DBMS Project Report

PES University

Database Management Systems

UE18CS252

Hospital database

Submitted By

PES1201801937 Hemanth Alva R

CSE ' J section

Summary:

In this project, we have developed a database on a hospital. The main aim is to develop a database that can store details related to patients, doctors, interns and staffs working in the hospital. This database stores the data of people who have been consulted and admitted in the hospital, their personal details and the floor in which they are admitted in the hospital. And doctor and staff data consists of their id, name, and respective department. This project deals with creating a schema in the database, transaction details, queries etc.

We derive functional dependency in this model from which we get the third normal form. Using this third normal form we derive the schema of the database and define the tables of the database. There are 7 tables defined department, doctor, patient, room, staff, intern and patient_admitted.

The defined trigger stores the patient id and patient admission time in the table patient_admitted using the data inserted from patient table. Here in this project we define queries to find the number of doctors having more than one patient and the number of doctors who don't have interns under them. Doctor with the maximum number of patients. The list of doctors with patients using outer join. List of rooms with department name and the department with maximum number of staff working in that department.

The database helps us record important hospital details regarding the patients, the doctors and the staff working there. It tells us which patient is under which doctor's care and in which room the patient has been admitted and it also tells about the interns who work under a certain doctor. It just given an overall idea how a hospital is managed.

