

[HEALTHCARE PORTAL]

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) G.Vidhathri(AP22110010196)
- 2) Y.Sindhu(AP22110010180)
- 3) K.Darvini Latha(AP22110010167)
- 4)N.Poojitha Sai Priya(AP22110010187)
- 5)G.Oshitha(AP22110010174)



SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

[MAY, 2024]

1. Identification of Project related to DBMS project

Project Title:Healthcare Portal Database Management System (HP DBMS)

2. Project Background

The Healthcare Portal Database Management System(HP DBMS) aims to create an integrated platform for managing various aspects of healthcare services.

This system will facilitate efficient management of patient records, appointments, medical history, healthcare provider information, billing, and administrative tasks within the healthcare organization.

Designing and implementing a comprehensive Healthcare Portal Database Management System (HP DBMS) to address the challenges faced by healthcare organizations.

Providing a user-friendly interface for patients, healthcare providers, and administrative staff to access and manage relevant information securely.

Automating appointment scheduling, medical record management, billing, and reporting to improve operational efficiency and reduce administrative burden.

Enhancing patient engagement, communication, and satisfaction through online services and real-time information access.

Ensuring data security, privacy, and regulatory compliance within the healthcare portal.

3. Description of the Project

The Healthcare Portal Database Management System (HP DBMS) is a comprehensive digital solution designed to streamline and enhance various aspects of healthcare management.

Here's a detailed description of the project:

Overview:

The Healthcare Portal Database Management System (HP DBMS) is a web-based platform that integrates a robust database management system with user-friendly interfaces for patients, healthcare providers, and administrative staff. It aims to digitize and automate key healthcare processes to improve efficiency, communication, and patient care.

Key Features:

- **Patient Management:**
 - Patients can register, update their profiles, and access personal health records securely.

- Appointment scheduling allows patients to book, reschedule, or cancel appointments online.
- Automated reminders and notifications keep patients informed about upcoming appointments and follow-ups.
- **Healthcare Provider Tools:**
 - Healthcare providers (doctors, nurses, specialists) have access to patient records, medical histories, and treatment plans.
 - Provider schedules and availability are managed efficiently to avoid conflicts and optimize patient care.
 - Communication tools facilitate collaboration among healthcare teams and improve patient outcomes.
- **Medical Records Management:**
 - Electronic medical records (EMRs) store comprehensive patient data, including diagnoses, treatments, medications, and lab results.
 - Secure access controls ensure that only authorized personnel can view, edit, or update medical records.
 - Integration with diagnostic systems and health information exchanges enhances data accuracy and accessibility.
- **Appointment and Billing Integration:**
 - Appointment scheduling is integrated with patient records, provider schedules, and availability.
 - Billing and invoicing functionalities streamline payment processes, insurance claims, and financial reporting.
 - Transparent billing statements and online payment options improve patient satisfaction and revenue management.
- **Administrative Tools:**
 - Administrators have tools for managing user accounts, roles, permissions, and system configurations.
 - Reporting and analytics dashboards provide insights into patient demographics, appointment trends, billing cycles, and resource utilization.
 - Compliance features ensure adherence to healthcare regulations (e.g., HIPAA) and data security standards.

User Interfaces:

- **Patient Portal:** Patients can log in to view their health records, schedule appointments, communicate with providers, and access educational resources.
- **Provider Dashboard:** Healthcare providers have a centralized dashboard for managing patient care, appointments, prescriptions, and communication with colleagues.
- **Administrator Console:** Administrators can configure system settings, manage user accounts, generate reports, and monitor system performance.

