National Institute of Technology, Warangal

By Prasanth Reddy Maddirala

B.Tech MME, 4th Year

HOSPITAL DATABASE MANAGEMENT PROJECT

Problem Statement

Developing a hospital management system in order to effectively manage most aspects of hospitals such as booking appointments, managing patient records and keeping medical history.

Overview:

In this project, we have designed a database management system for hospital management. Hospitals interact with a lot of people in a day and there are various activities involved in day to day operations of hospitals, for example booking of appointments, managing doctor schedules, managing patient diagnoses, managing medical histories of patients, etc. The aim of this project is to show how data related to these tasks can be made easier to manage using databases

The database will contain important information about the doctors their schedules, patients and their medical history.

The project scope is:

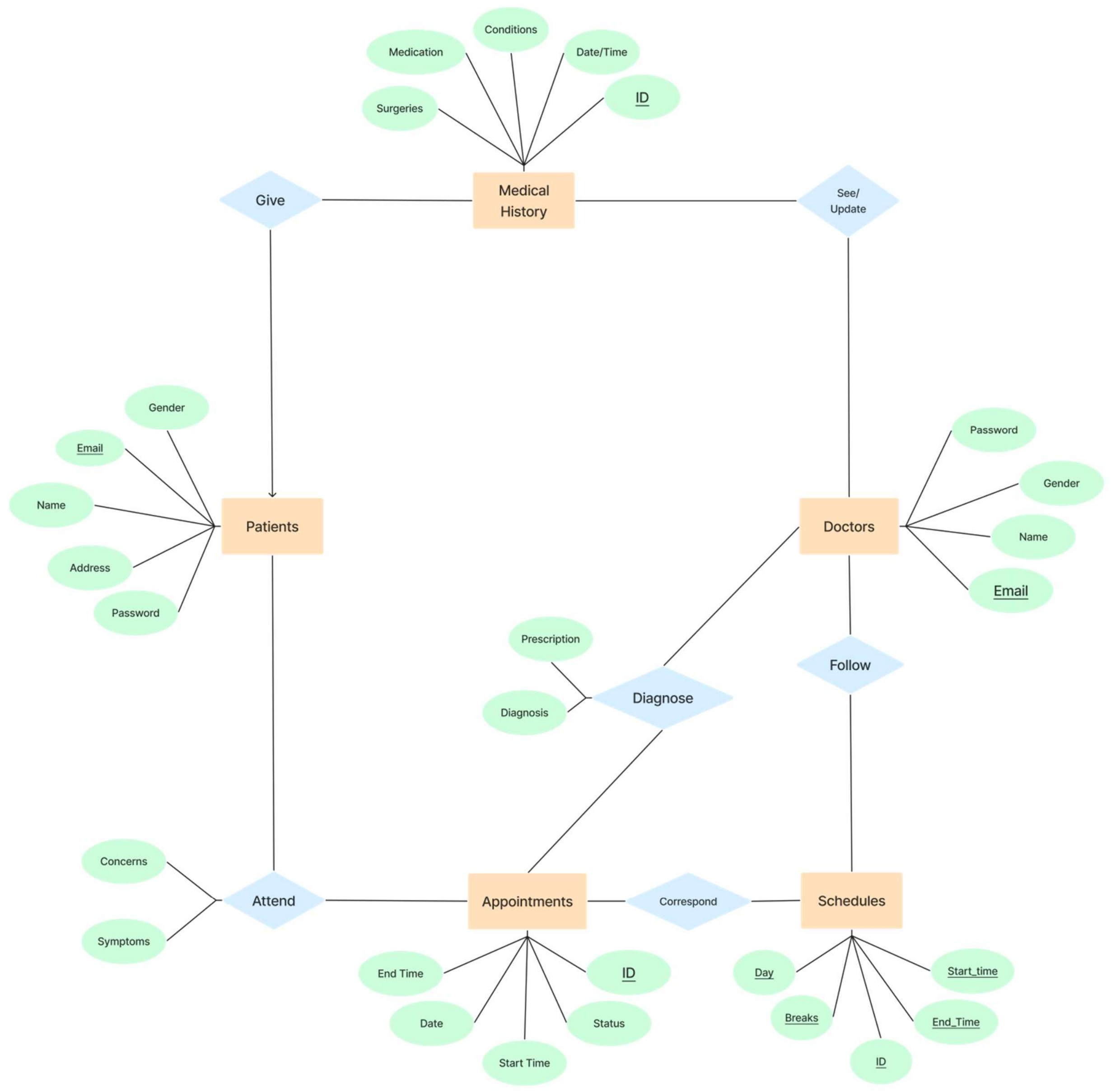
- To have a user-friendly and easy-to-use database application for hospital management.
- To have an application that secures data records.
- To have an application that can track the activities of the patients and doctors and their records easily.
- To have an application that organizes the appointments of patients and schedule of doctors clearly.
- To Minimise
  - Paper-based record keeping.
  - Mis-management of data due to manual and paper-based handling.