Introduction 1

Data Model 2

FD and Normalization 4

DDL 6

Triggers 13

SQL Queries 14

Conclusion 19

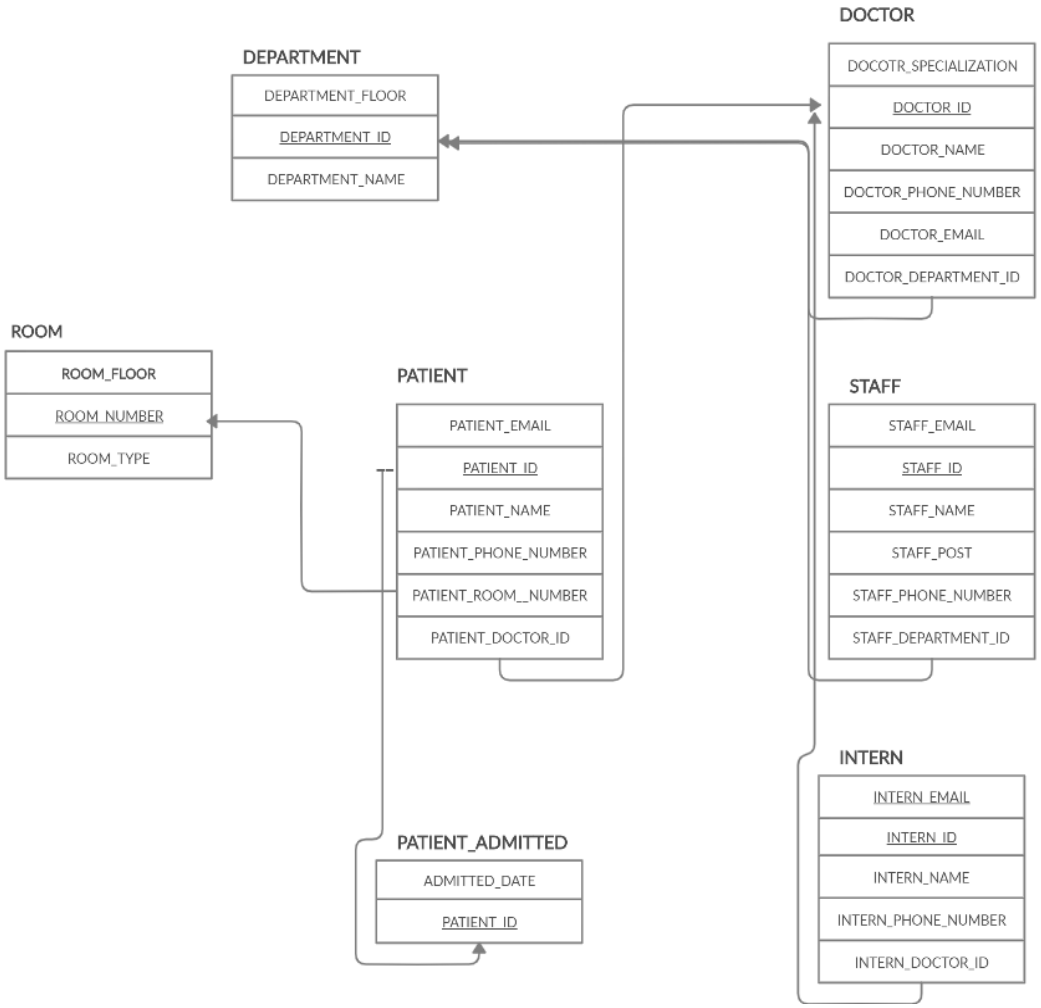
Introduction

This hospital data base is created to store data related to hospital and to get a clear picture about a hospital is managed. This database has relation regarding the patient, the rooms they are admitted in, doctors and the staff working in the hospital. Along with this it also stores data related to the interns working under the doctors. The database also keeps track of the patients entry date and time and the rooms they occupied. It also tells us about the kind of departments present and kind of rooms there.

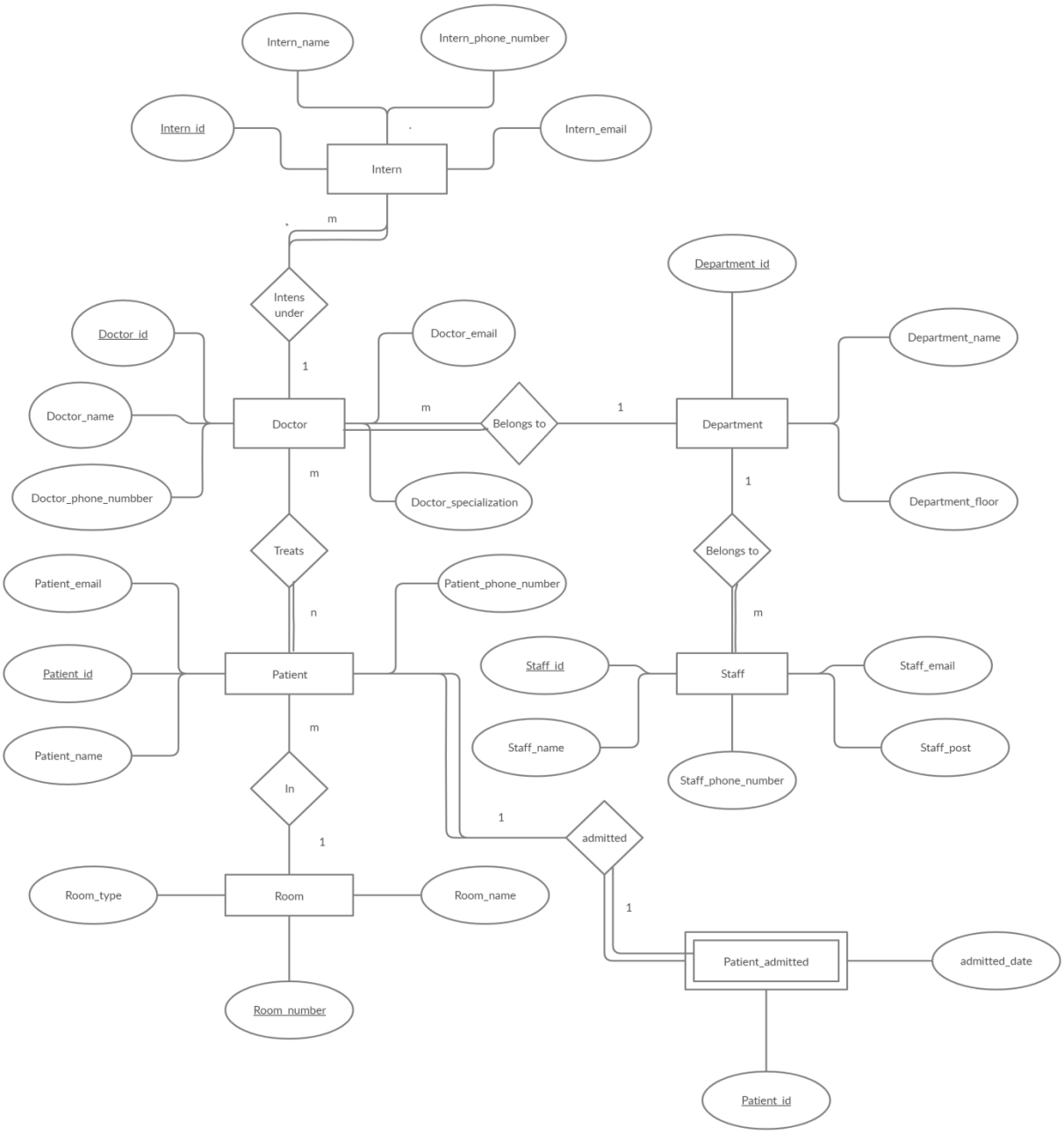
The objective of this is to create a basic ER diagram and a schema, and from which derive a set of functional dependencies. Using this we will be performing a set of queries using psql. This database consists of 7 tables namely department, doctor, room, patient, staff, intern and patient_admitted with primary keys department_id, doctor_id, room_number, patient_id, staff_id, intern_id and patient_id respectively.

