DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

<<PES1201802101>>   <<Girish G N>>

The database system that has been developed is Hospital Database Management System. The Hospital database management system is a database design used for easily and securely managing hospital functions and events. It enables the admin to register a patient for the hospital, store their details into the database, store the details about doctors working there, labs available for the testing and the bill details. Any of the staff members, doctor & admin is able to add, view, edit, update or delete data.

Introduction

There are seven common features of Hospital Management System Database Design such as Managing Patients, Doctors, laboratory, Inpatient, Outpatient, Rooms, and Hospital Bills information.

ER Diagram of Hospital Database Management System

● Patients: This table of ER Diagram Hospital Management System will be used for storing and managing the patient information.

● Doctor: This table will be used for storing and managing the Doctor info and login account.

● Laboratory: This table will be used for storing and managing the Laboratory transaction.

● Inpatient: This table will be used for storing and managing the inpatient information and diagnosis.

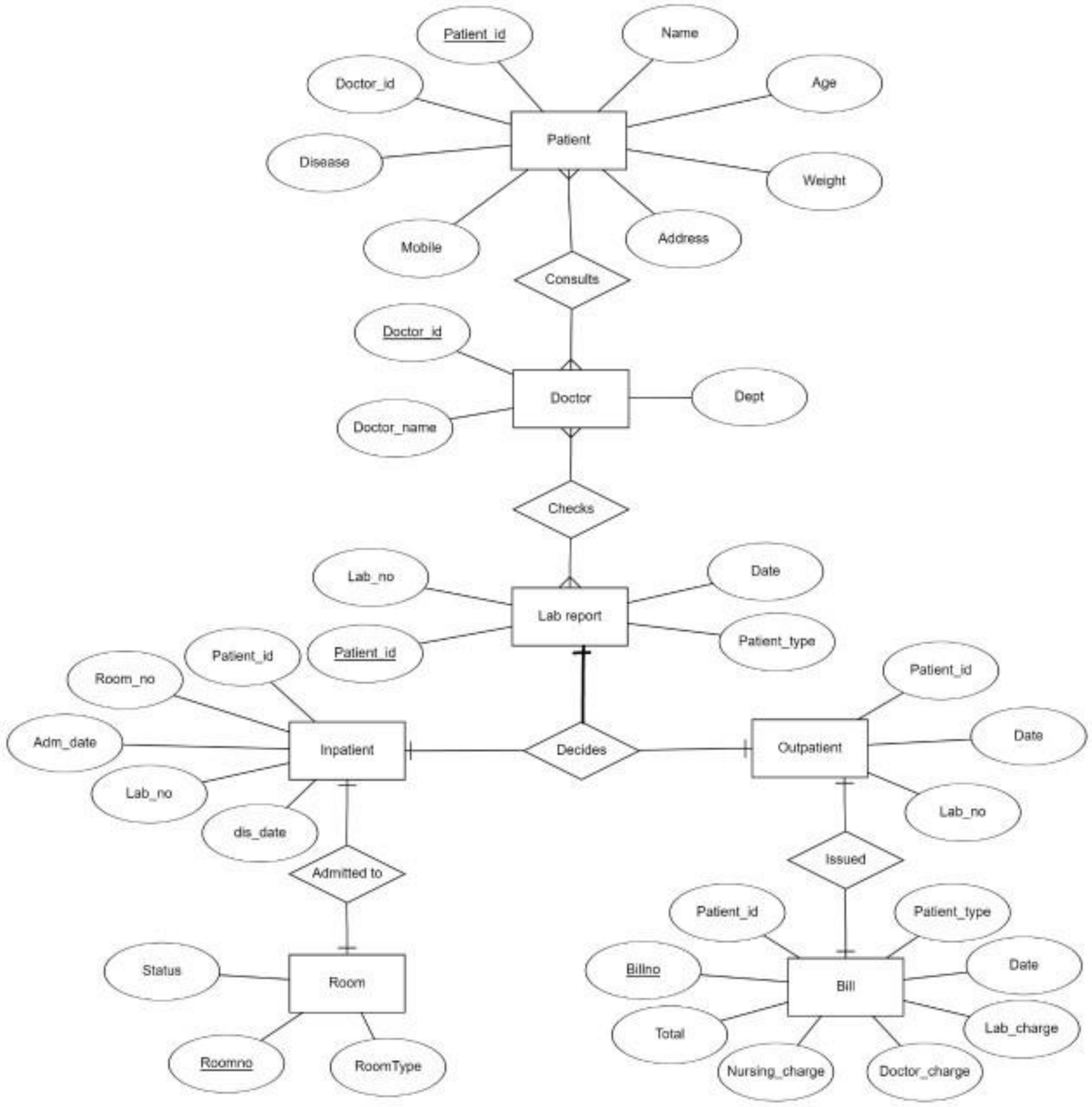
● Outpatient: This table will be used for storing and managing the inpatient information and diagnosis.

●Room: This table will be used for storing and managing the room information and assigning patients in every room.

