JSS MAHAVIDYAPEETHA
JSS SCIENCE AND TECHNOLOGY UNIVERSITY
SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING
JSS Technical Institutions Campus, Mysuru – 570006

“Implementation of Hospital Management System”

Mini project report submitted in partial fulfillment of curriculum prescribed for the Database Management Systems (IS520) course for the award of the degree of

BACHELOR OF ENGINEERING
IN
INFORMATION SCIENCE AND ENGINEERING

by

D.V.Varun Reddy
(01JST18IS009)

Under the Guidance of

Dr. Trisiladevi C. Nagavi
Assistant Professor,
Dept. of CS&E ,
SJCE, JSS STU Mysore

SEPTEMBER 2021

JSS MAHAVIDYAPEETHA
JSS SCIENCE AND TECHNOLOGY UNIVERSITY
SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING
JSS Technical Institutions Campus, Mysuru – 570006

CERTIFICATE

This is to certify that the work entitled **“Hospital Management System”** is a bonafied work carried out **by D.V.Varun Reddy** in partial fulfillment of the award of the degree of **Bachelor of Engineering in Information Science and Engineering of SJCE, JSS Science and Technology, Mysuru during the year 2021**. It is certified that all corrections / suggestions indicated during CIE have been incorporated in the report. The mini project report has been approved as it satisfies the academic requirements in respect of mini project work prescribed for the **Database Management Systems (IS520)** course.

Course in Charge and Guide

Dr. Trisiladevi C. Nagavi
Assistant Professor,
Dept. of CS&E ,
SJCE, JSS STU Mysore

Place: Mysore **Signature:**

Date:

CONTENTS

1.INTRODUCTION

Background:

Hospitals interact with a lot of people in a day and there are various activities involved in day to day operations for example booking of appointments, managing doctor schedules, managing patient diagnoses, managing medical histories of patients, etc.

Objective of the Project:

The aim of this project is to show how data related to these tasks can be made easier to manage using databases

Features of the Project: Patient Side Features:

1. There is a separate interface for patients. Patients have a separate login.

2. Patients can book appointments.

3. Patients can give previous medical history.

4. Patients can view/update/cancel already booked appointments if necessary.

5. Cancelled appointments create free slots for other patients.

6. The system avoids clash of appointments with other patients. Each patient is therefore ensured his/her slot.

7. Patients are able to see complete diagnosis, prescriptions and medical history.

8. Patient medical history is only available to the doctor with whom the appointment is booked to ensure privacy.

Doctor Side Features:

1. There is a separate interface for doctors. Doctors have a separate login.

2. The system takes into consideration doctor schedules and does not allow appointments when a doctor is already busy or has a break.