Data Model

Schema:



ER diagram:



The following primary keys:

Department_id, doctor_id, patient_id, room_number, intern_id, staff_id

Any combination of the above attribute can be candidate keys. These are all prime attributes and from above functional dependencies we can see that these are all possible keys.

Patient_admitted is a week entity because its primary key is a derived from the entity (table) patient.

FD and Normalization

A **functional dependency** (FD) is a relationship between two attributes, typically between the Primary key and other non-key attributes within a table.

Functional dependency:

- Department_id -> department_name,department_floor
- Doctor_id -> doctor_name, doctor_phone_number, doctor_email, doctor_specialization, doctor_department_id
- Room_number-> room_type, room_floor
- Patient_id-> patient_name,patient_email, patient_phone_number, patient_room_number, patient_doctor_id
- Staff_id -> staff_name, staff_post, staff_phone_number, staff_email, staff_department_id
- Intern_id -> intern_name, intern_email, intern_phone_number, intern_doctor_id
- Patient_id-> admitted_date

For 1NF, we need atomic data. In our database all columns are single valued , hence all the data is atomic. Hence our functional dependencies are already in 1NF.

For 2NF, we would need all non key attributes to be depended on only prime attribute. Since in this database all entities (tables) have only 1 primary key, therefore the database is in 2NF.

For 3NF, since the relation is in 2NF and there is no transitive dependency for non-prime attributes as well, the database is in 3NF.
In the doctor table (relation), by adding the column dortor_department_name and changing the primary key to doctor_id and department_id, it makes the relation to violate 2NF :

Doctor_id, doctor_department_id -> doctor_name, doctor_phone_number, doctor_email, doctor_specialization, doctor_department_name.

Here doctor_department_name (non key attribute) is partially depended on the primary key (it's only dependent on doctor_department_id) .
The 2NF obtained is as follows:

- Doctor_id -> doctor_name, doctor_phone_number, doctor_email, doctor_specialization, doctor_department_id
- Doctor_department_id -> doctor_department_name.

Thus we have obtained FD in 2NF. And the relation is already in 3NF, as we have no transitive dependencies.

Lossless join:

Decomposition is lossless if it is feasible to reconstruct relation R from decomposed tables using Joins. This is the preferred choice.
We shall consider the decomposed relations –

- R: (Doctor_id, doctor_name, doctor_phone_number, doctor_email, doctor_specialization, doctor_department_id).

- R1: (Doctor_id, doctor_name, doctor_phone_number, doctor_email, doctor_specialization)
- R2: (Doctor_department_id, doctor_id)

R: Doctor:

Doctor_id	Doctor_name	Doctor_phone_number	Doctor_email	Doctor_specialization	Doctor_department_id
11	Ramesh	9341235687	ramesh27@gmail.com	Neurologist	3
32	Suresh	8314667890	Suresh1928@gmail.com	Gynecologist	4

R1: Doctor Info:

Doctor_id	Doctor_name	Doctor_phone_number	Doctor_email	Doctor_specialization
11	Ramesh	9341235687	ramesh27@gmail.com	Neurologist
32	Suresh	8314667890	Suresh1928@gmail.com	Gynecologist