Contents:

ER Model Assumptions

1. Patients Give their medical history to Doctors
2. Two Patients can have a same Doctor Treating the Patient for different Instances.
3. Patient EMAILID can uniquely identify the Patients.
4. Doctor examines and may also update the patient's medical history in order to treat them.
5. Doctor EMAILID can uniquely identify the Doctors and they can have same name.
6. Patients Take appointments for treatment, which will be later attended by both Doctors and Patients according to the schedule.
7. Two Patients cannot have same Appointment for same Doctor.
8. Doctors then examine the patient and treat the patient with suitable diagnosis.
9. After the appointment the view status of Patient will be updated.

ER DIAGRAM:



Entities:

Medical History

Doctors

Schedules

Appointments

Patients

Relations:

See/Update

Follow

Correspond

Attend

Diagnose

Give

Tables

1. DOCTOR

Attributes	Data Type	Constraints and Characteristics
Email	VARCHAR	Primary Key, Not NULL
Name	VARCHAR	Not NULL
Gender	VARCHAR	Not NULL
Password	VARCHAR	Not NULL

2. PATIENT

Attributes	Data Type	Constraints and Characteristics
Email	VARCHAR	Primary Key, NOT NULL
Password	VARCHAR	NOT NULL
Name	VARCHAR	NOT NULL
Address	VARCHAR	NOT NULL
Gender	VARCHAR	NOT NULL

3. MEDICAL HISTORY

Attributes	Data Type	Constraints and Characteristics
Id	INT	Primary Key, Not NULL
Date	DATE	Not NULL
Conditions	VARCHAR	Not NULL
Surgeries	VARCHAR	Not NULL
Medication	VARCHAR	Not NULL

4. APPOINTMENT

Attributes	Data Type	Constraints and Characteristics
Id	INT	Primary Key, Not NULL
Date	DATE	Not NULL

Start time	TIME	Not NULL
End time	TIME	Not NULL
status	VARCHAR	Not NULL

5. PATIENTSATTENDEDAPPOINTMENTS

Attributes	**Data Type**	**Constraints and Characteristics**
Patient	VARCHAR	Primary key(1),Foreign Key
Appt	Int	Primary key(2), Foreign key
Concerns	VARCHAR	Not NULL
Symptoms	VARCHAR	Not NULL

6. SCHEDULE

Attributes	**Data Type**	**Constraints and Characteristics**
Id	INT	Primary Key(1), Not NULL
Start time	TIME	Primary key(2), Not NULL
End time	TIME	Primary key(3), Not NULL
breaktime	TIME	Primary key(4), Not NULL
Day	VARCHAR	Primary key(5), Not NULL

7. PatientsFillHistory

Attributes	**Data Type**	**Constraints and Characteristics**
Patient	VARCHAR	Foreign key, Not NULL
History	INT	Primary key, Foreign key,Not NULL

8. DIAGNOSE

Attributes	**Data Type**	**Constraints and Characteristics**
Appt	INT	Primary key(1), Not NULL

Doctor	VARCHAR	Primary key(2), Not NULL
Diagnosis	VARCHAR	Not NULL
Prescription	VARCHAR	Not NULL

9. DocsHaveSchedules

Attributes	Data Type	Constraints and Characteristics
sched	INT	Primary Key(1), Not NULL
Start time	TIME	Primary key(2), Not NULL
End time	TIME	Primary key(3), Not NULL
breaktime	TIME	Primary key(4), Not NULL
Day	VARCHAR	Primary key(5), Not NULL

10. Doctors View History

Attributes	Data Type	Constraints and Characteristics
History	INT	Primary Key(1), Not NULL, Foreign Key
Doctor	VARCHAR	Primary key(2), Not NULL, Foreign Key

Functional Requirements:

1. Separate interfaces for patients and doctors. Patients and doctors should have separate logins.
2. Allow patients to book appointments and give previous medical history.
3. Allow patients to view/update/cancel already booked appointments if necessary.
4. Allow doctors to cancel appointments.
5. Cancelled appointments should create free slots for other patients.
6. The system should avoid clash of appointments.