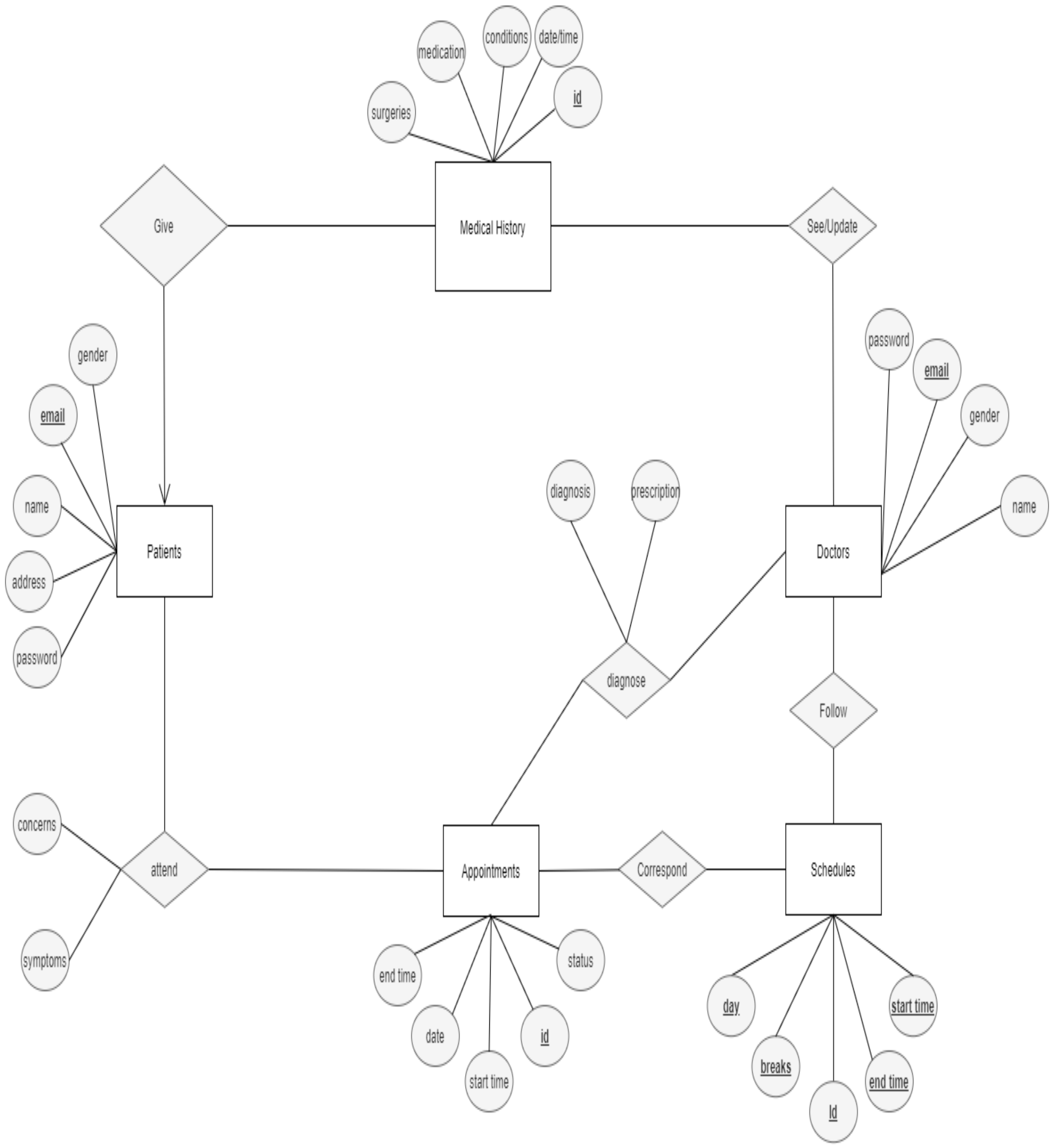
3. Doctors are able to access patient history and profile, and add to patient history.