R2: Department Info:

Doctor_department_id	Doctor_id
3	11
4	32

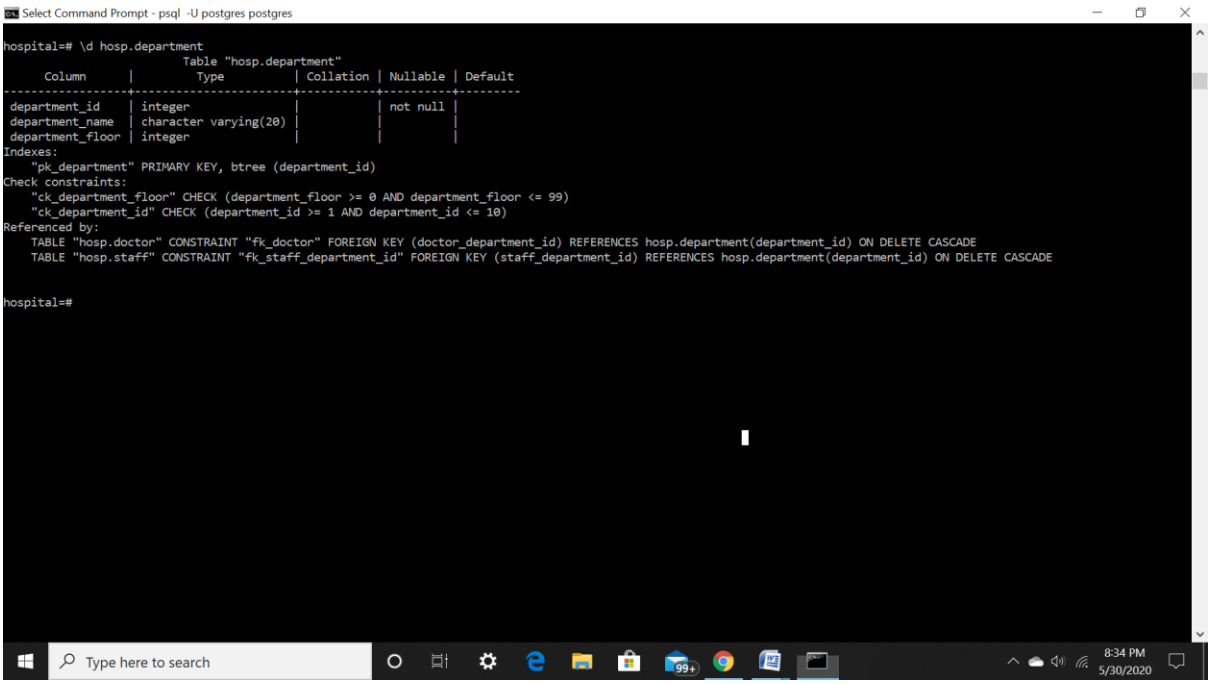
Now on natural join of the above two table R1 and R2:
The result will be:

Doctor_id	Doctor_name	Doctor_phone_number	Doctor_email	Doctor_specialization	Doctor_department_id
11	Ramesh	9341235687	ramesh27@gmail.com	Neurologist	3
32	Suresh	8314667890	Suresh1928@gmail.com	Gynecologist	4

