their stay.

**ER DIAGRAM:**

i

Room

Room\_charges

Room\_Id

BILL

Bill\_num

Total\_charges

PATIENT

Slot

Age,gender

Name

P\_id

area\_of\_treatment

phone\_number

DOCTOR

Doctor\_charges

area\_of\_treatment

doc\_name

d\_id

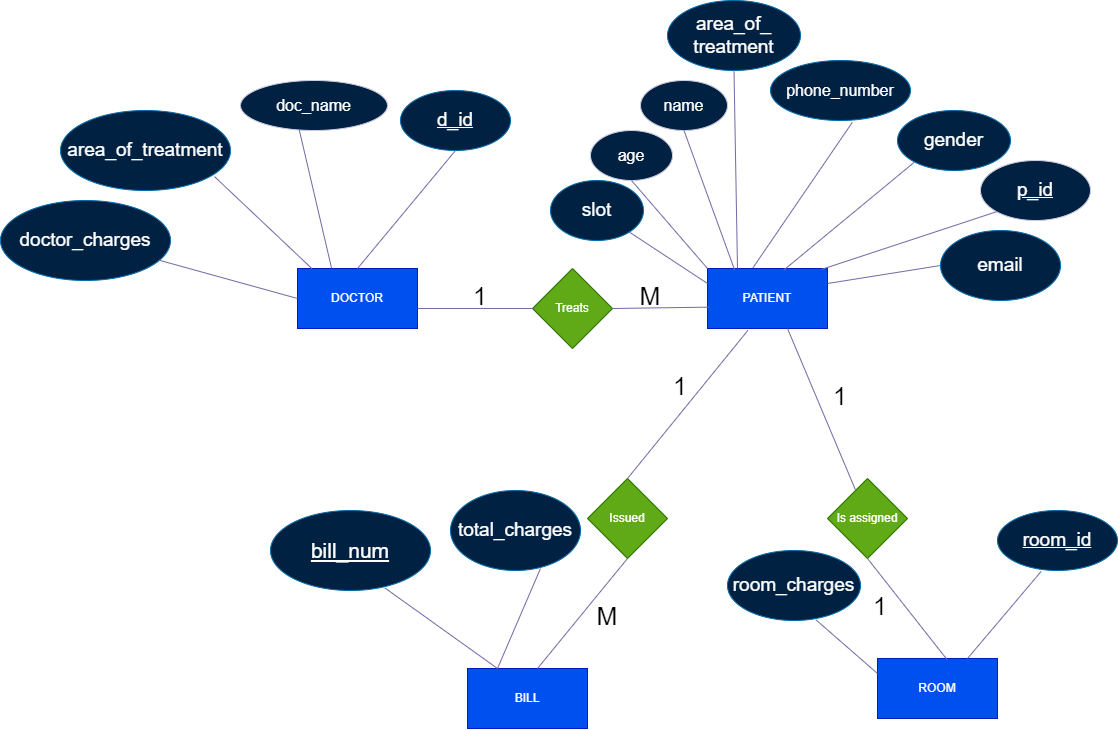
treat

Is assigned

bill

💡

Letʼs have a look on our er diagram:



* **DESCRIPTION OF ER DIAGRAM:**

In an Entity-Relationship (ER) model, symbols are graphical representations used to represent various components of the model. These symbols help visualize the structure of the data model and the relationships between different entities. Here are some common symbols used in ER diagrams:

**1. Entity:** An entity is represented by a rectangle. It represents a real-world object or concept, such as a person, place, thing, or event. The entity's name is written inside the rectangle.

**2. Attribute**: An attribute is represented by an oval or ellipse connected to its respective entity. It represents a property or characteristic of an entity. The attribute's name is written inside the oval.

**3. Primary Key:** The primary key is an attribute (or combination of attributes) that uniquely identifies each record in an entity. It is underlined within the attribute oval.

**4. Relationship:** A relationship is represented by a diamond shape connecting two or more entities. It indicates how entities are related to each other. The type of relationship (such as one-to-one, one-to-many, or many-to-many) is often labeled near the diamond.