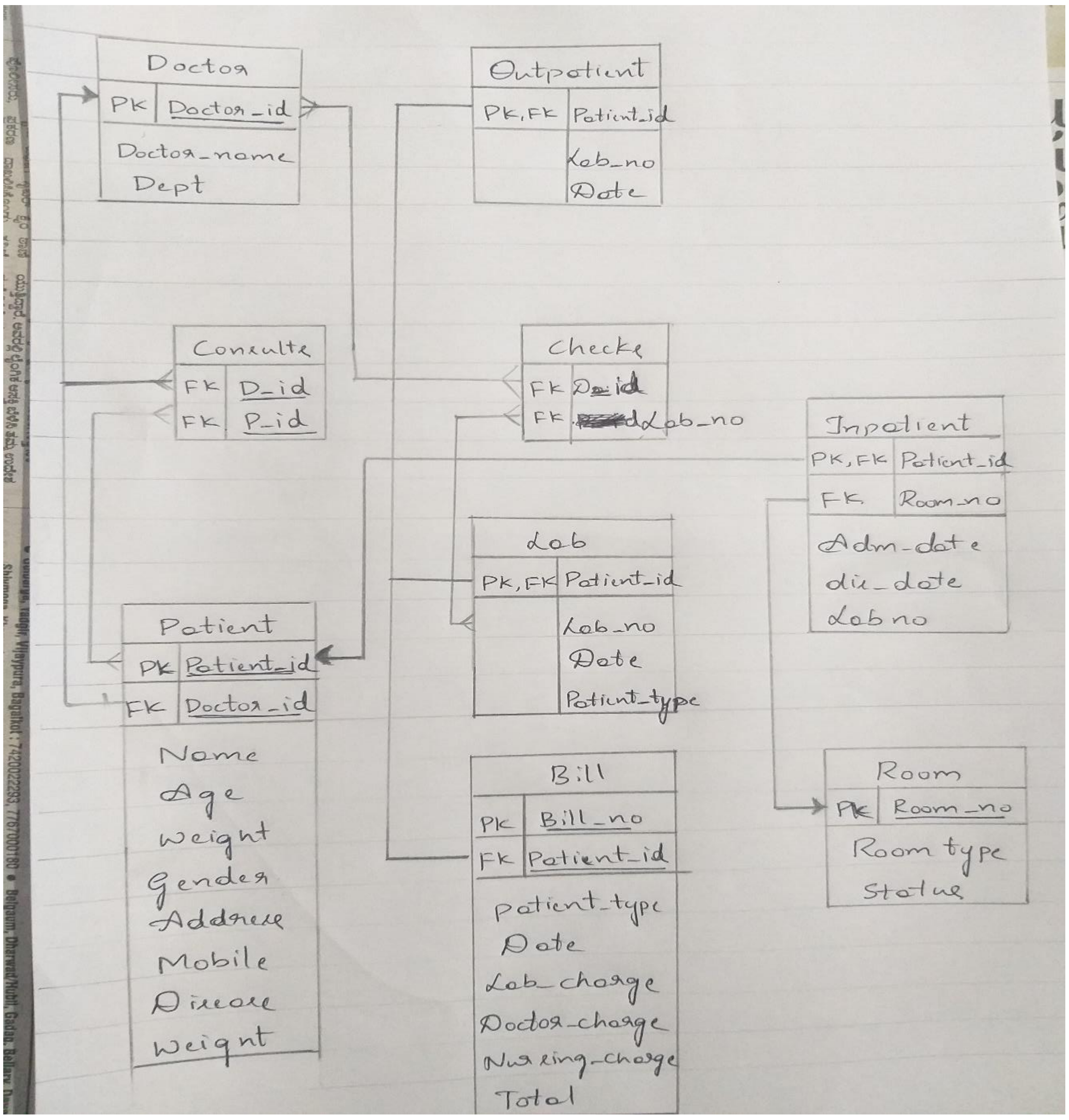
● Bills: The billing table will be used for managing the statement of accounts per patient and for the collection of bills.

Data Model

E R DIAGRAM



DATBASE SCHEMA



FD and Normalization

From the above given schema, it is easy to develop the functional dependencies of the tables

● Doctor : {Doctor_id} -> {Doctor_name, Dept}

Primary Key - Doctor_id

● Room : {Room_no} -> {RoomType, Status}

Primary Key - Room_No

● Patient : {Patient_id} -> {Name, Age, Weight, Gender, Address, Mobile, Disease, Doctorid}

Primary Key - Patient_id

Foreign Key - Doctor_id

● Lab : {Patient_id} -> {Lab_no, Date, Patient_type}

Primary Key - Patient_id

Foreign Key - Patient_id

● Inpatient : {Patient_id} -> {Room_No, Adm_Date, Dis_Date, Lab_no}

Primary Key - Patient_id

Foreign Key - Patient_id

Foreign Key - Room_No

● Outpatient : {Patient_id} -> {Room_No, Date, Lab_no}

Primary Key - Patient_id

Foreign Key - Patient_id

● Bill : {Bill_no, Pid} -> {Patient_type, Date, Lab_charge, Doctor_charge, Nursing_charge, Total}

Primary Key – Bill_no

Foreign Key – Patient_id

NORMALIZATION

First Normal Form - As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

Second Normal Form - A table is said to be in 2NF if both the following conditions hold:

1. Table is in First Normal form.

2. No non prime attribute is dependent on the proper subset of the candidate key of the table.

Third Normal Form - A table is said to be in 3NF if both the following conditions hold:

1. Table should be in Second Normal form.

2. Transitive functional dependency of non-prime attribute on any super key should be removed.

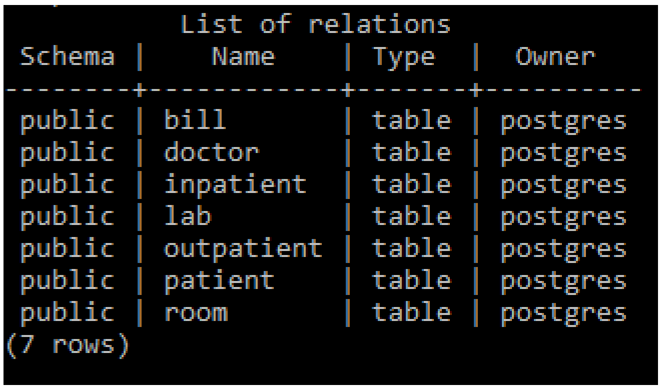
From functional dependencies, we can observe that all rows can have atomic or single values, therefore, schema is in 1NF.

Since it is in first normal form and there is no partial dependency present in of the relations we can say that the schema is in 2NF.

Also, the schema is in 2NF and there is no transitive dependency of non-prime attributes on super key, so the schema is in 3NF.

The schema is already in 3NF. So there is no need to transform it further(as it goes on to become stricter and stricter) and hence, there is no need to test for lossless join property.

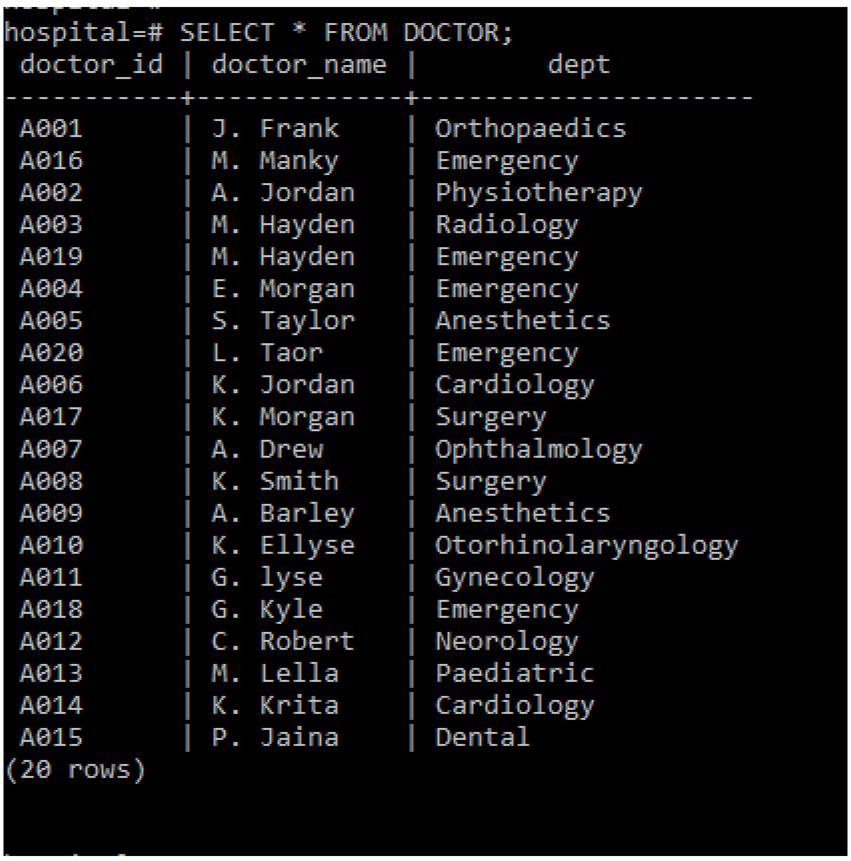
LIST OF RELATIONS



```
           List of relations
 Schema |    Name    | Type  |  Owner
--------+------------+-------+----------
 public | bill       | table | postgres
 public | doctor     | table | postgres
 public | inpatient  | table | postgres
 public | lab        | table | postgres
 public | outpatient | table | postgres
 public | patient    | table | postgres
 public | room       | table | postgres
(7 rows)
```