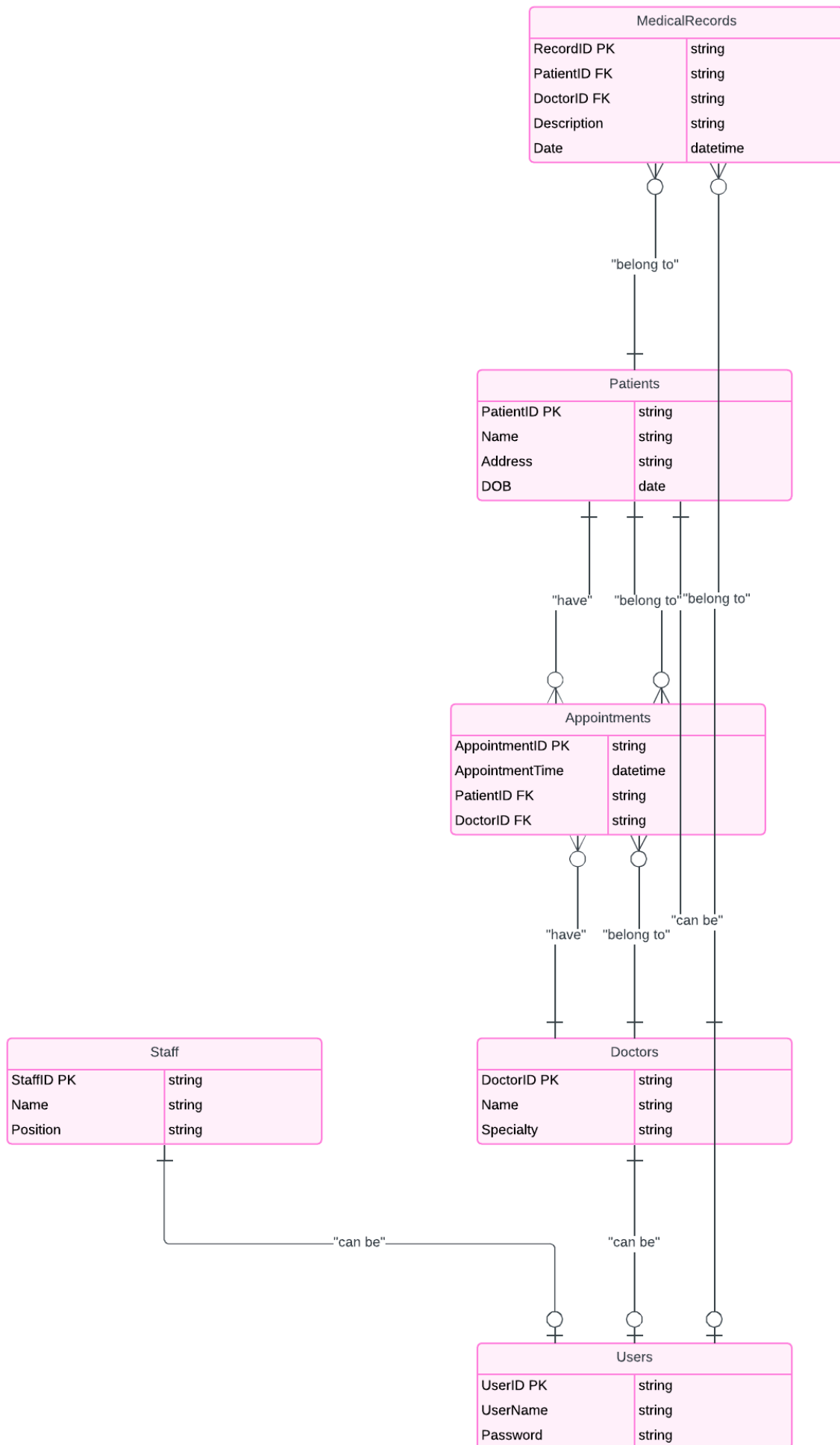
Benefits:

- **Improved Patient Experience:** Enhanced communication, convenience, and access to healthcare services.
- **Operational Efficiency:** Streamlined processes, reduced administrative tasks, and optimize resource utilization.
- **Data Accuracy and Compliance:** Centralised data management, secure access controls, and regulatory compliance.
- **Cost Savings:** Efficient appointment scheduling, billing processes, and resource allocation lead to cost savings and revenue growth.

Overall, the Healthcare Portal Database Management System (HP DBMS) empowers healthcare organizations to deliver high-quality patient care, improve operational workflows, and adapt to the digital transformation of the healthcare industry.

4. ER Diagram Creation

- Entities: Patients, Doctors, Appointments, MedicalRecords, Staff, Users
- Relationships:
 - Patients have Appointments (1:M)
 - Doctors have Appointments (1:M)
 - Appointments belong to Patients (M:1) and Doctors (M:1)
 - MedicalRecords belong to Patients (1:M) and Doctors (1:M)
 - Users can be Staff or Patients or Doctors (1:1 with Users)



5. Description of ER diagram

The Entity-Relationship (ER) diagram for the Healthcare Portal Database Management System (HP DBMS) represents the structure and relationships between various entities in the system. Here's a description of the entities, relationships, and cardinalities depicted in the ER diagram:

1. Entities:

- **Patients:** Represents individuals seeking healthcare services. Attributes include patient_id (primary key), name, email, phone, and address.
- **Doctors:** Represents healthcare providers serving patients. Attributes include doctor_id (primary key), name, specialization, email, and phone.
- **Appointments:** Represents scheduled appointments between patients and doctors. Attributes include appointment_id (primary key), appointment_date, and status (e.g., Scheduled, Canceled, Completed).
- **MedicalRecords:** Represents medical records associated with patient visits. Attributes include record_id (primary key), diagnosis, treatment, and date_created.
- **Staff:** Represents administrative staff or personnel. Attributes include staff_id (primary key), name, role, email, and phone.
- **Users:** Represents users of the system, including patients, doctors, and staff. Attributes include user_id (primary key), username, password, and role (Admin, Doctor, Patient).