7. The system should take into consideration hospital and doctor schedules and allow appointments only when a doctor is not already busy or does not have a break.
8. Doctors should be able access patient history and profile, and add to patient history.
9. Doctors should be able to give diagnosis and prescriptions.
10. Patients should be able to see complete diagnosis, prescriptions and medical history.

Functional Dependencies and Normalisation

1. **Patient** :

   R = (**Email**, Password, Name, Address, Gender)
   FDs:
   a. Email -> Password
   b. Email -> Name
   c. Email -> Address
   d. Email -> Gender

   Table is in 1NF since all attributes are atomic.

   Table is in 2NF since there is no partial dependency.

   Table is in 3NF due to absence of any transitive dependency.

2. Medical History :

   R = (**id**, Date, Conditions, Surgeries, Medication)
   FDs:
   a. id -> Password
   b. id -> Date
   c. id -> Conditions
   d. id -> Surgeries
   e. id -> Medication

   Table is in 1NF since all attributes are atomic.

   Table is in 2NF since there is no partial dependency.

   Table is in 3NF due to absence of any transitive dependency.

3. Doctor :

   R = (**email**, gender, password, name)
   FDs:
   a. email -> gender
   b. email -> password
   c. email -> name

   Table is in 1NF since all attributes are atomic.

   Table is in 2NF since there is no partial dependency.

   Table is in 3NF due to absence of any transitive dependency.

4. Appointment:

   R = (**id**, date, start time, end time, status)
   FDs:
   a. id -> date
   b. id -> start time
   c. id -> end time
   d. id -> status

   Table is in 1NF since all attributes are atomic.

   Table is in 2NF since there is no partial dependency.

   Table is in 3NF due to absence of any transitive dependency.

5. PatientsAttendAppointments:

   R = (**patient, appointment**, concerns, symptoms)
   FDs:
   a. (patient, appointment) -> concerns
   b. (patient, appointment) -> symptoms

   Table is in 1NF since all attributes are atomic.

   Table is in 2NF since there is no partial dependency.

   Table is in 3NF due to absence of any transitive dependency.

6. Schedule:

   R = (**id, start time, end time, break time, day**)

   Since entire table is the key, it does not have partial and transitive dependencies. It also has atomic attributes. Hence it is in 3NF.

7. PatientsFillHistory:

R = (Patient, **History**)
FDs:
a. History -> Patient

Table is in 1NF since all attributes are atomic.

Table is in 2NF since there is no partial dependency.

Table is in 3NF due to absence of any transitive dependency.

8. Diagnose:

R = (**appointment, doctor,diagnosis,prescription**)
FDs:

a. (appointment, doctor) -> diagnosis

b. (appointment, doctor) -> prescription

Table is in 1NF since all attributes are atomic.

Table is in 2NF since there is no partial dependency.

Table is in 3NF due to absence of any transitive dependency.