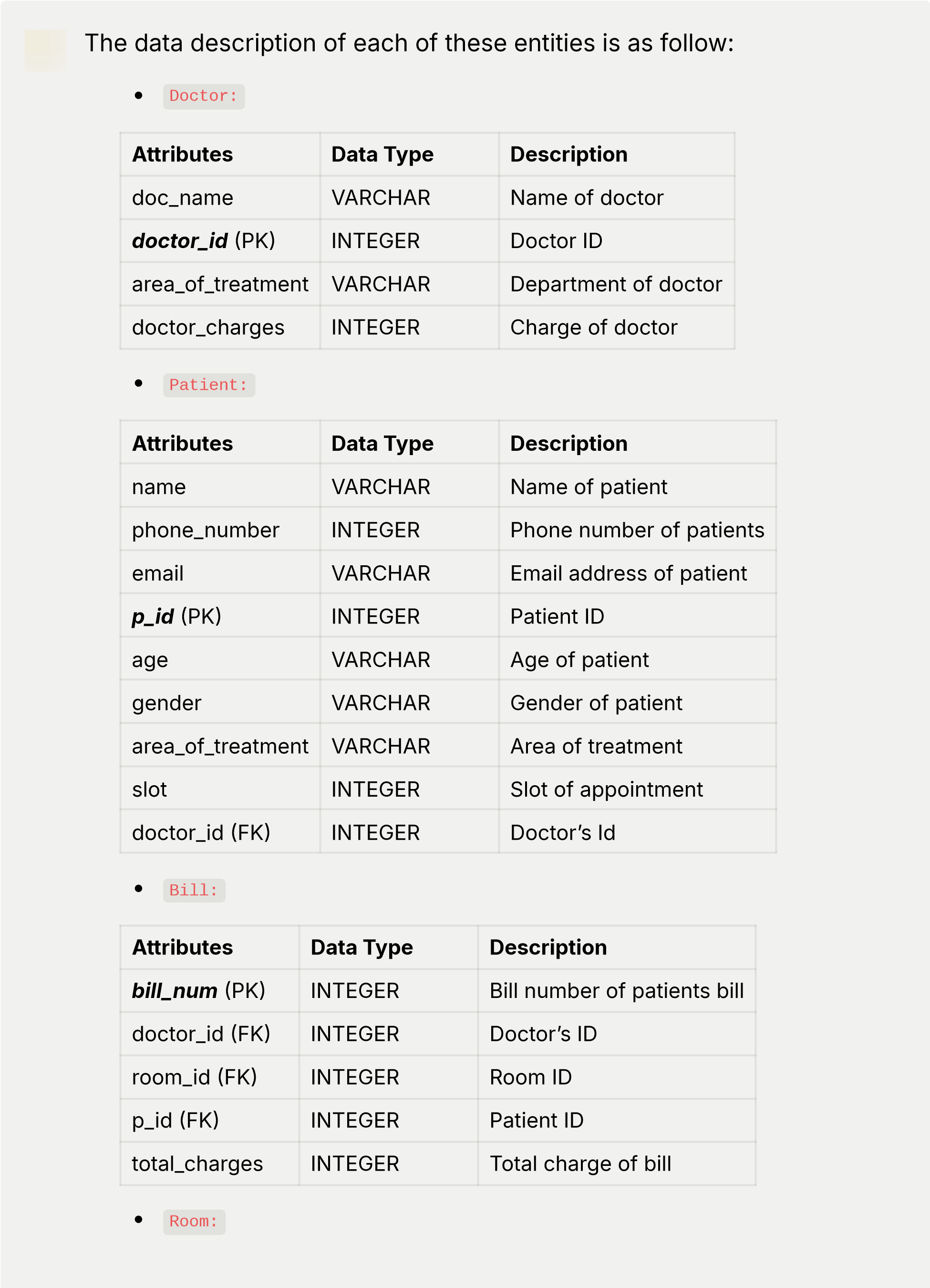
**5. Cardinality:** Cardinality describes the numerical relationship between entities in a relationship. It is represented using symbols near the relationship lines, such as "1" (for one), "M" (for many), or a range (e.g., "1..N").

**6. Weak Entity:** A weak entity is an entity that depends on another entity (called the owner entity) for its existence. It is represented by a double-bordered rectangle.

**7. Derived Attribute:** A derived attribute is an attribute whose value can be calculated from other attributes. It is represented by a dashed oval connected to its derived-from attributes.

**8. Recursive Relationship:** A recursive relationship occurs when an entity is related to itself. It is represented similarly to other relationships but with the same entity on both ends of the relationship line.

These symbols provide a standardized way to represent the structure and relationships within an ER model, making it easier for stakeholders to understand the data model's design and constraints.



**Attributes**

**Data Type**

**Description**

***room\_id***

(PK)

INTEGER

ID of room assigned to patient

room\_charges

INTEGER

Room charges

p\_id (FK)

INTEGER

Patientʼs ID

**CONVERSION OF ER DIAGRAMS INTO TABLES:**

**1.Strong Entities:**

💡

These are the strong entities:

Doctor

***d\_id***

doc\_name

area\_of\_treatment

doctor\_charges

Patient

***p\_id***

phone\_number

email

name

age

gender

area\_of\_treatment

slot

Bill

***bill\_num***

total\_charges

Room

***room\_id***

room\_charges

**2.Weak Entities:**

|  |
| --- |
| None |

**3.Mapping Cardinalities:** 💡 **Mapping of One-to-One Relationship Types:**

**PATIENT** *is assigned* **ROOM** (each patient is assigned one room) **Mapping of One-to-Many Relationship Types:**

**DOCTOR** *treats* **PATIENT** (one doctor can treat many patients, but each patient is treated by one doctor)

**PATIENT** *issued* **BILL** (one patient can issue multiple bills, but each bill is issued by one patient)

**4.Mapping multivalued attributes:**

|  |
| --- |
| No multivalued attributes |

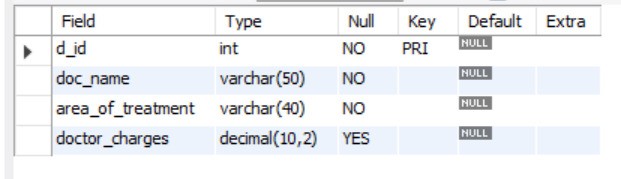
**5.Mapping N-array Entities:**

|  |
| --- |
| No N-array Entities |

**Description of Tables:**

💡

Doctor:



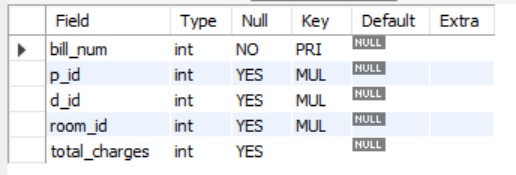
💡

Patient:



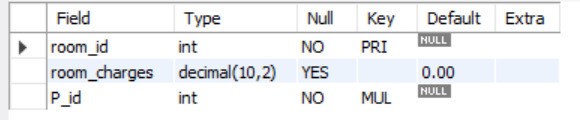
💡

Bill:



💡

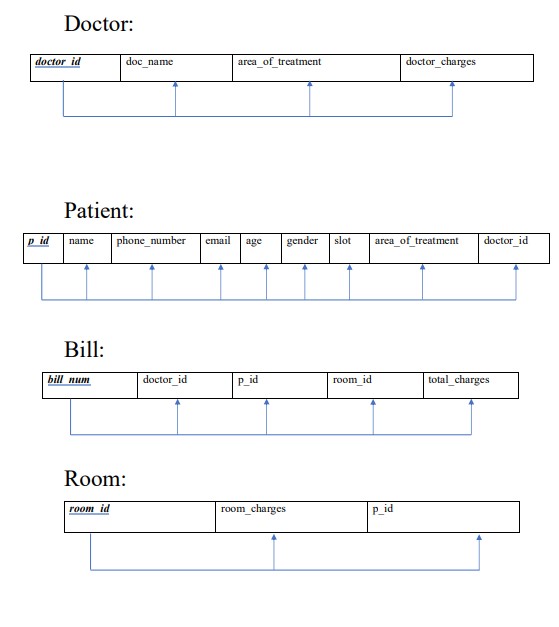
Room:



**NORMALIZATION OF TABLES :**

💡

Here, we have the FDs:



💡 First Normal Form (1NF):

* + All attributes within the database contain atomic values, meaning they are indivisible.
  + For instance, attributes like doc\_name, area\_of\_treatment, doctor\_charges in the Doctor table, and attributes like name, phone\_number, email in the Patient table hold singular, non-divisible values.
  + This satisfies the requirement of 1NF as each attribute contains only atomic values without any multi-valued or composite attributes.

Second Normal Form (2NF):

* Each table has a primary key, ensuring uniqueness and identifying each record uniquely.
* All non-key attributes are fully functionally dependent on the entire primary key.
* For example, in the Doctor table, attributes like doc\_name, area\_of\_treatment, doctor\_charges depend entirely on the primary key d\_id.
* This satisfies the requirement of 2NF as each non-key attribute is dependent on the entire primary key, eliminating partial dependencies.

Third Normal Form (3NF):

* No transitive dependencies exist in the database.
* Each non-key attribute directly depends on the primary key without any indirect dependencies on other non-key attributes.
* For instance, in the Doctor table, attributes like doc\_name and area\_of\_treatment directly depend on the primary key d\_id, without depending on each other.
* This satisfies the requirement of 3NF as there are no transitive dependencies present in the database.

**SQL QUERIES:**

CREATE DATABASE Hospital\_management\_system;

USE Hospital\_management\_system;

CREATE TABLE Doctor( d\_id INT PRIMARY KEY, doc\_name VARCHAR(50) NOT NULL, area\_of\_treatment VARCHAR(40) NOT NULL, doctor\_charges DECIMAL(10,2)); DESC Doctor;

CREATE TABLE Patient( p\_id INT PRIMARY KEY, name VARCHAR(50) NOT NULL, phone\_number INT NOT NULL UNIQUE, gender ENUM('Male', 'Female'), age INT NOT NULL, email VARCHAR(50) NOT NULL UNIQUE, area\_of\_treatment VARCHAR(50) NOT NULL, slot VARCHAR(20) NOT NULL, d\_id INT NOT NULL,

FOREIGN KEY(d\_id) REFERENCES Doctor(d\_id));

DESC Patient;

CREATE TABLE Room( room\_id INT PRIMARY KEY, room\_charges DECIMAL(10,2) DEFAULT 0.0,

P\_id INT NOT NULL,

FOREIGN KEY (p\_id) REFERENCES Patient(p\_id));

DESC Room;