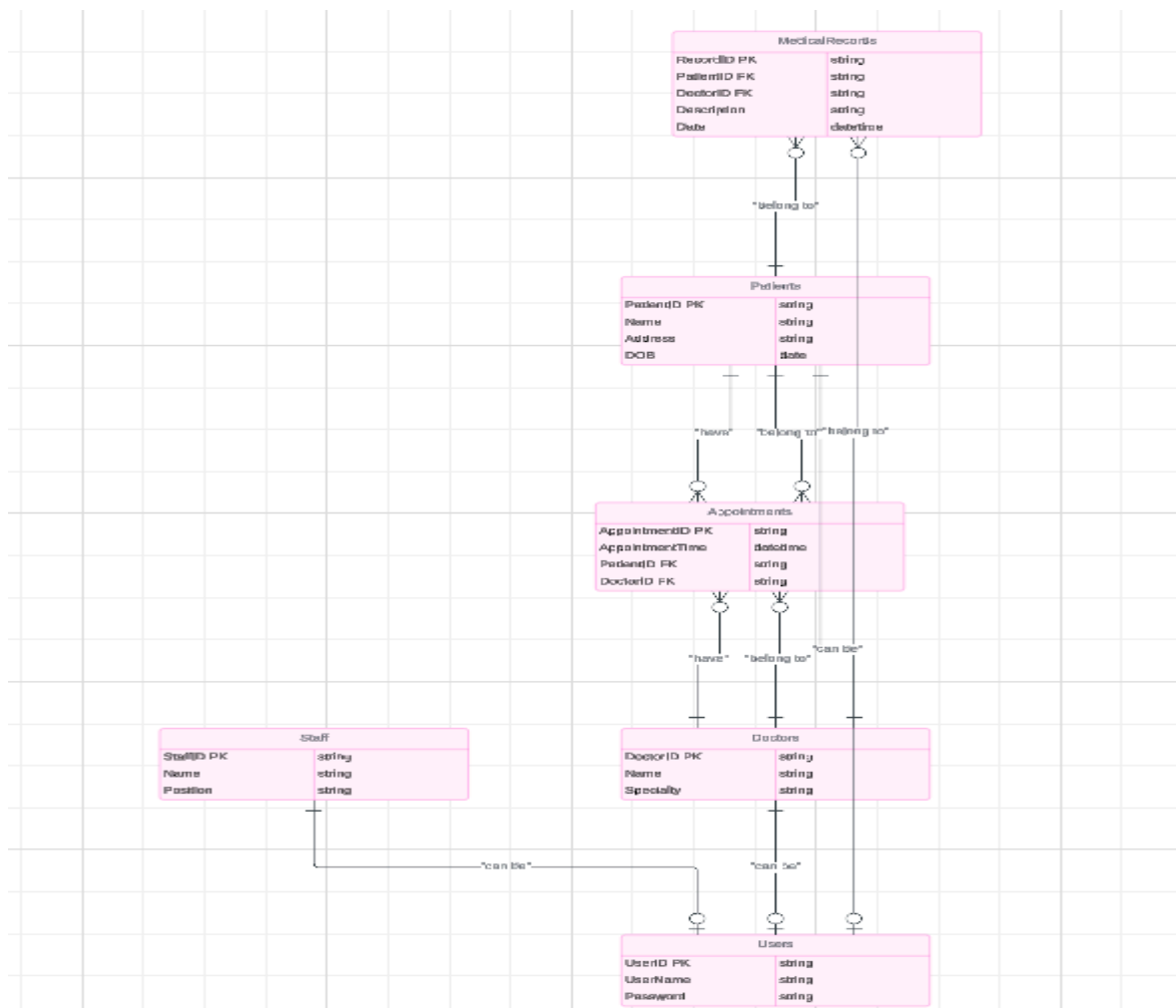
2. Relationships:

- **Patients - Appointments:** One patient can have multiple appointments (1:M relationship). Each appointment belongs to one patient (M:1 relationship).
- **Doctors - Appointments:** One doctor can have multiple appointments (1:M relationship). Each appointment belongs to one doctor (M:1 relationship).
- **Appointments - MedicalRecords:** Each appointment has one associated medical record (1:1 relationship). Each medical record belongs to one appointment (1:1 relationship).
- **Patients - MedicalRecords:** One patient can have multiple medical records (1:M relationship). Each medical record belongs to one patient (M:1 relationship).
- **Doctors - MedicalRecords:** One doctor can have multiple medical records (1:M relationship). Each medical record belongs to one doctor (M:1 relationship).
- **Users - Patients/Doctors/Staff:** Users can be associated with either Patients, Doctors, or Staff entities through a subtype/supertype relationship (1:1 relationship).