4. Doctors are able to give diagnosis and prescriptions.

5. Doctors are able to modify diagnosis and prescriptions.

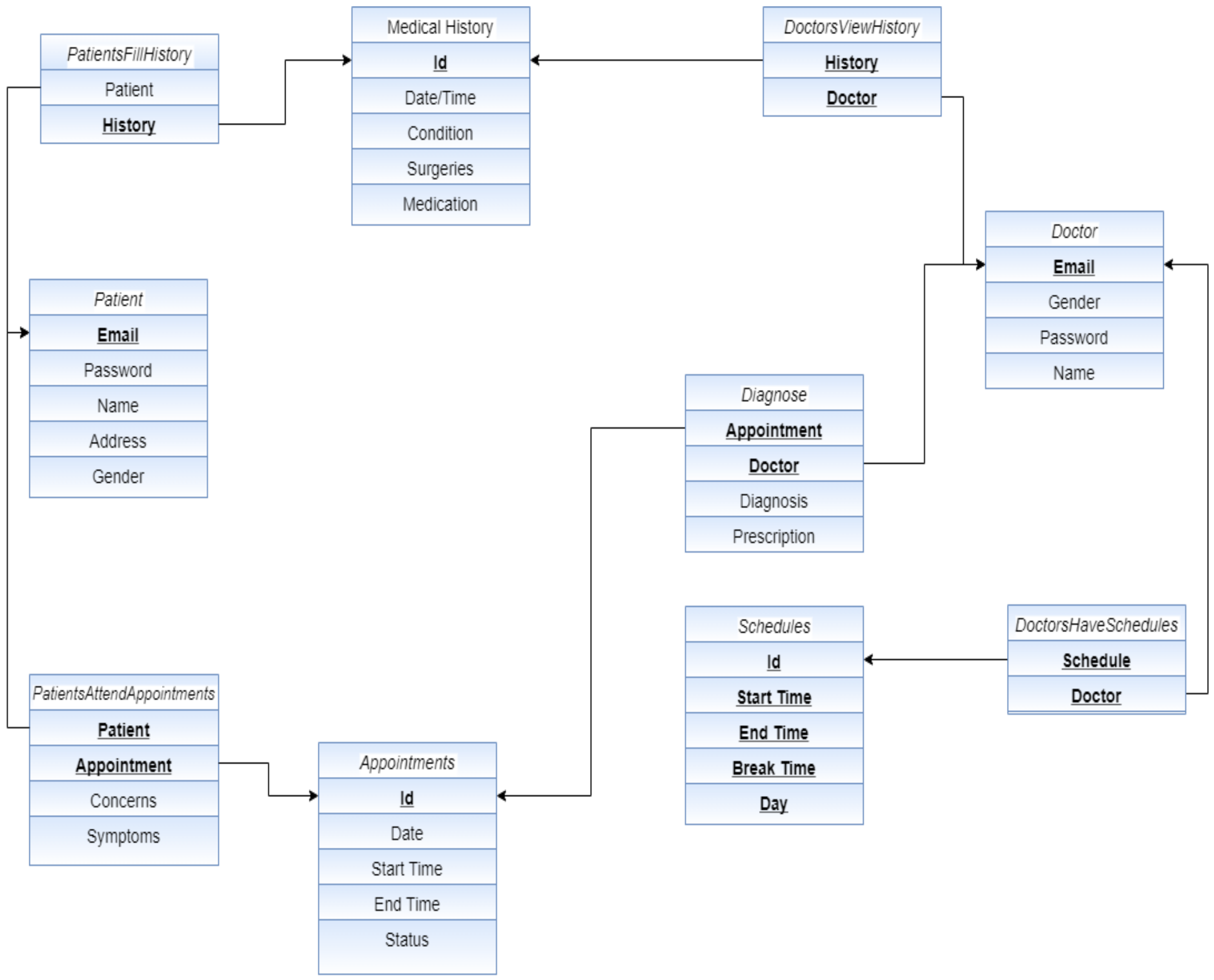
2. System Design

ER Diagram:



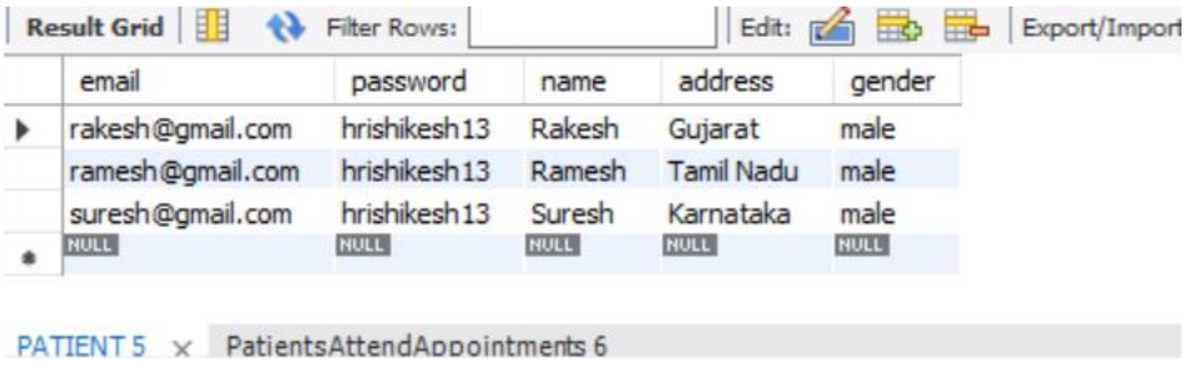
medication
conditions
date/time
surgeries
id
Give
Medical History
See/Update
gender
email
name
address
password
Patients
password
email
gender
name
Doctors
diagnosis
prescription
diagnose
Follow
concerns
attend
symptoms
Appointments
Correspond
Schedules
end time
status
date
id
start time
day
start time
breaks
end time
Id