DDL

```
CREATE TABLE DOCTOR
( Doctor_id VARCHAR(5) NOT NULL,
Doctor_name VARCHAR(30) NOT NULL,
Dept VARCHAR(30) NOT NULL,
PRIMARY KEY (Doctor_id));
```



```
hospital=# SELECT * FROM DOCTOR;
 doctor_id | doctor_name |        dept
-----------+-------------+---------------------
 A001      | J. Frank    | Orthopaedics
 A016      | M. Manky    | Emergency
 A002      | A. Jordan   | Physiotherapy
 A003      | M. Hayden   | Radiology
 A019      | M. Hayden   | Emergency
 A004      | E. Morgan   | Emergency
 A005      | S. Taylor   | Anesthetics
 A020      | L. Taor     | Emergency
 A006      | K. Jordan   | Cardiology
 A017      | K. Morgan   | Surgery
 A007      | A. Drew     | Ophthalmology
 A008      | K. Smith    | Surgery
 A009      | A. Barley   | Anesthetics
 A010      | K. Ellyse   | Otorhinolaryngology
 A011      | G. lyse     | Gynecology
 A018      | G. Kyle     | Emergency
 A012      | C. Robert   | Neorology
 A013      | M. Lella    | Paediatric
 A014      | K. Krita    | Cardiology
 A015      | P. Jaina    | Dental
(20 rows)
```