3. Cardinalities:

- **One-to-Many (1:M):** Represents relationships where one entity instance is associated with multiple instances of another entity. Example: One patient can have multiple appointments.
- **Many-to-One (M:1):** Represents relationships where multiple instances of an entity are associated with one instance of another entity. Example: Multiple appointments belong to one doctor.
- **One-to-One (1:1):** Represents relationships where one entity instance is associated with exactly one instance of another entity. Example: Each appointment has one associated medical record.

The ER diagram visually organizes these entities, attributes, relationships, and cardinalities, providing a clear understanding of how data is structured and interconnected within the Healthcare Portal Database Management System (HP DBMS).



6. Conversion of ER diagram into Tables

→ Patients Table:

◆ Attributes:

- patient_id (Primary Key)
- name
- email
- phone
- address

SQL code for Patients table creation

```
CREATE TABLE Patients (  
    patient_id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(255) NOT NULL,  
    email VARCHAR(255) UNIQUE,  
    phone VARCHAR(15),  
    address VARCHAR(255)  
);
```

→ Doctors Table:

Attributes:

- doctor_id (Primary Key)
- name
- specialization
- email
- phone

SQL code for Doctors table creation

```
CREATE TABLE Doctors (  
    doctor_id INT AUTO_INCREMENT PRIMARY KEY,
```



```
name VARCHAR(255) NOT NULL,  
specialization VARCHAR(255),  
email VARCHAR(255) UNIQUE,  
phone VARCHAR(15)  
);
```

→ Appointments Table:

Attributes:

- appointment_id (Primary Key)
- patient_id (Foreign Key)
- doctor_id (Foreign Key)
- appointment_date
- status

SQL code for Appointments table creation

```
CREATE TABLE Appointments (  
    appointment_id INT AUTO_INCREMENT PRIMARY KEY,  
    patient_id INT,  
    doctor_id INT,  
    appointment_date DATE,  
    status ENUM('Scheduled', 'Canceled', 'Completed'),  
    FOREIGN KEY (patient_id) REFERENCES Patients(patient_id),  
    FOREIGN KEY (doctor_id) REFERENCES Doctors(doctor_id)  
);
```

→ MedicalRecords Table:

Attributes:

- record_id (Primary Key)
- patient_id (Foreign Key)
- doctor_id (Foreign Key)

- diagnosis
- treatment
- date_created

SQL code for MedicalRecords table creation

```
CREATE TABLE MedicalRecords (
    record_id INT AUTO_INCREMENT PRIMARY KEY,
    patient_id INT,
    doctor_id INT,
    diagnosis TEXT,
    treatment TEXT,
    date_created TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (patient_id) REFERENCES Patients(patient_id),
    FOREIGN KEY (doctor_id) REFERENCES Doctors(doctor_id)
);
```

→ Staff Table:

Attributes:

- staff_id (Primary Key)
- name
- role
- email
- phone

SQL code for Staff table creation

```
CREATE TABLE Staff (
    staff_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    role VARCHAR(255),
```

email VARCHAR(255) UNIQUE,

phone VARCHAR(15)

);

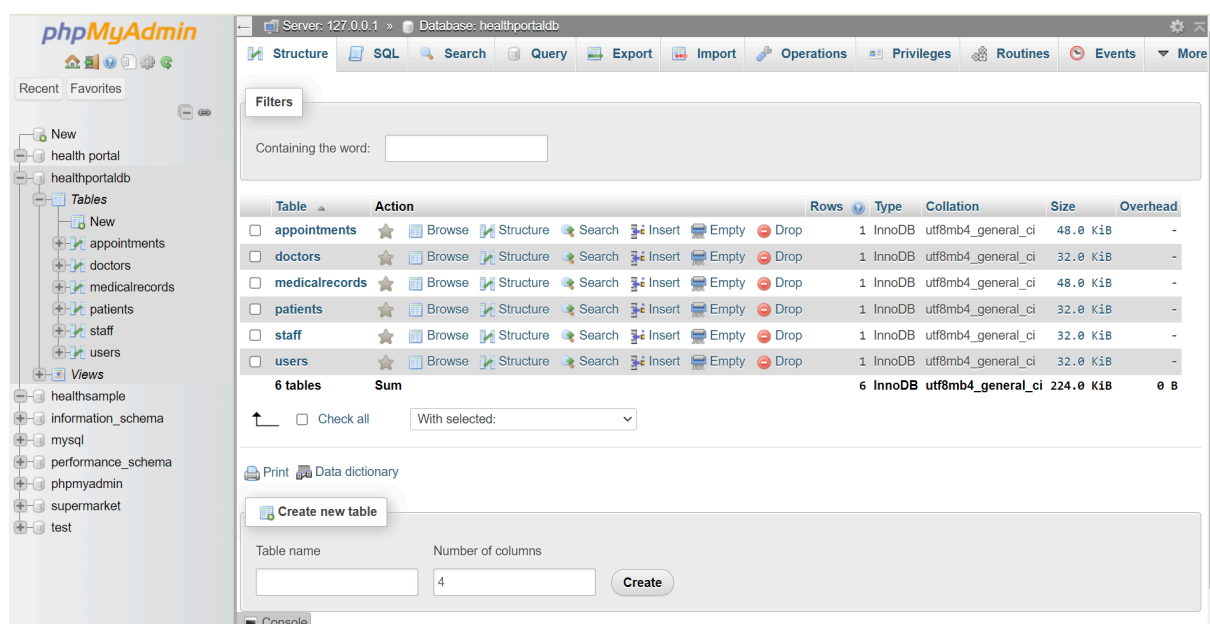
→ Users Table:

Attributes:

- user_id (Primary Key)
- username (Unique)
- password
- role (Enum: 'Admin', 'Doctor', 'Patient')

SQL code for Users table creation

```
CREATE TABLE Users (  
    user_id INT AUTO_INCREMENT PRIMARY KEY,  
    username VARCHAR(50) UNIQUE NOT NULL,  
    password VARCHAR(255) NOT NULL,  
    role ENUM('Admin', 'Doctor', 'Patient') NOT NULL  
);
```



7. Description of Tables

1. Patients Table:

- This table stores information about patients registered in the healthcare portal.
- **Attributes:**
 - **patient_id:** Unique identifier for each patient (Primary Key).
 - **name:** Full name of the patient.
 - **email:** Email address of the patient (Unique).
 - **phone:** Contact phone number of the patient.
 - **address:** Residential or mailing address of the patient.

2. Doctors Table:

- This table contains details of healthcare providers, such as doctors, within the system.
- **Attributes:**
 - **doctor_id:** Unique identifier for each doctor (Primary Key).
 - **name:** Full name of the doctor.
 - **specialization:** Area of specialization or medical expertise.
 - **email:** Email address of the doctor (Unique).
 - **phone:** Contact phone number of the doctor.

3. Appointments Table:

- This table manages appointments scheduled between patients and doctors.
- **Attributes:**
 - **appointment_id:** Unique identifier for each appointment (Primary Key).
 - **patient_id:** Foreign key referencing the patient_id in the Patients table.
 - **doctor_id:** Foreign key referencing the doctor_id in the Doctors table.
 - **appointment_date:** Date of the appointment.
 - **status:** Status of the appointment (e.g., Scheduled, Canceled, Completed).

4. MedicalRecords Table:

- This table stores medical records associated with patient visits and treatments.
- **Attributes:**
 - **record_id:** Unique identifier for each medical record (Primary Key).
 - **patient_id:** Foreign key referencing the patient_id in the Patients table.
 - **doctor_id:** Foreign key referencing the doctor_id in the Doctors table.
 - **diagnosis:** Diagnosis or medical condition identified during the visit.
 - **treatment:** Treatment plan or actions prescribed for the patient.

- **date_created:** Timestamp indicating when the medical record was created.

5. Staff Table:

- This table contains information about administrative staff or personnel involved in the healthcare organization.
- **Attributes:**
 - **staff_id:** Unique identifier for each staff member (Primary Key).
 - **name:** Full name of the staff member.
 - **role:** Role or position within the healthcare organization.
 - **email:** Email address of the staff member (Unique).
 - **phone:** Contact phone number of the staff member.

6. Users Table:

- This table represents users of the system, including patients, doctors, and administrative staff.
- **Attributes:**
 - **user_id:** Unique identifier for each user (Primary Key).
 - **username:** Username used for login (Unique).
 - **password:** Encrypted password for user authentication.
 - **role:** Role of the user in the system (Admin, Doctor, Patient).

These tables collectively form the foundation of the Healthcare Portal Database Management System (HP DBMS), enabling efficient management of patient data, appointments, medical records, staff information, and user access within the healthcare organization.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	user_id 🔑	int(11)			No	None		AUTO_INCREMENT
2	username 🔑	varchar(50)	utf8mb4_general_ci		No	None		
3	password	varchar(255)	utf8mb4_general_ci		No	None		
4	role	enum('Admin', 'Doctor', 'Patient')	utf8mb4_general_ci		No	None		
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	staff_id 🔑	int(11)			No	None		AUTO_INCREMENT
2	name	varchar(255)	utf8mb4_general_ci		No	None		
3	role	varchar(255)	utf8mb4_general_ci		Yes	NULL		
4	email 🔑	varchar(255)	utf8mb4_general_ci		Yes	NULL		
5	phone	varchar(15)	utf8mb4_general_ci		Yes	NULL		

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	patient_id	int(11)			No	None		AUTO_INCREMENT
2	name	varchar(255)	utf8mb4_general_ci		No	None		
3	email	varchar(255)	utf8mb4_general_ci		Yes	NULL		
4	phone	varchar(15)	utf8mb4_general_ci		Yes	NULL		
5	address	varchar(255)	utf8mb4_general_ci		Yes	NULL		