Schema Diagram:



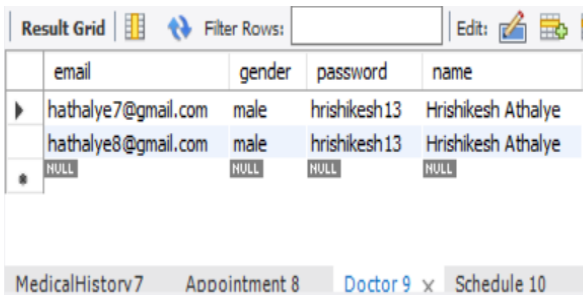
State Diagram:

Patient:



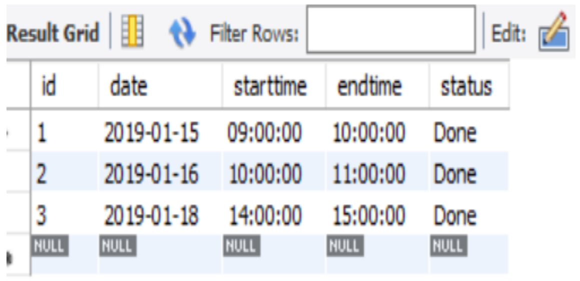
email	password	name	address	gender
rakesh@gmail.com	hrishikesh13	Rakesh	Gujarat	male
ramesh@gmail.com	hrishikesh13	Ramesh	Tamil Nadu	male
suresh@gmail.com	hrishikesh13	Suresh	Karnataka	male
NULL	NULL	NULL	NULL	NULL

Doctor:



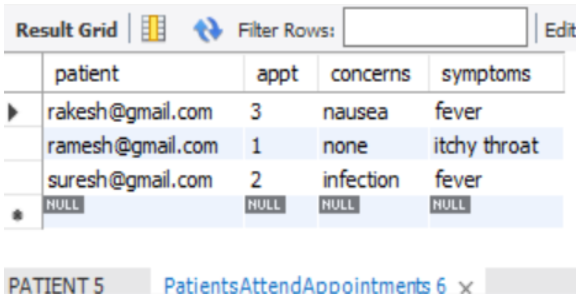
email	gender	password	name
hathalye7@gmail.com	male	hrishikesh13	Hrishikesh Athalye
hathalye8@gmail.com	male	hrishikesh13	Hrishikesh Athalye
NULL	NULL	NULL	NULL

Appointment:



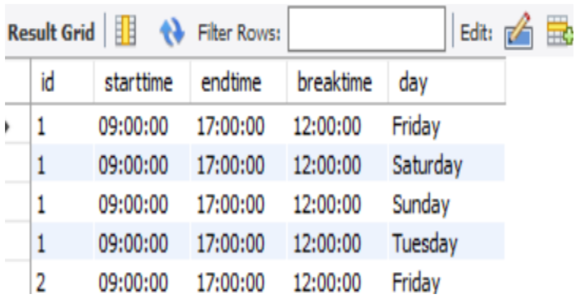
id	date	starttime	endtime	status
1	2019-01-15	09:00:00	10:00:00	Done
2	2019-01-16	10:00:00	11:00:00	Done
3	2019-01-18	14:00:00	15:00:00	Done
NULL	NULL	NULL	NULL	NULL