CREATE TABLE Bill( bill\_num INT PRIMARY KEY,

p\_id INT, d\_id INT, room\_id INT, total\_charges INT,

FOREIGN KEY (p\_id) REFERENCES Patient(p\_id),

FOREIGN KEY (d\_id) REFERENCES Doctor(d\_id),

FOREIGN KEY (room\_id) REFERENCES Room(room\_id)); DESC Bill;

INSERT INTO Doctor(d\_id,doc\_name,area\_of\_treatment,doctor\_charges)

VALUES (1,"Dr.Drisana","Pediatrics",1500.00),

(2,"Dr.John Doe","Cardiology",2000.00),

(3,"Dr.Olivia Smith","Gynaecology",2500.00),

(4,"Dr.Jane Smith","Neurology",1800.00);

ALTER TABLE Patient MODIFY COLUMN phone\_number BIGINT NOT NULL;

INSERT INTO Patient (p\_id, name, phone\_number, gender, age, email, area\_of\_treatment, slot, d\_id)

VALUES

(101, 'Jack Brown', 1234567890, 'Male', 35, 'jack@example.com', 'Cardiology', 'Morning', 2),

(102, 'Bob Johnson', 9876543280, 'Male', 45, 'bob@example.com', 'Neurology', 'Afternoon', 4),

(103, 'Charlie Brown', 5551234567, 'Male', 28, 'charlie@example.com', 'Pediatrics', 'Morning', 1),

(104, 'Diana Lee', 1112223333, 'Female', 50, 'diana@example.com', 'Neurology', 'Evening', 4),

(105, 'Ella Chen', 9998887777, 'Female', 40, 'ella@example.com', 'Gynaecology', 'Afternoon', 3),

(106, 'Frank Wilson', 7779998888, 'Male', 55, 'frank@example.com', 'Cardiology', 'Evening', 2),

(107, 'Grace Taylor', 2223334444, 'Female', 30, 'grace@example.com',

'Pediatrics', 'Afternoon', 1),

(108, 'Ivy White', 8887776666, 'Female', 25, 'ivy@example.com', 'Gynaecology', 'Evening', 3);

INSERT INTO Room (room\_id, p\_id, room\_charges)

VALUES

(1001, 101, 0.00),

(1002, 102, 300.00),

(1003, 103, 0.00),

(1004, 104, 400.00),

(1005, 105, 0.00),

(1006, 106, 450.00),

(1007, 107, 350.00),

(1008, 108, 450.00);

INSERT INTO Bill (bill\_num,p\_id, d\_id, room\_id, total\_charges)

SELECT

12000 ( ROW\_NUMBER() OVER () AS bill\_num,

Patient.p\_id,

Doctor.d\_id,

Room.room\_id,

(Doctor.doctor\_charges +Room.room\_charges) AS total\_charges

FROM

Patient

JOIN

Room ON Patient.p\_id = Room.p\_id

JOIN

Doctor ON Patient.d\_id =Doctor.d\_id;

select \* from Bill;

/\*Retrieve the details of patients along with the room they are assigned to, the name of the doctor treating them, and the corresponding doctor's charges.\*

SELECT

[Patient.name](http://patient.name/) AS patient\_name,

Room.room\_id,

Doctor.doc\_name AS doctor\_name,

Doctor.doctor\_charges

FROM

Patient

INNER JOIN

Room ON Patient.p\_id = Room.p\_id

INNER JOIN

Doctor ON Patient.d\_id = Doctor.d\_id;

/\*Retrieve the area of treatment along with the maximum, minimum, and total charges for doctors in each area. Round the charges to two decimal places. Show only those areas where the total charges for doctors exceed $1000. Sort the results by the total charges in descending order.\*/

SELECT area\_of\_treatment,

ROUND(MAX(doctor\_charges), 2) AS max\_charges,

ROUND(MIN(doctor\_charges), 2) AS min\_charges,

ROUND(SUM(doctor\_charges), 2) AS total\_charges

FROM Doctor

GROUP BY area\_of\_treatment

HAVING total\_charges > 1000

ORDER BY total\_charges DESC;

/\*Retrieve the average age of patients treated by each doctor.\*/

SELECT

Doctor.doc\_name AS doctor\_name,

AVGPatient.age) AS average\_patient\_age

FROM

Patient

INNER JOIN

Doctor ON Patient.d\_id =Doctor.d\_id

GROUP BY

Doctor.doc\_name;

/\*Retrieve the details of patients whose names contain the letter 'a' followed by exactly two characters, regardless of case.\*

SELECT

\*

FROM

Patient

WHERE

name LIKE '%a\_%' OR name LIKE '%A\_%';

/\*Create a trigger that automatically calculates the total charges for a bill whenever a new bill is inserted.\*

DELIMITER //

CREATE TRIGGER calculate\_total\_charges

AFTER INSERT ON Bill

FOR EACH ROW

BEGIN

UPDATE Bill

SET total\_charges = (SELECT doctor\_charges FROM Doctor WHERE d\_id =

NEW.d\_id) + (SELECT room\_charges FROM Room WHERE room\_id =

NEW.room\_id)