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	record_id	int(11)			No	None		AUTO_INCREMENT
2	patient_id	int(11)			Yes	NULL		
3	doctor_id	int(11)			Yes	NULL		
4	diagnosis	text	utf8mb4_general_ci		Yes	NULL		
5	treatment	text	utf8mb4_general_ci		Yes	NULL		
6	date_created	timestamp			No	current_timestamp()		

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	doctor_id	int(11)			No	None		AUTO_INCREMENT
2	name	varchar(255)	utf8mb4_general_ci		No	None		
3	specialization	varchar(255)	utf8mb4_general_ci		Yes	NULL		
4	email	varchar(255)	utf8mb4_general_ci		Yes	NULL		
5	phone	varchar(15)	utf8mb4_general_ci		Yes	NULL		

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
<input type="checkbox"/> 1	appointment_id	int(11)			No	None		AUTO_INCREMENT
<input type="checkbox"/> 2	patient_id	int(11)			Yes	NULL		
<input type="checkbox"/> 3	doctor_id	int(11)			Yes	NULL		
<input type="checkbox"/> 4	appointment_date	date			Yes	NULL		
<input type="checkbox"/> 5	status	enum('Scheduled', 'Cancelled', 'Completed')	utf8mb4_general_ci		Yes	NULL		

8. Normalization of tables up to 3-NF

Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. The goal is to eliminate data anomalies and ensure that each piece of data is stored in only one place. To normalize the tables of the Healthcare Portal Database Management System (HP DBMS) up to the Third Normal Form (3NF), we'll analyze each table and make necessary adjustments.

Here are the steps to normalize the tables up to 3NF:

Patients Table:

1. 1st Normal Form (1NF):

- The Patients table is already in 1NF as it contains atomic values in each column.

2. 2nd Normal Form (2NF):

- No partial dependencies exist since all attributes depend on the entire primary key (patient_id). So, the table is in 2NF.

3. 3rd Normal Form (3NF):

- There are no transitive dependencies as well. Each non-key attribute directly depends on the primary key. Therefore, the Patients table is in 3NF.

Doctors Table:

1. 1st Normal Form (1NF):

- The Doctors table is already in 1NF as it contains atomic values in each column.

2. 2nd Normal Form (2NF):

- No partial dependencies exist since all attributes depend on the entire primary key (doctor_id). So, the table is in 2NF.

3. 3rd Normal Form (3NF):

- There are no transitive dependencies as well. Each non-key attribute directly depends on the primary key. Therefore, the Doctors table is in 3NF.

Appointments Table:

1. 1st Normal Form (1NF):

- The Appointments table is already in 1NF as it contains atomic values in each column.

2. 2nd Normal Form (2NF):

- No partial dependencies exist since all non-key attributes depend on the entire composite primary key (appointment_id, patient_id, doctor_id). So, the table is in 2NF.

3. 3rd Normal Form (3NF):

- There are no transitive dependencies as well. Each non-key attribute directly depends on the primary key. Therefore, the Appointments table is in 3NF.

MedicalRecords Table:

1. 1st Normal Form (1NF):

- The MedicalRecords table is already in 1NF as it contains atomic values in each column.

2. 2nd Normal Form (2NF):

- No partial dependencies exist since all non-key attributes depend on the entire composite primary key (record_id, patient_id, doctor_id). So, the table is in 2NF.

3. 3rd Normal Form (3NF):

- There are no transitive dependencies as well. Each non-key attribute directly depends on the primary key. Therefore, the MedicalRecords table is in 3NF.

Staff Table:

1. 1st Normal Form (1NF):

- The Staff table is already in 1NF as it contains atomic values in each column.

2. 2nd Normal Form (2NF):

- No partial dependencies exist since all attributes depend on the entire primary key (staff_id). So, the table is in 2NF.

3. 3rd Normal Form (3NF):

- There are no transitive dependencies as well. Each non-key attribute directly depends on the primary key. Therefore, the Staff table is in 3NF.

Users Table:

1. 1st Normal Form (1NF):

- The Users table is already in 1NF as it contains atomic values in each column.

2. 2nd Normal Form (2NF):

- No partial dependencies exist since all attributes depend on the entire primary key (user_id). So, the table is in 2NF.

3. 3rd Normal Form (3NF):

- There are no transitive dependencies as well. Each non-key attribute directly depends on the primary key. Therefore, the Users table is in 3NF.

All tables in the Healthcare Portal Database Management System (HP DBMS) are already normalized up to the Third Normal Form (3NF).