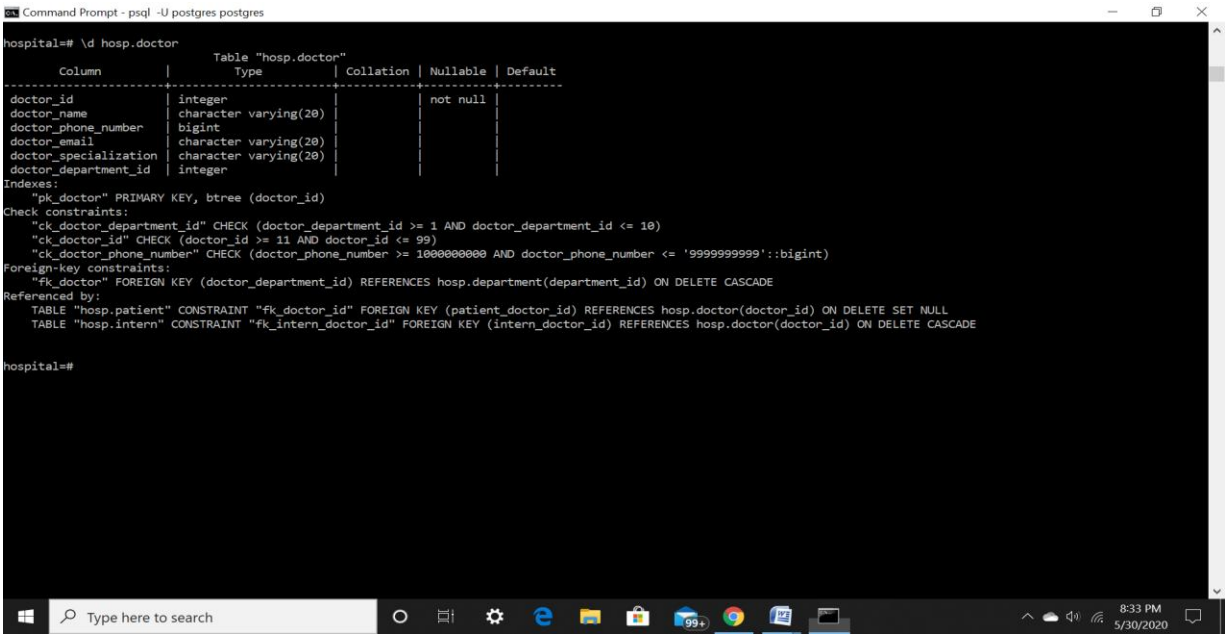
DDL

Create table:
Create schema hosp;

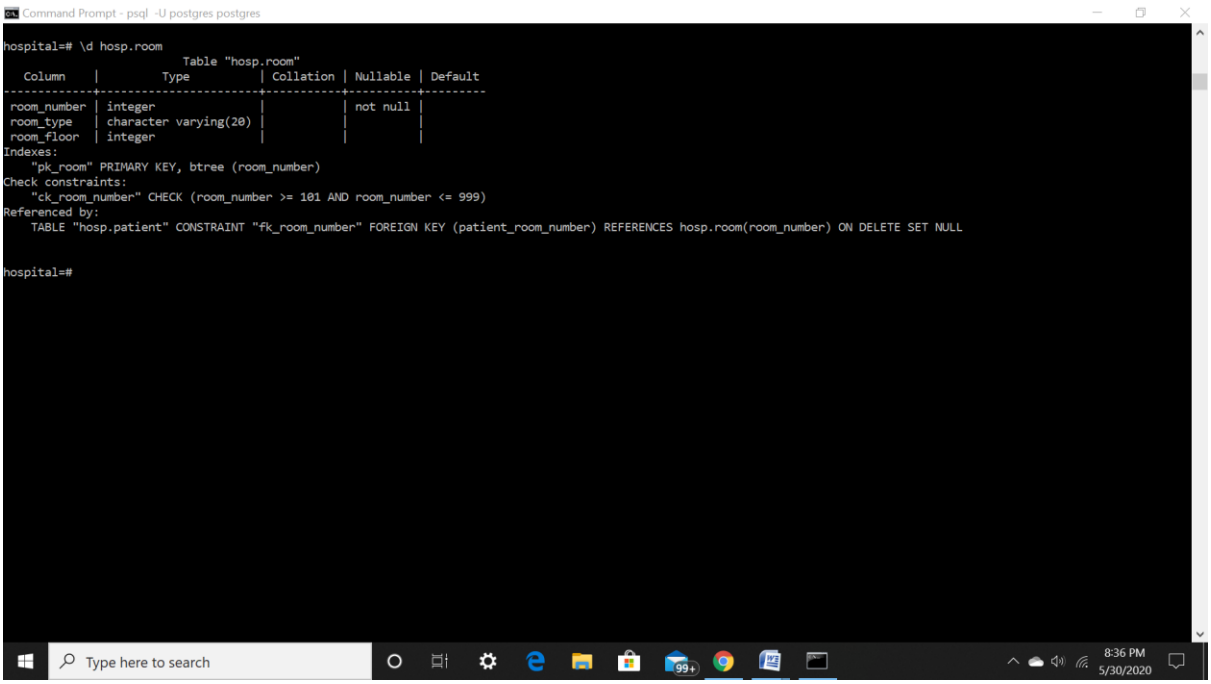
Create table hosp.department(
Department_id integer,
Department_name varchar(20),
Department_floor integer,
Constraint pk_department primary key(department_id),
Constraint ck_department_floor check (department_floor between 0 and 99),
Constraint ck_department_id check(department_id between 0 and 10));