WHERE bill\_num = NEW.bill\_num;

END//

DELIMITER ;

Select \* from Bill;

**CREATION OF VIEWS**

**This view will provide details about rooms and the bills associated with them.**

CREATE VIEW Room\_Bill\_Info\_View AS

SELECT

r.room\_id AS Room\_ID,

r.room\_charges AS Room\_Charges,

[p.name](http://p.name/) AS Patient\_Name,

b.bill\_num AS Bill\_Number,

b.total\_charges AS Total\_Charges

FROM

Room r

JOIN

Patient p ON r.p\_id = p.p\_id

JOIN

Bill b ON r.room\_id = b.room\_id;

SELECT \* FROM Room\_Bill\_Info\_View;

**This view will provide a detailed summary of bills, including information about the patient, doctor, and room charges.**

CREATE VIEW Detailed\_Bill\_View AS

SELECT

b.bill\_num AS Bill\_Number,

[p.name](http://p.name/) AS Patient\_Name,

d.doc\_name AS Doctor\_Name,

r.room\_id AS Room\_ID,

b.total\_charges AS Total\_Charges

FROM

Bill b

JOIN

Patient p ON b.p\_id = p.p\_id JOIN

Doctor d ON b.d\_id = d.d\_id

JOIN

Room r ON b.room\_id = r.room\_id;

SELECT \* FROM Detailed\_Bill\_View;

**This view will provide details about Doctor performance view.**

CREATE VIEW DoctorPerformance AS

SELECT DoctorID, Name, Specialty, COUNT(AppointmentID) AS ConsultationCount, AVG(Duration) AS AverageConsultationDuration

FROM Doctors

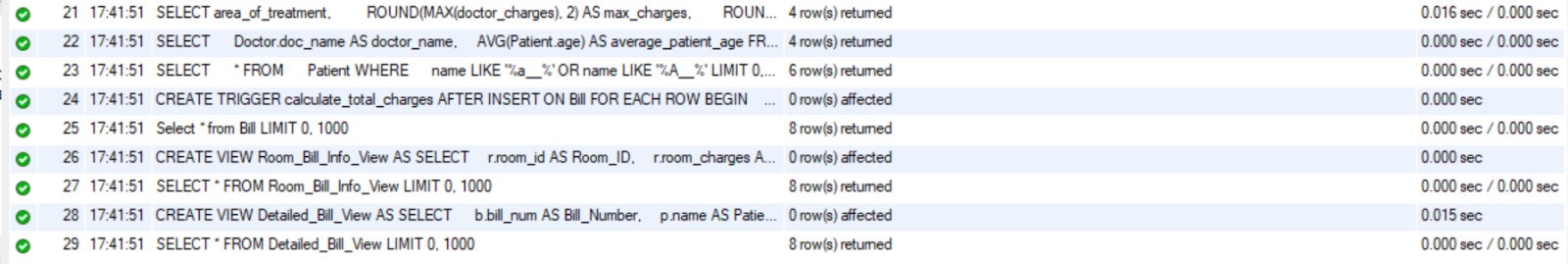
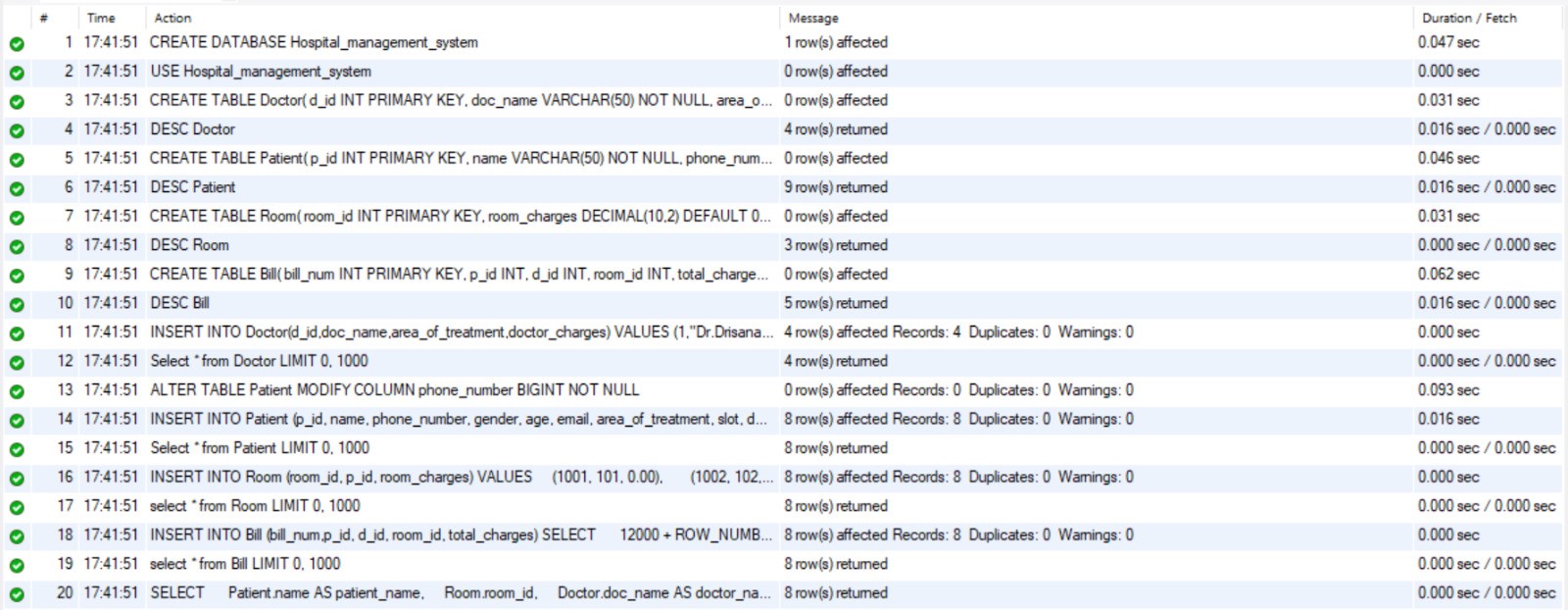
LEFT JOIN Appointments ON Doctors.DoctorID = Appointments.DoctorID