9. Creation of Data in the tables (at least 5 tables)

Here are SQL INSERT statements to create sample data in at least 5 tables from the Healthcare Portal Database Management System (HP DBMS):

1. Insert Data into Patients Table:

INSERT INTO Patients (patient_id,name, email, phone, address)

VALUES

(4,'Vidhathri', 'vids@example.com', '1234567890', '123 Main Street'),
(7,'Sindhu', 'sind@example.com', '9876543210', '456 Old Guntur'),
(3,'Darvs', 'darvs@example.com', '1112223333', '789 New Guntur');

		patient_id	name	email	phone	address
<input type="checkbox"/>	Edit Copy Delete	1	John Doe	johndoe@example.com	1234567890	123 Main St
<input type="checkbox"/>	Edit Copy Delete	3	Darvs	darvs@example.com	1112223333	789 New Guntur
<input type="checkbox"/>	Edit Copy Delete	4	Vidhathri	vids@example.com	1234567890	123 Main Street
<input type="checkbox"/>	Edit Copy Delete	7	Sindhu	sind@example.com	9876543210	456 Old Guntur

☐ Check all

With selected:

Edit Copy Delete Export

2. Insert Data into Doctors Table:

INSERT INTO Doctors (doctor_id,name, specialization, email, phone)
VALUES
(10,'Dr. Oshitha', 'Cardiologist', 'oshi@example.com', '5551234567'),
(11,'Dr. Poojitha', 'Pediatrician', 'pooji@example.com', '7778889999');

		doctor_id	name	specialization	email	phone
<input type="checkbox"/>	Edit Copy Delete	1	Dr. Smith	Cardiologist	drsmith@example.com	9876543210
<input type="checkbox"/>	Edit Copy Delete	10	Dr. Oshitha	Cardiologist	oshi@example.com	5551234567
<input type="checkbox"/>	Edit Copy Delete	11	Dr. Poojitha	Pediatrician	pooji@example.com	7778889999

☐ Check all

With selected:

Edit Copy Delete Export

3. Insert Data into Appointments Table:

INSERT INTO Appointments (appointment_id,patient_id, doctor_id, appointment_date,
status)
VALUES
(9,3, 1, '2024-05-02', 'Scheduled'),
(6,4, 10, '2024-05-03', 'Scheduled'),
(5,7, 11, '2024-05-04', 'Scheduled');

		appointment_id	patient_id	doctor_id	appointment_date	status
<input type="checkbox"/>	Edit Copy Delete	1	1	1	2024-05-10	Scheduled
<input type="checkbox"/>	Edit Copy Delete	3	NULL	NULL	2024-05-02	Scheduled
<input type="checkbox"/>	Edit Copy Delete	5	7	11	2024-05-04	Scheduled
<input type="checkbox"/>	Edit Copy Delete	6	4	10	2024-05-03	Scheduled
<input type="checkbox"/>	Edit Copy Delete	9	3	1	2024-05-02	Scheduled

4. Insert Data into MedicalRecords Table:

```
INSERT INTO MedicalRecords (record_id,patient_id, doctor_id, diagnosis, treatment)
```

```
VALUES
```

```
(2,3, 1, 'Heart condition', 'Medication and follow-up'),
```

```
(3,4, 10, 'Common cold', 'Rest and fluids'),
```

```
(4,7, 11, 'High blood pressure', 'Lifestyle changes and medication');
```

		record_id	patient_id	doctor_id	diagnosis	treatment	date_created
<input type="checkbox"/>	Edit Copy Delete	1	1	1	Heart condition	Medication and follow-up	2024-05-01 15:39:18
<input type="checkbox"/>	Edit Copy Delete	2	3	1	Heart condition	Medication and follow-up	2024-05-02 18:43:25
<input type="checkbox"/>	Edit Copy Delete	3	4	10	Common cold	Rest and fluids	2024-05-02 18:43:25
<input type="checkbox"/>	Edit Copy Delete	4	7	11	High blood pressure	Lifestyle changes and medication	2024-05-02 18:43:25

5. Insert Data into Staff Table:

```
INSERT INTO Staff (staff_id,name, role, email, phone)
```

```
VALUES
```

```
(1,'Admin', 'Administrator', 'admin@example.com', '5551234567'),
```

```
(2,'Receptionist 1', 'Receptionist', 'reception@example.com', '1112223333');
```

		staff_id	name	role	email	phone
<input type="checkbox"/>	Edit Copy Delete	1	Admin	Administrator	admin@example.com	5551234567
<input type="checkbox"/>	Edit Copy Delete	2	Receptionist 1	Receptionist	reception@example.com	1112223333

After executing these SQL INSERT statements in your database management tool (e.g., phpMyAdmin), the respective tables will be populated with sample data for testing and demonstration purposes. Adjust the data values as needed based on your requirements.