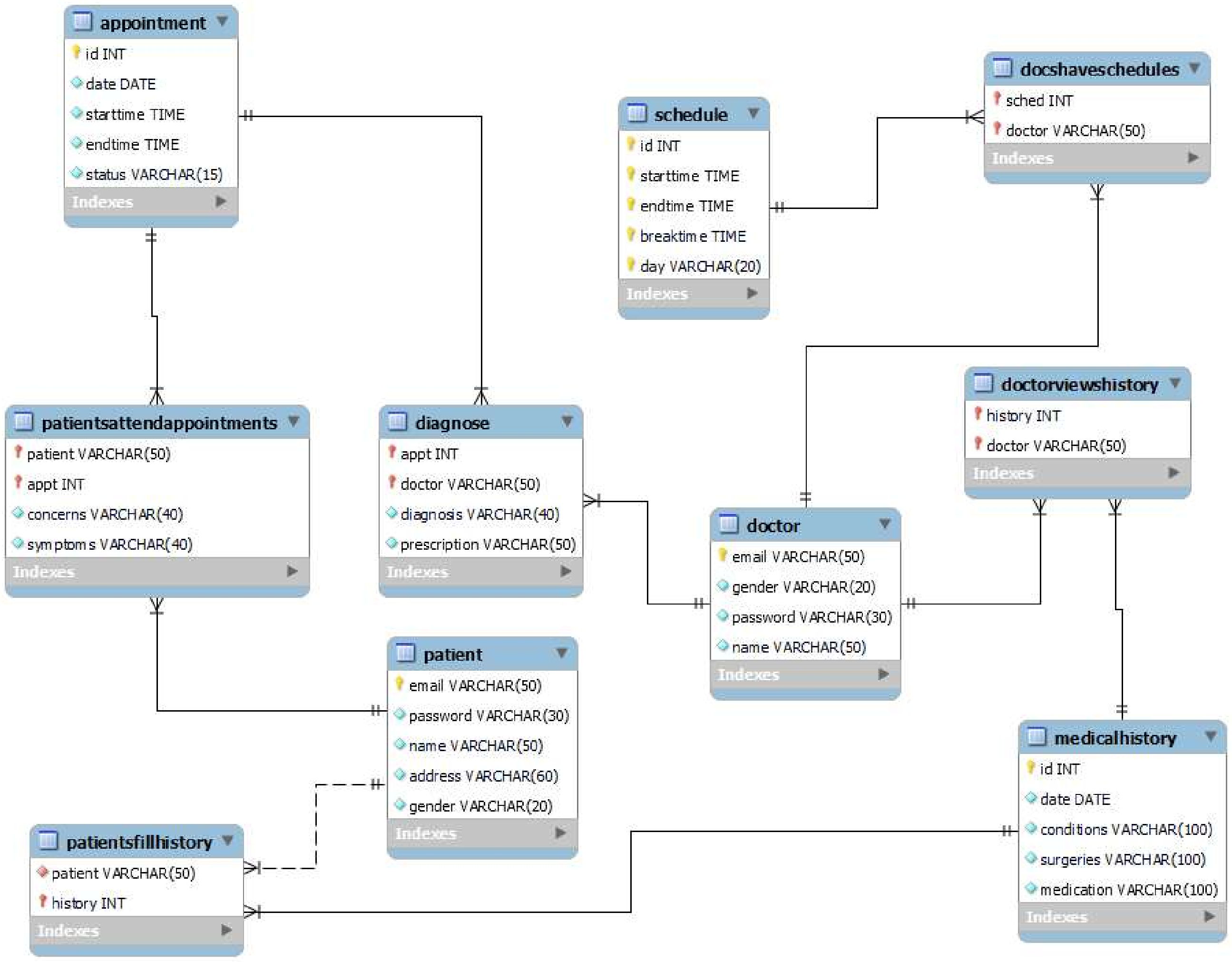
9. DoctorsHaveSchedules:

R = (**Schedule, Doctor**)
Since entire table is the key, it does not have partial and transitive dependencies. It also has atomic attributes.
Hence it is in 3NF.

10. DoctorViewsHistory:

R = (**history, doctor**)
Since entire table is the key, it does not have partial and transitive dependencies. It also has atomic attributes.
Hence it is in 3NF.

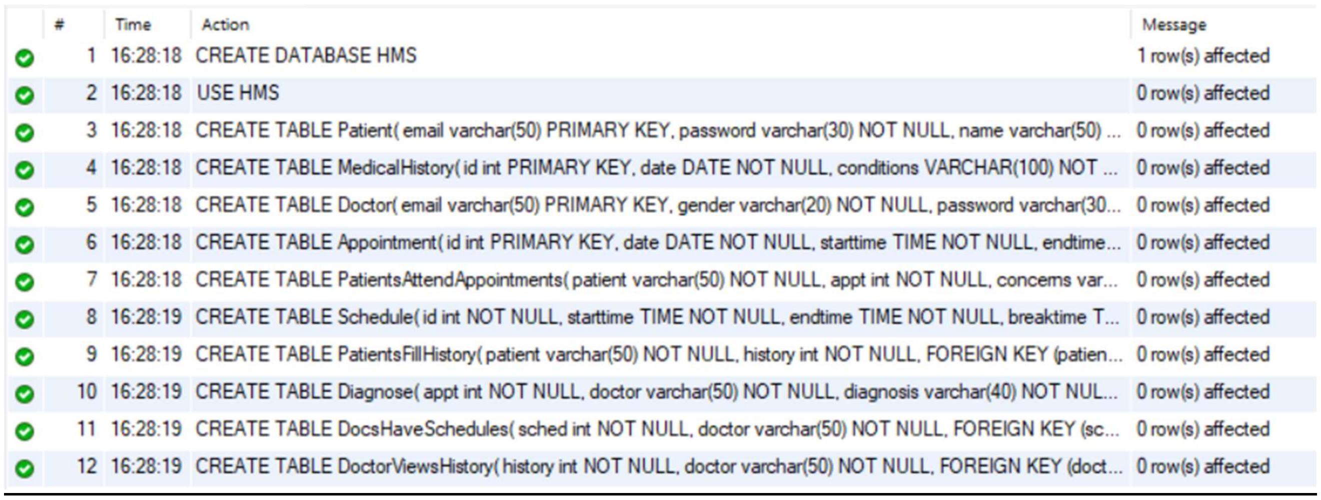
RELATIONAL SCHEMA WITH NORMALIZED TABLES:



SQL CODE:

```
CREATE DATABASE HMS;
USE HMS;

CREATE TABLE Patient(
email varchar(50) PRIMARY KEY,
password varchar(30) NOT NULL,
name varchar(50) NOT NULL,
address varchar(60) NOT NULL,
gender VARCHAR(20) NOT NULL
);

CREATE TABLE MedicalHistory(
id int PRIMARY KEY,
date DATE NOT NULL,
conditions VARCHAR(100) NOT NULL,
surgeries VARCHAR(100) NOT NULL,
medication VARCHAR(100) NOT NULL
);

CREATE TABLE Doctor(
email varchar(50) PRIMARY KEY,
gender varchar(20) NOT NULL,
password varchar(30) NOT NULL,
name varchar(50) NOT NULL
);

CREATE TABLE Appointment(
id int PRIMARY KEY,
date DATE NOT NULL,
starttime TIME NOT NULL,
endtime TIME NOT NULL,
status varchar(15) NOT NULL
);

CREATE TABLE PatientsAttendAppointments(
patient varchar(50) NOT NULL,
appt int NOT NULL,
concerns varchar(40) NOT NULL,
symptoms varchar(40) NOT NULL,
FOREIGN KEY (patient) REFERENCES Patient (email) ON DELETE CASCADE,
FOREIGN KEY (appt) REFERENCES Appointment (id) ON DELETE CASCADE,
PRIMARY KEY (patient, appt)
);

CREATE TABLE Schedule(
id int NOT NULL,
starttime TIME NOT NULL,
endtime TIME NOT NULL,
breaktime TIME NOT NULL,
day varchar(20) NOT NULL,
PRIMARY KEY (id, starttime, endtime, breaktime, day)
);

CREATE TABLE PatientsFillHistory(
patient varchar(50) NOT NULL,
history int NOT NULL,
FOREIGN KEY (patient) REFERENCES Patient (email) ON DELETE CASCADE,
FOREIGN KEY (history) REFERENCES MedicalHistory (id) ON DELETE
CASCADE,
PRIMARY KEY (history)
);

CREATE TABLE Diagnose(
appt int NOT NULL,
doctor varchar(50) NOT NULL,
diagnosis varchar(40) NOT NULL,
prescription varchar(50) NOT NULL,
FOREIGN KEY (appt) REFERENCES Appointment (id) ON DELETE CASCADE,
FOREIGN KEY (doctor) REFERENCES Doctor (email) ON DELETE CASCADE,
PRIMARY KEY (appt, doctor)
);

CREATE TABLE DocsHaveSchedules(
sched int NOT NULL,
doctor varchar(50) NOT NULL,
FOREIGN KEY (sched) REFERENCES Schedule (id) ON DELETE CASCADE,
FOREIGN KEY (doctor) REFERENCES Doctor (email) ON DELETE CASCADE,
PRIMARY KEY (sched, doctor)
);

CREATE TABLE DoctorViewsHistory(
history int NOT NULL,
doctor varchar(50) NOT NULL,
FOREIGN KEY (doctor) REFERENCES Doctor (email) ON DELETE CASCADE,
FOREIGN KEY (history) REFERENCES MedicalHistory (id) ON DELETE
CASCADE,
PRIMARY KEY (history, doctor)
);
```