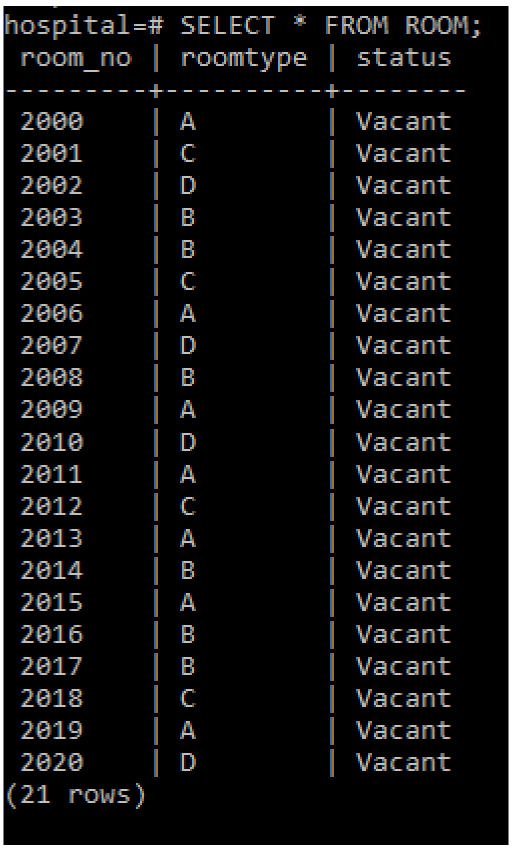
CREATE TABLE ROOM

(Room_No VARCHAR(5) NOT NULL,

RoomType VARCHAR(10) NOT NULL,

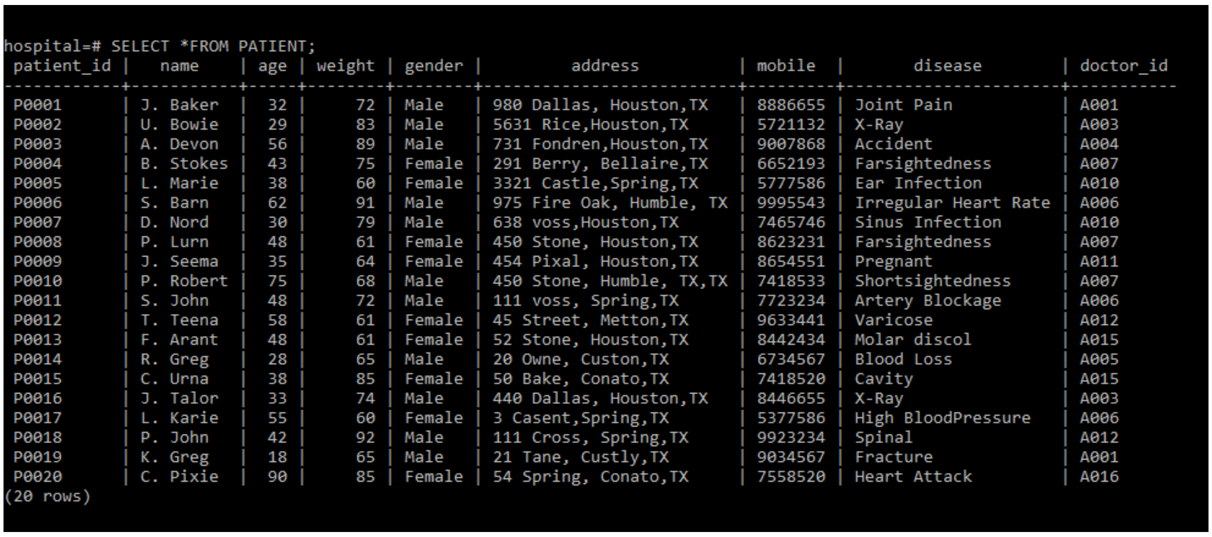
Status VARCHAR(10) NOT NULL,

PRIMARY KEY (Room_No));



```
hospital=# SELECT * FROM ROOM;
 room_no | roomtype | status
---------+----------+--------
 2000    | A        | Vacant
 2001    | C        | Vacant
 2002    | D        | Vacant
 2003    | B        | Vacant
 2004    | B        | Vacant
 2005    | C        | Vacant
 2006    | A        | Vacant
 2007    | D        | Vacant
 2008    | B        | Vacant
 2009    | A        | Vacant
 2010    | D        | Vacant
 2011    | A        | Vacant
 2012    | C        | Vacant
 2013    | A        | Vacant
 2014    | B        | Vacant
 2015    | A        | Vacant
 2016    | B        | Vacant
 2017    | B        | Vacant
 2018    | C        | Vacant
 2019    | A        | Vacant
 2020    | D        | Vacant
(21 rows)
```

```
CREATE TABLE PATIENT

( Patient_id VARCHAR(5) NOT NULL,

Name VARCHAR(20) NOT NULL,

Age INT NOT NULL,

Weight INT NOT NULL,

Gender VARCHAR(10) NOT NULL,

Address VARCHAR(50) NOT NULL,

Mobile INT NOT NULL,

Disease VARCHAR(20) NOT NULL,

Doctor_id VARCHAR(5) NOT NULL,

PRIMARY KEY ( Patient_id),

FOREIGN KEY (Doctor_id) REFERENCES DOCTOR (Doctor_id) ON DELETE

CASCADE ON UPDATE CASCADE);
```



```
hospital=# SELECT *FROM PATIENT;
 patient_id |   name    | age | weight | gender |         address          | mobile  |        disease        | doctor_id
------------+-----------+-----+--------+--------+--------------------------+---------+-----------------------+-----------
 P0001      | J. Baker  |  32 |     72 | Male   | 980 Dallas, Houston,TX   | 8886655 | Joint Pain            | A001
 P0002      | U. Bowie  |  29 |     83 | Male   | 5631 Rice,Houston,TX     | 5721132 | X-Ray                 | A003
 P0003      | A. Devon  |  56 |     89 | Male   | 731 Fondren,Houston,TX   | 9007868 | Accident              | A004
 P0004      | B. Stokes |  43 |     75 | Female | 291 Berry, Bellaire,TX   | 6652193 | Farsightedness        | A007
 P0005      | L. Marie  |  38 |     60 | Female | 3321 Castle,Spring,TX    | 5777586 | Ear Infection         | A010
 P0006      | S. Barn   |  62 |     91 | Male   | 975 Fire Oak, Humble, TX | 9995543 | Irregular Heart Rate  | A006
 P0007      | D. Nord   |  30 |     79 | Male   | 638 voss,Houston,TX      | 7465746 | Sinus Infection       | A010
 P0008      | P. Lurn   |  48 |     61 | Female | 450 Stone, Houston,TX    | 8623231 | Farsightedness        | A007
 P0009      | J. Seema  |  35 |     64 | Female | 454 Pixal, Houston,TX    | 8654551 | Pregnant              | A011
 P0010      | P. Robert |  75 |     68 | Male   | 450 Stone, Humble, TX,TX | 7418533 | Shortsightedness      | A007
 P0011      | S. John   |  48 |     72 | Male   | 111 voss, Spring,TX      | 7723234 | Artery Blockage       | A006
 P0012      | T. Teena  |  58 |     61 | Female | 45 Street, Metton,TX     | 9633441 | Varicose              | A012
 P0013      | F. Arant  |  48 |     61 | Female | 52 Stone, Houston,TX     | 8442434 | Molar discol          | A015
 P0014      | R. Greg   |  28 |     65 | Male   | 20 Owne, Custon,TX       | 6734567 | Blood Loss            | A005
 P0015      | C. Urna   |  38 |     85 | Female | 50 Bake, Conato,TX       | 7418520 | Cavity                | A015
 P0016      | J. Talor  |  33 |     74 | Male   | 440 Dallas, Houston,TX   | 8446655 | X-Ray                 | A003
 P0017      | L. Karie  |  55 |     60 | Female | 3 Casent,Spring,TX       | 5377586 | High BloodPressure    | A006
 P0018      | P. John   |  42 |     92 | Male   | 111 Cross, Spring,TX     | 9923234 | Spinal                | A012
 P0019      | K. Greg   |  18 |     65 | Male   | 21 Tane, Custly,TX       | 9034567 | Fracture              | A001
 P0020      | C. Pixie  |  90 |     85 | Female | 54 Spring, Conato,TX     | 7558520 | Heart Attack          | A016
(20 rows)
```