GROUP BY DoctorID;

**CREATION OF DATA IN THE TABLES:**

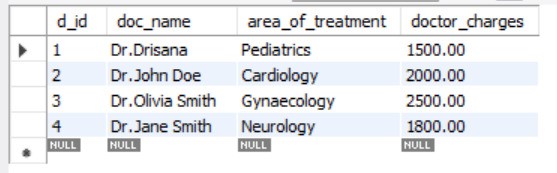
💡

Action Output:



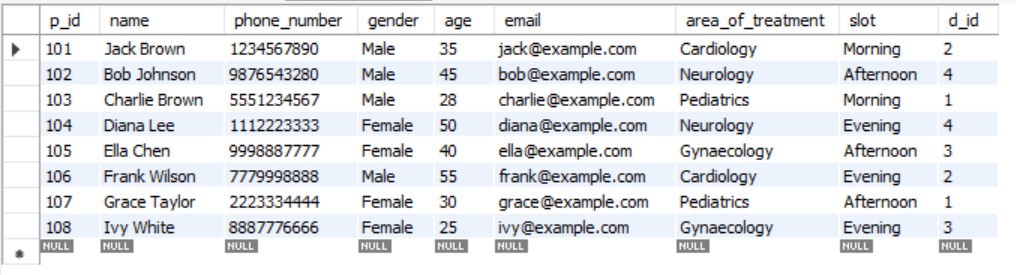
💡

Inserting doctor details:



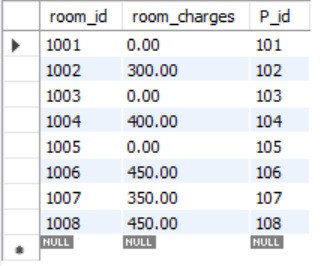
💡

Inserting patient details:



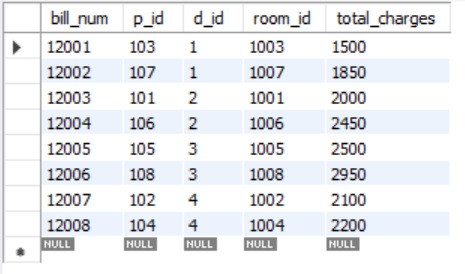
💡

Inserting room details:



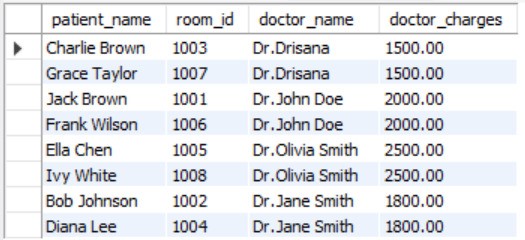
💡

Inserting bill details:



💡

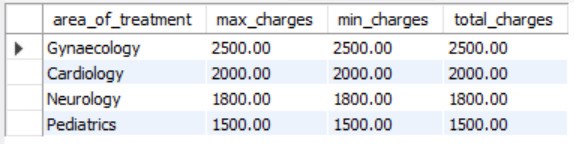
Retrieving patients, room and doctor details:



💡

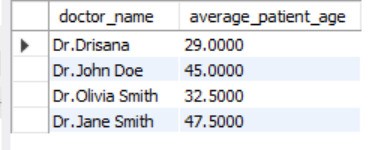
Retrieving minimum, maximum and total charges for each area of

treatment:



💡

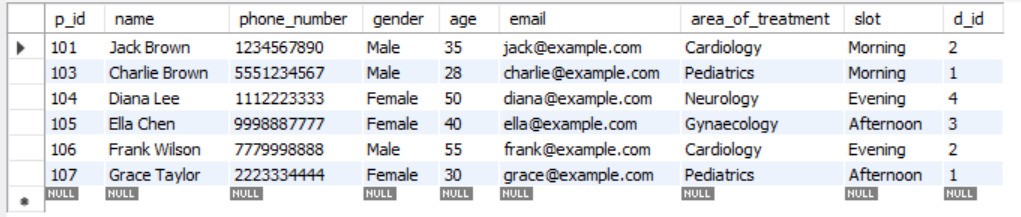
Retrieving average age of patient:



💡

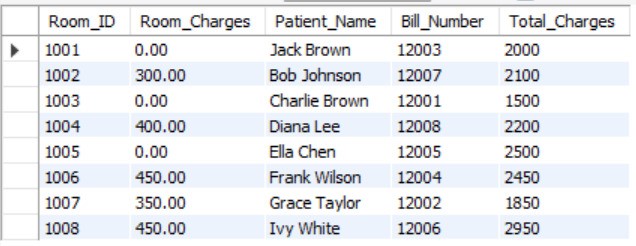
Retrieving patient details containing letter ‘aʼ followed by exactly 2

characters:



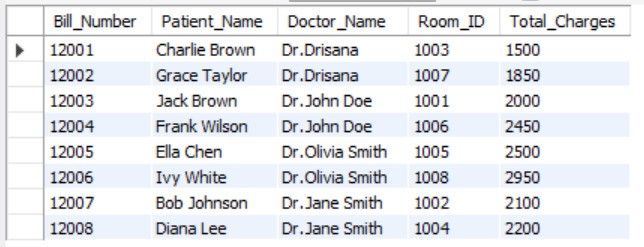
💡

Room\_Bill\_Info\_View:



💡

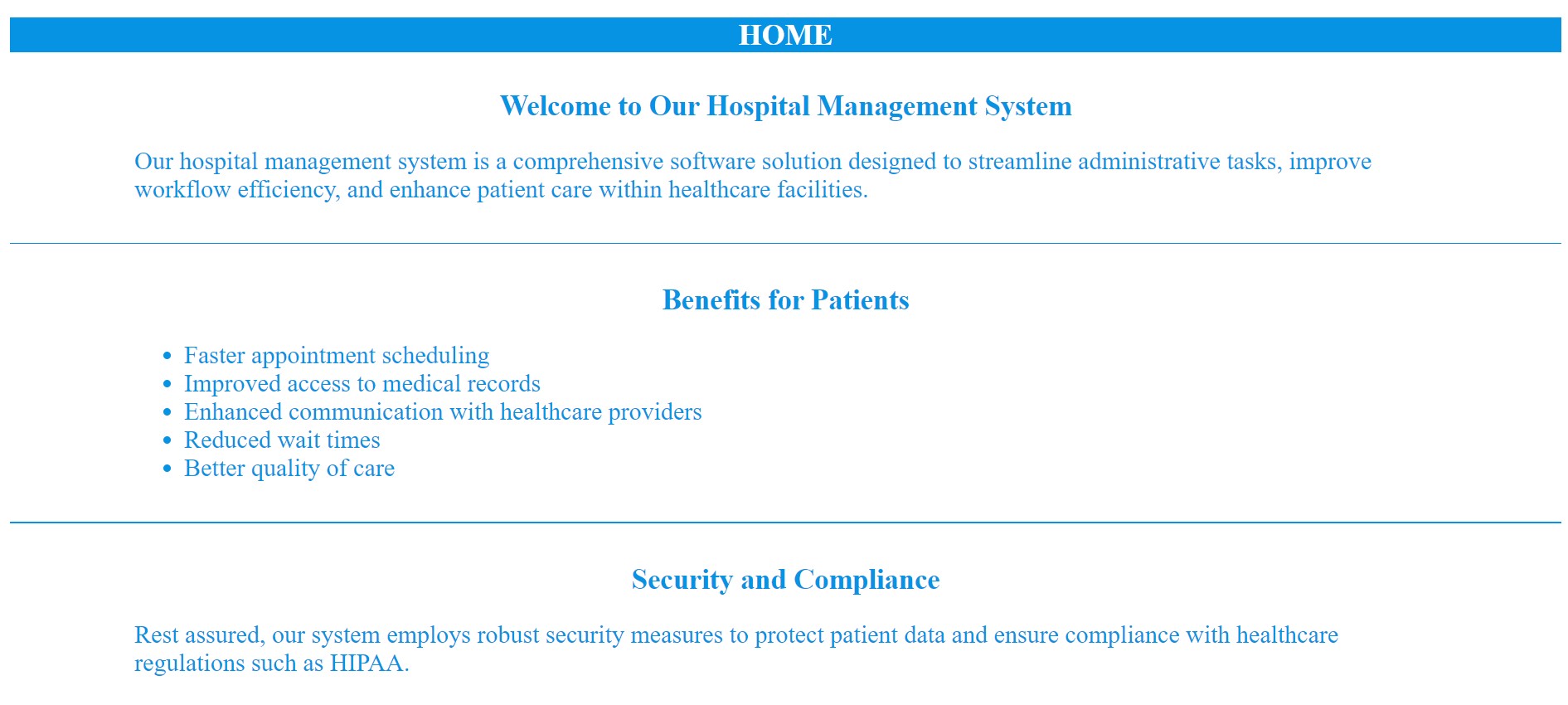
Detailed\_Bill\_View:



**CareMaster Expert (Front-end Website):**

💡

Letʼs have a look:



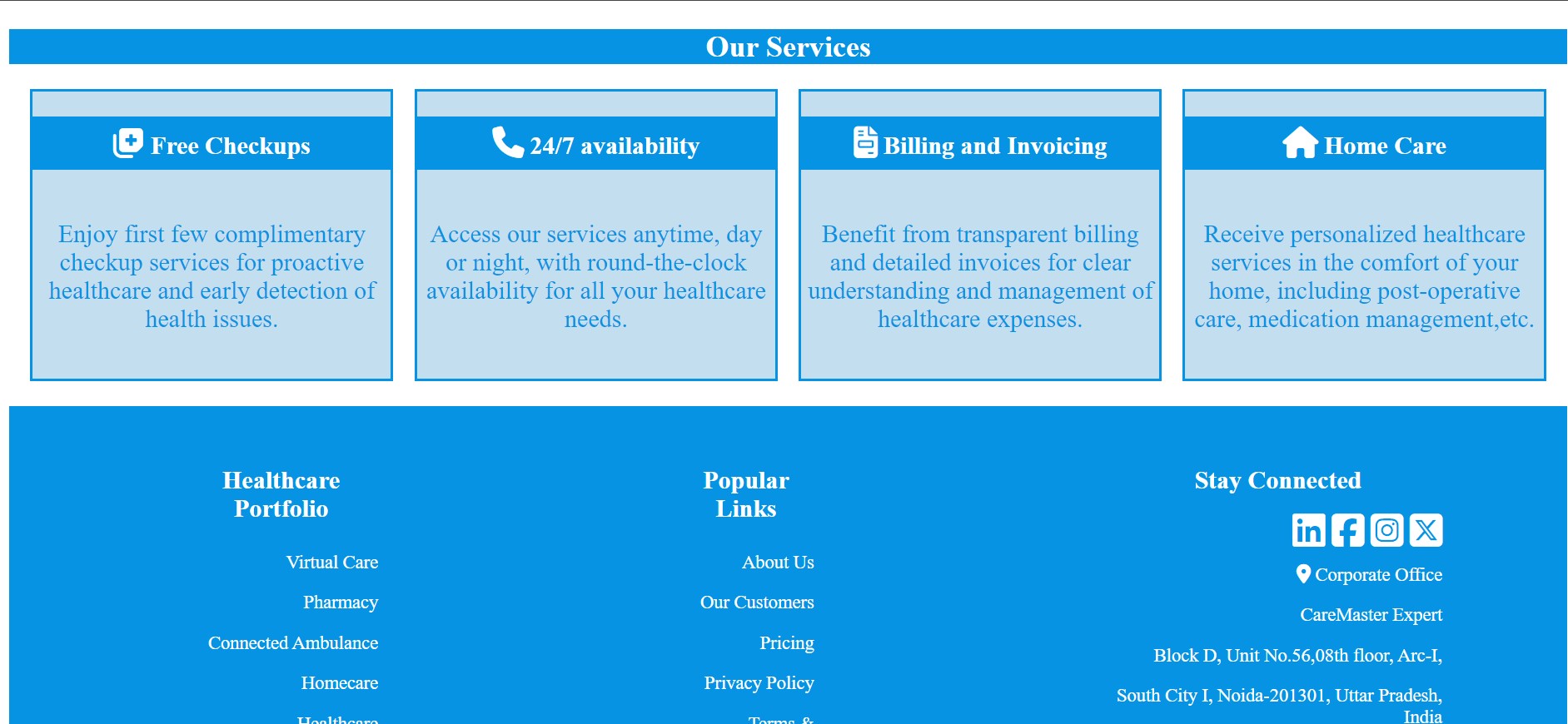
💡

About us:



💡

BOOK YOUR SLOT



**Conclusion:**

In conclusion, the Hospital Management System project offers a comprehensive solution to optimize healthcare administration and elevate patient care delivery. Through modules like Doctor and Patient Management, Billing, and Room Assignments, the system ensures efficient tracking and handling of crucial healthcare processes.

By integrating a frontend website for appointment booking and bill generation, the project enhances accessibility and transparency for patients. The Billing System promotes financial transparency by generating clear invoices, while Room Assignments prioritize patient comfort by allocating rooms based on medical needs.

Overall, the project demonstrates the power of technology in addressing healthcare complexities while prioritizing patient welfare. With its robust database design and user-friendly interface, the system serves as a valuable tool for healthcare facilities seeking to improve operational efficiency and enhance the quality of care provided.