10. Few sql queries on the created tables

Here are a few SQL queries on the created tables from the Healthcare Portal Database Management System (HP DBMS):

1. Retrieve All Patients:

- Query to fetch all patient records from the Patients table.

```
SELECT * FROM Patients;
```

2. Retrieve Doctors by Specialization:

- Query to get doctors based on their specialization from the Doctors table.

```
SELECT * FROM Doctors WHERE specialization = 'Cardiologist';
```

3. List Appointments for a Patient:

- Query to display appointments for a specific patient from the Appointments table.

```
SELECT * FROM Appointments WHERE patient_id = 1;
```

4. Count Total Appointments per Doctor:

- Query to count the total appointments for each doctor from the Appointments table.

```
SELECT doctor_id, COUNT(appointment_id) AS TotalAppointments  
FROM Appointments  
GROUP BY doctor_id;
```

5. Retrieve Latest Medical Record for a Patient:

- Query to fetch the latest medical record for a specific patient from the MedicalRecords table.

```
SELECT * FROM MedicalRecords  
WHERE patient_id = 1  
ORDER BY date_created DESC  
LIMIT 1;
```

6. List Staff Members by Role:

- Query to display staff members based on their roles from the Staff table.

```
SELECT * FROM Staff WHERE role = 'Receptionist';
```

7. Authenticate User Login:

- Query to authenticate a user login based on username and password from the Users table.

```
SELECT * FROM Users WHERE username = 'admin' AND password = 'admin123';
```

8. Retrieve User Role by Username:

- Query to fetch the role of a user by their username from the Users table.

```
SELECT role FROM Users WHERE username = 'doctor1';
```

9. List All Scheduled Appointments:

- Query to display all appointments that are scheduled from the Appointments table.

```
SELECT * FROM Appointments WHERE status = 'Scheduled';
```

10. Update Doctor's Email:

- Query to update the email address of a doctor in the Doctor's table.

```
UPDATE Doctors SET email = 'newemail@example.com' WHERE doctor_id = 1;
```

11. Creation of 5 views using the tables

Table	Action	Rows	Type	Collation	Size	Overhead
latestmedicalrecordperpatient	Browse Structure Search Insert Edit Drop	~0	View	---	-	-
totalappointmentsperdoctor	Browse Structure Search Insert Edit Drop	~0	View	---	-	-
2 tables	Sum	~0	InnoDB	utf8mb4_general_ci	0 B	0 B

TotalAppointmentsPerDoctor View:

1. This view calculates the total number of appointments per doctor.

```
CREATE VIEW TotalAppointmentsPerDoctor AS
```

```
SELECT Doctors.name AS DoctorName, COUNT(Appointments.appointment_id) AS  
TotalAppointments
```

```
FROM Appointments
```

```
JOIN Doctors ON Appointments.doctor_id = Doctors.doctor_id
```

```
GROUP BY Doctors.doctor_id;
```

DoctorName	TotalAppointments
Dr. Smith	2
Dr. Oshitha	1
Dr. Poojitha	1

2. LatestMedicalRecordPerPatient View:

This view retrieves the latest medical record for each patient.













```
CREATE VIEW LatestMedicalRecordPerPatient AS
```

```
SELECT Patients.name AS PatientName, MedicalRecords.diagnosis,
MedicalRecords.treatment
```

```
FROM MedicalRecords
```

```
JOIN Patients ON MedicalRecords.patient_id = Patients.patient_id
```

```
WHERE MedicalRecords.date_created = (SELECT MAX(date_created) FROM MedicalRecords
WHERE MedicalRecords.patient_id = Patients.patient_id);
```

				PatientName	diagnosis	treatment
<input type="checkbox"/>				John Doe	Heart condition	Medication and follow-up
<input type="checkbox"/>				Darvs	Heart condition	Medication and follow-up
<input type="checkbox"/>				Vidhathri	Common cold	Rest and fluids
<input type="checkbox"/>				Sindhu	High blood pressure	Lifestyle changes and medication

3. ActivePatients View:

This view shows patients who have upcoming appointments (active patients).

```
CREATE VIEW ActivePatients AS
```

```
SELECT DISTINCT Patients.*
```

```
FROM Patients
```

```
JOIN Appointments ON Patients.patient_id = Appointments.patient_id
```

```
WHERE Appointments.appointment_date >= CURDATE();
```

4. OverdueAppointments View:

This view lists appointments that are overdue (past appointment date).

```
CREATE VIEW OverdueAppointments AS
```

```
SELECT Appointments.*, Patients.name AS PatientName, Doctors.name AS DoctorName
```

```
FROM Appointments
```

```
JOIN Patients ON Appointments.patient_id = Patients.patient_id
```

```
JOIN Doctors ON Appointments.doctor_id = Doctors.doctor_id
```

```
WHERE Appointments.appointment_date < CURDATE();
```

5. AppointmentStatusCount View:

This view provides a count of appointments by status (Scheduled, Canceled, Completed).

```
CREATE VIEW AppointmentStatusCount AS  
SELECT status, COUNT(*) AS Count  
FROM Appointments  
GROUP BY status;
```

END