CREATE TABLE LAB

(Lab_no VARCHAR(5) NOT NULL ,

 Patient_id VARCHAR(5) NOT NULL,

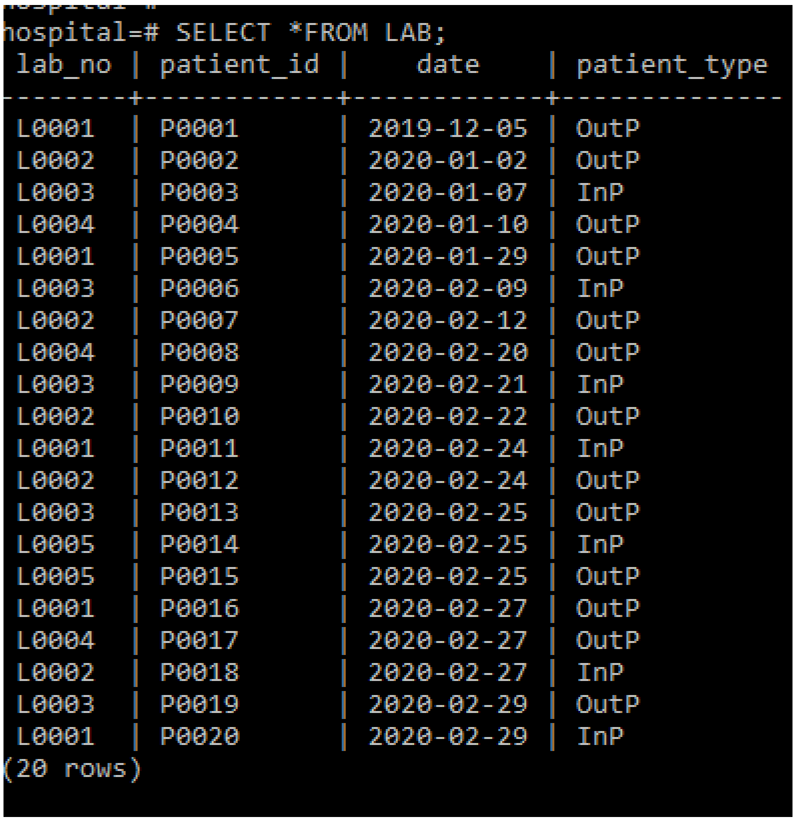
Date DATE NOT NULL,

Patient_type VARCHAR(5) NOT NULL,

PRIMARY KEY (Patient_id),

FOREIGN KEY (Patient_id) REFERENCES PATIENT (Patient_id) ON DELETE CASCADE ON

UPDATE CASCADE);



```
hospital=# SELECT *FROM LAB;
 lab_no | patient_id |    date    | patient_type
--------+------------+------------+--------------
 L0001  | P0001      | 2019-12-05 | OutP
 L0002  | P0002      | 2020-01-02 | OutP
 L0003  | P0003      | 2020-01-07 | InP
 L0004  | P0004      | 2020-01-10 | OutP
 L0001  | P0005      | 2020-01-29 | OutP
 L0003  | P0006      | 2020-02-09 | InP
 L0002  | P0007      | 2020-02-12 | OutP
 L0004  | P0008      | 2020-02-20 | OutP
 L0003  | P0009      | 2020-02-21 | InP
 L0002  | P0010      | 2020-02-22 | OutP
 L0001  | P0011      | 2020-02-24 | InP
 L0002  | P0012      | 2020-02-24 | OutP
 L0003  | P0013      | 2020-02-25 | OutP
 L0005  | P0014      | 2020-02-25 | InP
 L0005  | P0015      | 2020-02-25 | OutP
 L0001  | P0016      | 2020-02-27 | OutP
 L0004  | P0017      | 2020-02-27 | OutP
 L0002  | P0018      | 2020-02-27 | InP
 L0003  | P0019      | 2020-02-29 | OutP
 L0001  | P0020      | 2020-02-29 | InP
(20 rows)
```

CREATE TABLE INPATIENT

(Patient_id VARCHAR(5) NOT NULL,

Room_No VARCHAR(5) NOT NULL,

Adm_Date DATE NOT NULL,

Dis_Date DATE NOT NULL CHECK(Adm_Date<=Dis_Date),

Lab_no VARCHAR(5) NOT NULL,

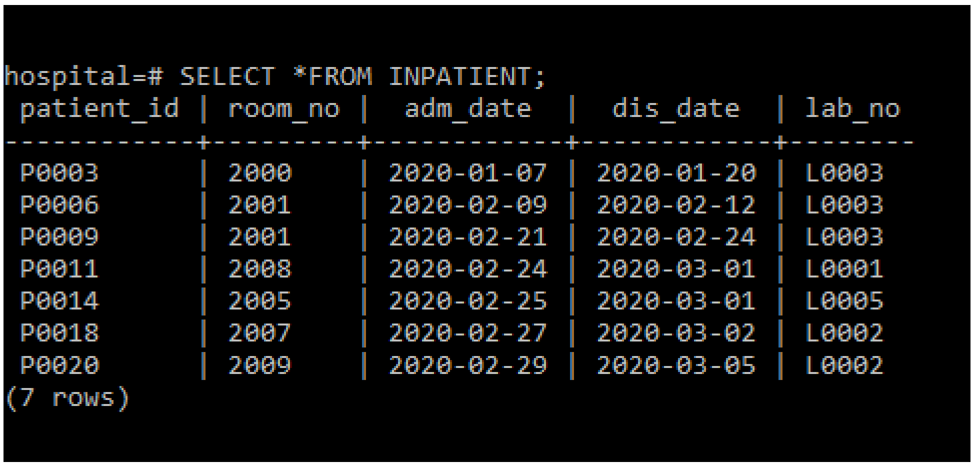
PRIMARY KEY (Patient_id),

FOREIGN KEY (Patient_id) REFERENCES PATIENT (Patient_id) ON DELETE CASCADE ON

UPDATE CASCADE,

FOREIGN KEY (Room_No) REFERENCES ROOM (Room_No) ON DELETE

CASCADE ON UPDATE CASCADE);



```
hospital=# SELECT *FROM INPATIENT;
 patient_id | room_no |  adm_date  |  dis_date  | lab_no
------------+---------+------------+------------+--------
 P0003      | 2000    | 2020-01-07 | 2020-01-20 | L0003
 P0006      | 2001    | 2020-02-09 | 2020-02-12 | L0003
 P0009      | 2001    | 2020-02-21 | 2020-02-24 | L0003
 P0011      | 2008    | 2020-02-24 | 2020-03-01 | L0001
 P0014      | 2005    | 2020-02-25 | 2020-03-01 | L0005
 P0018      | 2007    | 2020-02-27 | 2020-03-02 | L0002
 P0020      | 2009    | 2020-02-29 | 2020-03-05 | L0002
(7 rows)
```

CREATE TABLE OUTPATIENT

(Patient_id VARCHAR(5) NOT NULL,

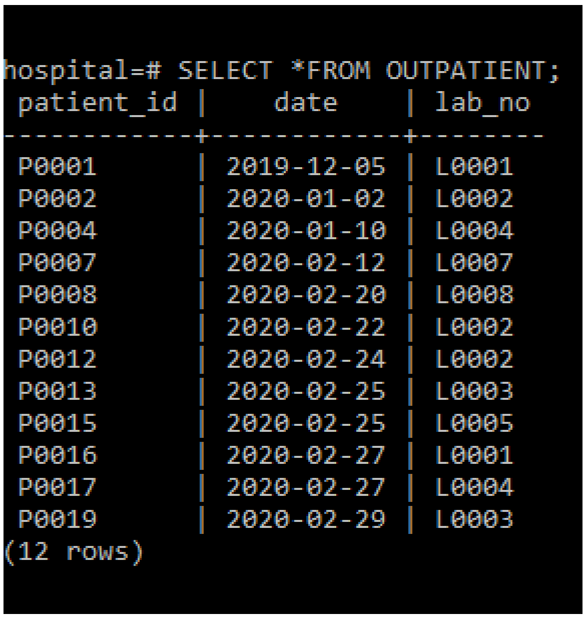
Date DATE NOT NULL,

Lab_no VARCHAR(5) NOT NULL,

PRIMARY KEY (Patient_id),

FOREIGN KEY (Patient_id) REFERENCES PATIENT (Patient_id) ON DELETE CASCADE ON

UPDATE CASCADE);



```
hospital=# SELECT *FROM OUTPATIENT;
 patient_id |    date    | lab_no
------------+------------+--------
 P0001      | 2019-12-05 | L0001
 P0002      | 2020-01-02 | L0002
 P0004      | 2020-01-10 | L0004
 P0007      | 2020-02-12 | L0007
 P0008      | 2020-02-20 | L0008
 P0010      | 2020-02-22 | L0002
 P0012      | 2020-02-24 | L0002
 P0013      | 2020-02-25 | L0003
 P0015      | 2020-02-25 | L0005
 P0016      | 2020-02-27 | L0001
 P0017      | 2020-02-27 | L0004
 P0019      | 2020-02-29 | L0003
(12 rows)
```