OUTPUT OF CREATION:



```
#   Time      Action                                                                                                        Message
1   16:28:18  CREATE DATABASE HMS                                                                                           1 row(s) affected
2   16:28:18  USE HMS                                                                                                       0 row(s) affected
3   16:28:18  CREATE TABLE Patient( email varchar(50) PRIMARY KEY, password varchar(30) NOT NULL, name varchar(50) ...      0 row(s) affected
4   16:28:18  CREATE TABLE MedicalHistory( id int PRIMARY KEY, date DATE NOT NULL, conditions VARCHAR(100) NOT ...          0 row(s) affected
5   16:28:18  CREATE TABLE Doctor( email varchar(50) PRIMARY KEY, gender varchar(20) NOT NULL, password varchar(30...        0 row(s) affected
6   16:28:18  CREATE TABLE Appointment( id int PRIMARY KEY, date DATE NOT NULL, starttime TIME NOT NULL, endtime...          0 row(s) affected
7   16:28:18  CREATE TABLE PatientsAttendAppointments( patient varchar(50) NOT NULL, appt int NOT NULL, concerns var...      0 row(s) affected
8   16:28:19  CREATE TABLE Schedule( id int NOT NULL, starttime TIME NOT NULL, endtime TIME NOT NULL, breaktime T...          0 row(s) affected
9   16:28:19  CREATE TABLE PatientsFillHistory( patient varchar(50) NOT NULL, history int NOT NULL, FOREIGN KEY (patien...    0 row(s) affected
10  16:28:19  CREATE TABLE Diagnose( appt int NOT NULL, doctor varchar(50) NOT NULL, diagnosis varchar(40) NOT NUL...        0 row(s) affected
11  16:28:19  CREATE TABLE DocsHaveSchedules( sched int NOT NULL, doctor varchar(50) NOT NULL, FOREIGN KEY (sc...           0 row(s) affected
12  16:28:19  CREATE TABLE DoctorViewsHistory( history int NOT NULL, doctor varchar(50) NOT NULL, FOREIGN KEY (doct...       0 row(s) affected
```

INSERTION OF VALUES INTO DATABASE

```
INSERT INTO Patient(email,password,name,address,gender)
VALUES
('ramesh@gmail.com','hrishikesh13','Ramesh','Tamil Nadu', 'male'),
('suresh@gmail.com','hrishikesh13','Suresh','Karnataka', 'male'),
('rakesh@gmail.com','hrishikesh13','Rakesh','Gujarat', 'male')
;

INSERT INTO MedicalHistory(id,date,conditions,surgeries,medication)
VALUES
(1,'19-01-14','Pain in abdomen','Heart Surgery','Crocin'),
(2,'19-01-14','Frequent Indigestion','none','none'),
(3,'19-01-14','Body Pain','none','Iodex')
;

INSERT INTO Doctor(email, gender, password, name)
VALUES
('hathalye7@gmail.com', 'male', 'hrishikesh13', 'Hrishikesh Athalye'),
('hathalye8@gmail.com', 'male', 'hrishikesh13', 'Hrishikesh Athalye')
;

INSERT INTO Appointment(id,date,starttime,endtime,status)
VALUES
(1, '19-01-15', '09:00', '10:00', 'Done'),
(2, '19-01-16', '10:00', '11:00', 'Done'),
(3, '19-01-18', '14:00', '15:00', 'Done')
;
```

```
INSERT INTO
PatientsAttendAppointments(patient,appt,concerns,symptoms)
VALUES
('ramesh@gmail.com',1, 'none', 'itchy throat'),
('suresh@gmail.com',2, 'infection', 'fever'),
('rakesh@gmail.com',3, 'nausea', 'fever')
;

INSERT INTO Schedule(id,starttime,endtime,breaktime,day)
VALUES
(001,'09:00','17:00','12:00','Tuesday'),
(001,'09:00','17:00','12:00','Friday'),
(001,'09:00','17:00','12:00','Saturday'),
(001,'09:00','17:00','12:00','Sunday'),
(002,'09:00','17:00','12:00','Wednesday'),
(002,'09:00','17:00','12:00','Friday')
;

INSERT INTO PatientsFillHistory(patient,history)
VALUES
('ramesh@gmail.com', 1),
('suresh@gmail.com', 2),
('rakesh@gmail.com', 3)
;

INSERT INTO Diagnose (appt,doctor,diagnosis,prescription)
VALUES
(1,'hathalye7@gmail.com', 'Bloating', 'Ibuprofen as needed'),
(2,'hathalye8@gmail.com', 'Muscle soreness', 'Stretch morning/night'),
(3,'hathalye8@gmail.com', 'Vitamin Deficiency', 'Good Diet')
;

INSERT INTO DocsHaveSchedules(sched,doctor)
VALUES
(001,'hathalye7@gmail.com'),
(002,'hathalye8@gmail.com')
;

INSERT INTO DoctorViewsHistory(history,doctor)
VALUES
(1,'hathalye7@gmail.com'),
(2,'hathalye8@gmail.com'),
(3,'hathalye8@gmail.com')
;
```