PatientsAttendAppointments:



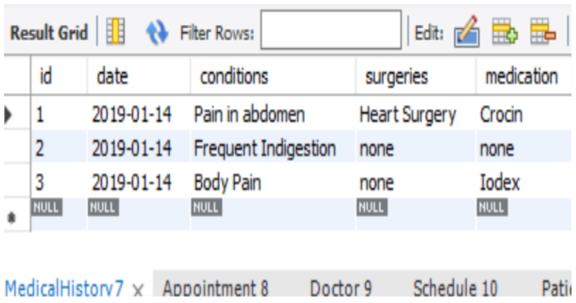
patient	appt	concerns	symptoms
rakesh@gmail.com	3	nausea	fever
ramesh@gmail.com	1	none	itchy throat
suresh@gmail.com	2	infection	fever
NULL	NULL	NULL	NULL

Schedule:



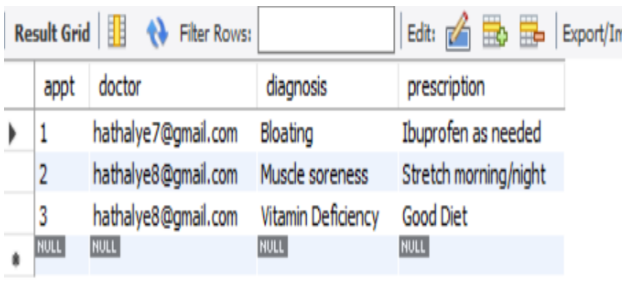
id	starttime	endtime	breaktime	day
1	09:00:00	17:00:00	12:00:00	Friday
1	09:00:00	17:00:00	12:00:00	Saturday
1	09:00:00	17:00:00	12:00:00	Sunday
1	09:00:00	17:00:00	12:00:00	Tuesday
2	09:00:00	17:00:00	12:00:00	Friday

MedicalHistory:



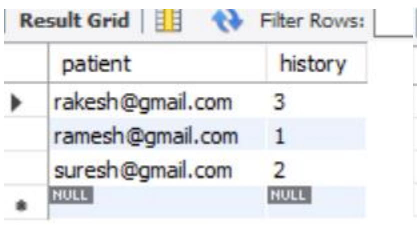
id	date	conditions	surgeries	medication
1	2019-01-14	Pain in abdomen	Heart Surgery	Crocin
2	2019-01-14	Frequent Indigestion	none	none
3	2019-01-14	Body Pain	none	Iodex
NULL	NULL	NULL	NULL	NULL

Diagnose:



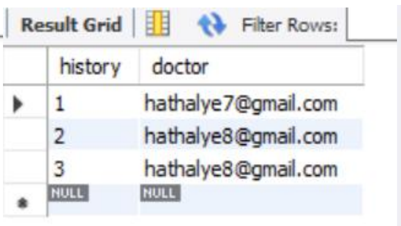
appt	doctor	diagnosis	prescription
1	hathalye7@gmail.com	Bloating	Ibuprofen as needed
2	hathalye8@gmail.com	Muscle soreness	Stretch morning/night
3	hathalye8@gmail.com	Vitamin Deficiency	Good Diet
NULL	NULL	NULL	NULL

PatientsFillHistory:



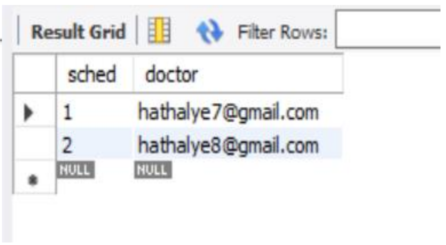
patient	history
rakesh@gmail.com	3
ramesh@gmail.com	1
suresh@gmail.com	2
NULL	NULL

DoctorViewsHistory:



history	doctor
1	hathalye7@gmail.com
2	hathalye8@gmail.com
3	hathalye8@gmail.com
NULL	NULL

DocsHaveSchedules:



sched	doctor
1	hathalye7@gmail.com
2	hathalye8@gmail.com
NULL	NULL

3.Normalization upto 3nf:

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.