CREATE TABLE BILL

(Bill_no VARCHAR(5) NOT NULL,

Patient_id VARCHAR(5) NOT NULL UNIQUE,

Patient_type VARCHAR(10) NOT NULL,

Date DATE NOT NULL,

Lab_charge INT NOT NULL,

Doctor_charge INT NOT NULL,

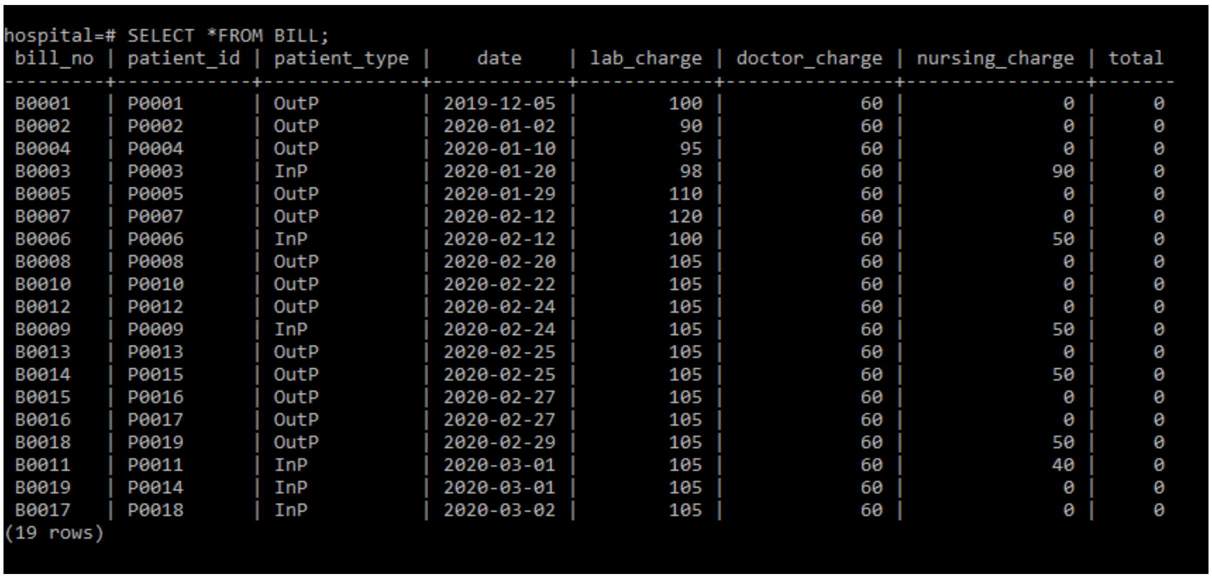
Nursing_charge INT NOT NULL,

Total INT,

PRIMARY KEY (Bill_no),

FOREIGN KEY (Patient_id) REFERENCES PATIENT (Patient_id) ON DELETE CASCADE ON

UPDATE CASCADE);



```
hospital=# SELECT *FROM BILL;
 bill_no | patient_id | patient_type |    date    | lab_charge | doctor_charge | nursing_charge | total
---------+------------+--------------+------------+------------+---------------+----------------+-------
 B0001   | P0001      | OutP         | 2019-12-05 |        100 |            60 |              0 |     0
 B0002   | P0002      | OutP         | 2020-01-02 |         90 |            60 |              0 |     0
 B0004   | P0004      | OutP         | 2020-01-10 |         95 |            60 |              0 |     0
 B0003   | P0003      | InP          | 2020-01-20 |         98 |            60 |             90 |     0
 B0005   | P0005      | OutP         | 2020-01-29 |        110 |            60 |              0 |     0
 B0007   | P0007      | OutP         | 2020-02-12 |        120 |            60 |              0 |     0
 B0006   | P0006      | InP          | 2020-02-12 |        100 |            60 |             50 |     0
 B0008   | P0008      | OutP         | 2020-02-20 |        105 |            60 |              0 |     0
 B0010   | P0010      | OutP         | 2020-02-22 |        105 |            60 |              0 |     0
 B0012   | P0012      | OutP         | 2020-02-24 |        105 |            60 |              0 |     0
 B0009   | P0009      | InP          | 2020-02-24 |        105 |            60 |             50 |     0
 B0013   | P0013      | OutP         | 2020-02-25 |        105 |            60 |              0 |     0
 B0014   | P0015      | OutP         | 2020-02-25 |        105 |            60 |             50 |     0
 B0015   | P0016      | OutP         | 2020-02-27 |        105 |            60 |              0 |     0
 B0016   | P0017      | OutP         | 2020-02-27 |        105 |            60 |              0 |     0
 B0018   | P0019      | OutP         | 2020-02-29 |        105 |            60 |             50 |     0
 B0011   | P0011      | InP          | 2020-03-01 |        105 |            60 |             40 |     0
 B0019   | P0014      | InP          | 2020-03-01 |        105 |            60 |              0 |     0
 B0017   | P0018      | InP          | 2020-03-02 |        105 |            60 |              0 |     0
(19 rows)
```

Triggers

1.It calculates the Total amount of BILL to be paid

CREATE OR REPLACE FUNCTION total_amt()

RETURNS TRIGGER AS

$BODY$

BEGIN

NEW.Total=(NEW.Lab_charge+NEW.Doctor_charge+NEW.Nursing_charge);

```
RETURN NEW;

END;

$BODY$

LANGUAGE plpgsql;

CREATE TRIGGER TOTAL

BEFORE INSERT OR UPDATE ON BILL

FOR EACH ROW

EXECUTE PROCEDURE total_amt();
```

2. It changes the status of the ROOM as soon as the BILL is paid.

```
CREATE OR REPLACE FUNCTION vacant_room()

RETURNS TRIGGER AS

$BODY$

BEGIN

UPDATE ROOM SET Status='Vacant'

WHERE RoomNo=( SELECT RoomNo FROM INPATIENT WHERE

INPATIENT.Pid=NEW.Pid);

RETURN NEW;

END;

$BODY$

LANGUAGE plpgsql;

CREATE TRIGGER Status_update

AFTER INSERT OR UPDATE ON BILL

FOR EACH ROW

WHEN(NEW.PATIENT_TYPE='InP')

EXECUTE PROCEDURE vacant_room();
```

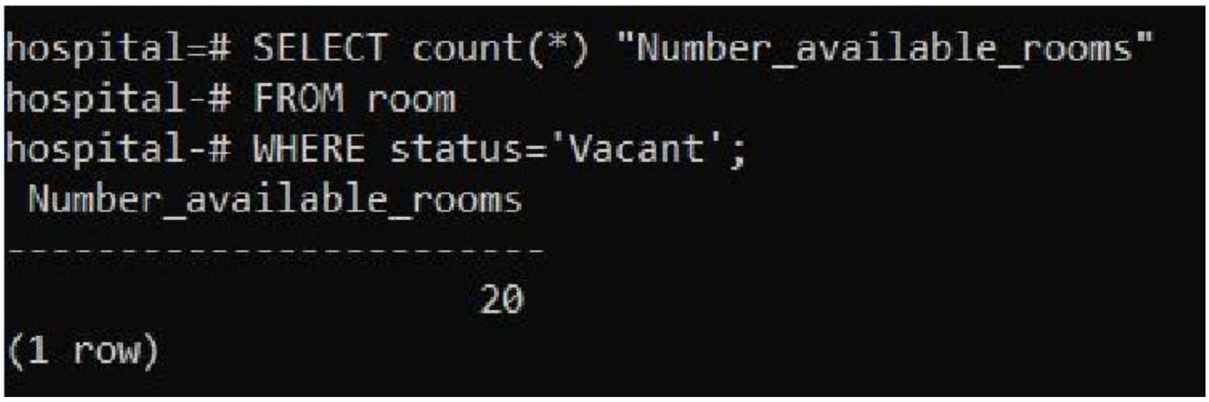
SQL Queries

1. To find the number of Vacant rooms.

SELECT COUNT(*) "Number_available_rooms"

FROM ROOM

WHERE Status='Vacant';



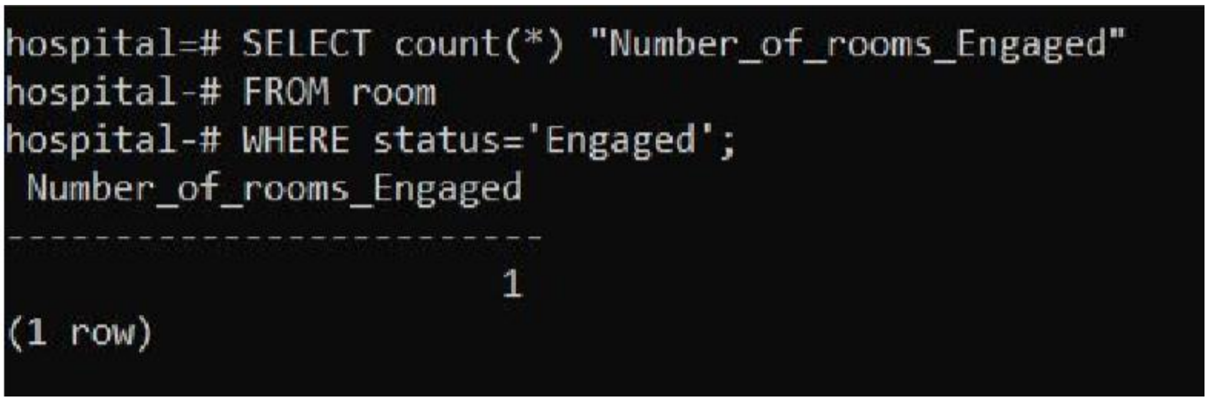
```
hospital=# SELECT count(*) "Number_available_rooms"
hospital-# FROM room
hospital-# WHERE status='Vacant';
 Number_available_rooms
------------------------
                     20
(1 row)
```

2. To find the number of Engaged rooms.

SELECT COUNT(*) "Number_of_rooms_Engaged"

FROM ROOM

WHERE Status='Engaged';

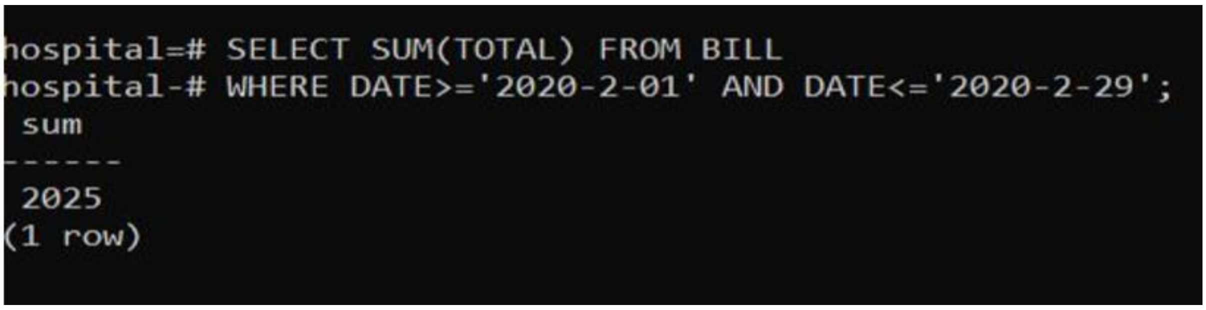


```
hospital=# SELECT count(*) "Number_of_rooms_Engaged"
hospital-# FROM room
hospital-# WHERE status='Engaged';
 Number_of_rooms_Engaged
-------------------------
                       1
(1 row)
```

3.To find the Total collection in the month of February, this year.

SELECT SUM(TOTAL) FROM BILL

WHERE DATE>='2020-2-01' AND DATE<='2020-2-29';