A relation is in 1NF if it contains an atomic value.

A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.

A relation will be in 3NF if it is in 2NF and no transition dependency exists.

Functional Dependencies and Normalization

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 = (**<u>id</u>**, 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 = (**<u>patient, appointment</u>**, 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 = (**<u>id, start time, end time, break time, day</u>**)

   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, **<u>History</u>**)
   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 = (**<u>appointment, doctor,</u> 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 = (**<u>Schedule, Doctor</u>**)
   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 = (**<u>history, doctor</u>**)
   Since entire table is the key, it does not have partial and transitive dependencies. It also has atomic attributes.
   Hence it is in 3NF.

4. System Implementation:

Introduction to MySQL:

MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by **Oracle Company**.

MySQL tells how to manage database and to manipulate data with the help of various SQL queries. These queries are: insert records, update records, delete records, select records, create tables, drop tables, etc. There are also given MySQL interview questions to help you better understand the MySQL database.

Why MySQL?

- MySQL is an open-source database, so you don't have to pay a single penny to use it.

- MySQL is a very powerful program that can handle a large set of functionality of the most expensive and powerful database packages.
- MySQL is customizable because it is an open-source database, and the open-source GPL license facilitates programmers to modify the SQL software according to their own specific environment.
- MySQL is quicker than other databases, so it can work well even with the large data set.
- MySQL supports many operating systems with many languages like PHP, PERL, C, C++, JAVA, etc.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL is very friendly with PHP, the most popular language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

Relational Algebraic Queries:

Select: Symbol- σ

$\sigma_{id='3'}$(MedicalHistory)

Output – selects the row in medical history table where id = 3.

Project: Symbol- π

Π_{email} (Patient)

Output – retrieves the column email in patient table.

SQL Commands:

Create Command:

Syntax:

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....);
```

```
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
);
```

A table is created with email, password, name, address, gender as column names. Here email is given as primary key.

Insert Command:

Syntax:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

```
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');
```

All these values are inserted in the table, the table gets populated with the insert command.

Update Command:

Syntax:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

```
UPDATE Patient
SET name='Varun'
WHERE email='suresh@gmail.com';
```

Here, Suresh gets replaced with Varun in name column and the WHERE helps us to choose a condition like we chose the row which has 'suresh@gmail.com' as the email.

Delete Command:

Syntax:

```
DELETE FROM table_name WHERE condition;
```

Now we shall consider two tables i.e one is referenced table(patient table) and the other is referencing table(PatientAttendsAppointments).

If you have a look at the values of Patient table and it's constraints, then you will observe that email is primary key.
And the same email Patient column in PatientAttendsAppointments table is foreign key which is referring to the primary key (email column) of Patient table.

```
FOREIGN KEY (patient) REFERENCES Patient (email) ON DELETE CASCADE
```
Here this I used in referencing table in create command.

On delete cascade is used to delete the record in child table when corresponding parent table record gets deleted. It helps to ensure referential integrity.

```
DELETE FROM Patient WHERE email='rakesh@gmail.com';
```
Here the record/row which has email as 'rakesh@gmail.com' is completely deleted in both referenced and referencing relation.

Constraints like NOT NULL is been used to make sure that no null values are entered.

DEFAULT is used to put a default value instead of leaving it null when no value is entered.

CHECK is been used to give a particular range for the domain of the specific column.

5. CONCLUSION:

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. Hence a good database design is needed with proper implementation for effective hospital management.

6. REFERENCES:

https://www.w3schools.com

https://www.guru99.com

Fundamentals of Database Systems (7th edition) by Shamkant B. Navathe,Ramez Elmasri