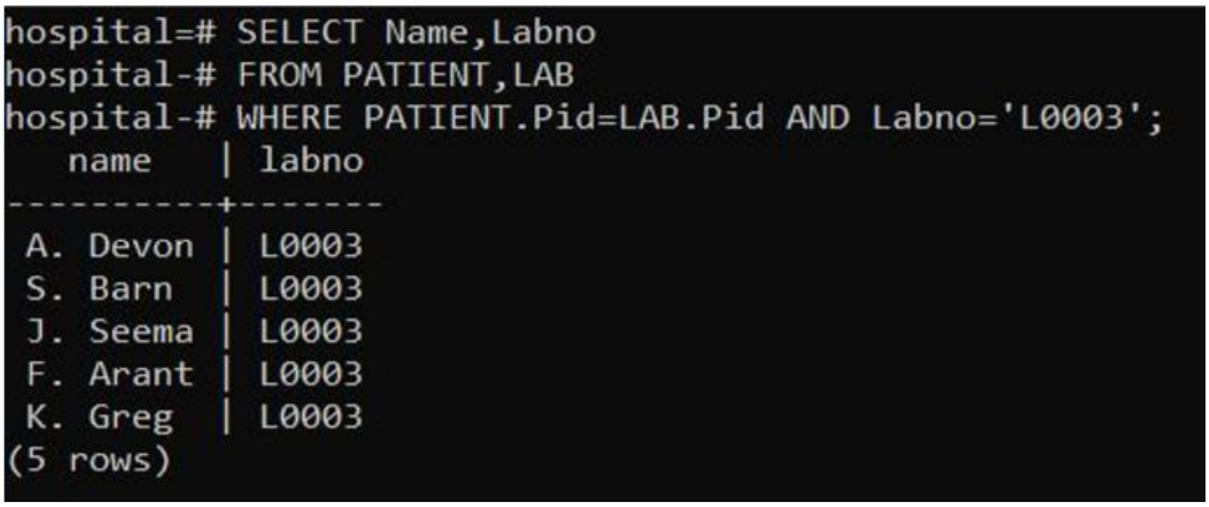
```
hospital=# SELECT SUM(TOTAL) FROM BILL
hospital-# WHERE DATE>='2020-2-01' AND DATE<='2020-2-29';
 sum
------
 2025
(1 row)
```

4. To find the Name of PATIENTs in LAB=’L003’

SELECT Name,Labno

FROM PATIENT, LAB

WHERE PATIENT.Pid=LAB.Pid AND Labno='L0003';



```
hospital=# SELECT Name,Labno
hospital-# FROM PATIENT,LAB
hospital-# WHERE PATIENT.Pid=LAB.Pid AND Labno='L0003';
   name    | labno
----------+-------
 A. Devon | L0003
 S. Barn  | L0003
 J. Seema | L0003
 F. Arant | L0003
 K. Greg  | L0003
(5 rows)
```

5. To find what Disease the PATIENT OF Room Number ‘2009’ has.

SELECT Disease

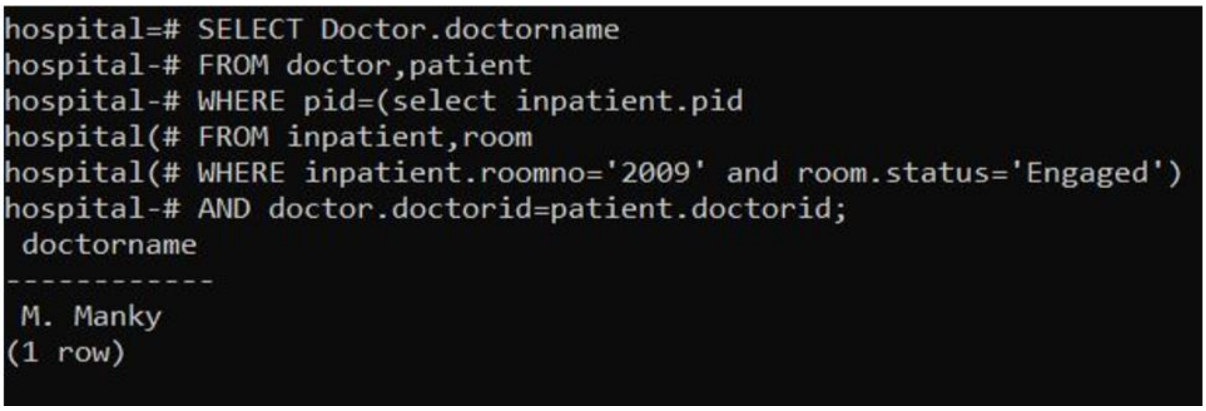
FROM patient

WHERE Pid=(SELECT INPATIENT.Pid

FROM INPATIENT, ROOM

WHERE INPATIENT.roomNo='2009' AND

ROOM.Status='Engaged');



```
hospital=# SELECT Doctor.doctorname
hospital-# FROM doctor,patient
hospital-# WHERE pid=(select inpatient.pid
hospital(# FROM inpatient,room
hospital(# WHERE inpatient.roomno='2009' and room.status='Engaged')
hospital-# AND doctor.doctorid=patient.doctorid;
 doctorname
------------
 M. Manky
(1 row)
```

6. To find what Disease the PATIENT OF Room Number ‘2009’ has.

SELECT Disease

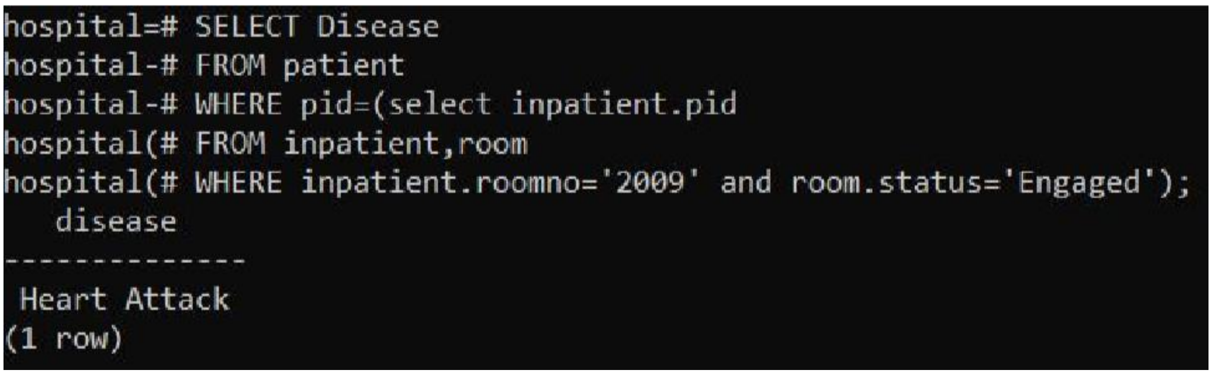
FROM patient

WHERE Pid=(SELECT INPATIENT.Pid

FROM INPATIENT, ROOM

WHERE INPATIENT.roomNo='2009' AND

ROOM.Status='Engaged');



```
hospital=# SELECT Disease
hospital-# FROM patient
hospital-# WHERE pid=(select inpatient.pid
hospital(# FROM inpatient,room
hospital(# WHERE inpatient.roomno='2009' and room.status='Engaged');
   disease
--------------
 Heart Attack
(1 row)
```

Conclusion

Project report on Hospital Database Management System gives us a picture of how the data can be managed at an institutional level like hospital, schools, colleges and other bodies, it gives us knowledge about different aspects to keep in mind while maintaining records. We learnt about Schema, Functional Dependencies, Normalization, Constraints, nested and aggregate queries and triggers.