Create table hosp.doctor(
Doctor_id integer,
Dotor_name varchar(20).
Doctor_phone_number bigint,
Doctor_email varchar(20),
Doctor_specialization varchar(20),
Doctor_department_id integer,
Constraint pk_doctor primary key(doctor_id),
Constraint fk_doctor foreign key(doctor_department_id) references
hosp.department(department_id) on delete cascade,
Constraint ck_doctor_id check(doctor_id between 11 and 99),
Constraint ck_doctor_ department_id check(doctor_department_id between 0 and 10)
Constraint ck_doctor_phone_number check(doctor_phone_number between 1000000000
and 9999999999));

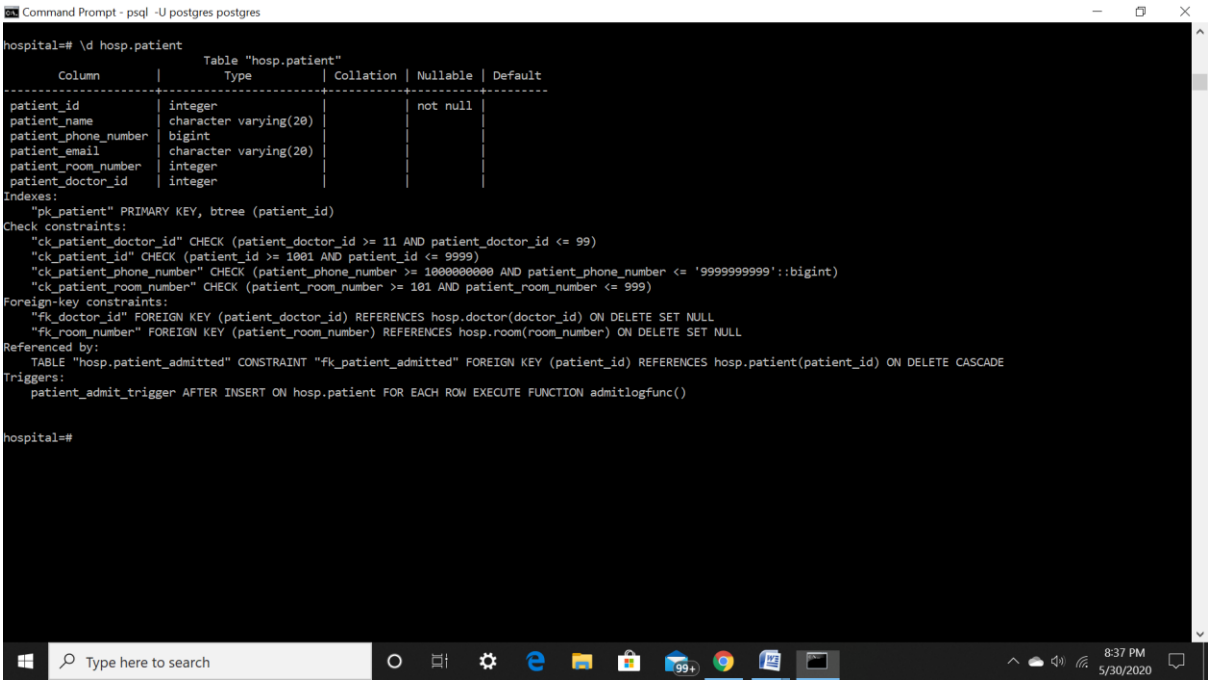


Create table hosp.room(
Room_number integer,
Room_type varchar(20),
Room_floor integer,
Constraint pk_room primary key(room_number),
Constraint ck_room_number check(room_number between 101 and 999));

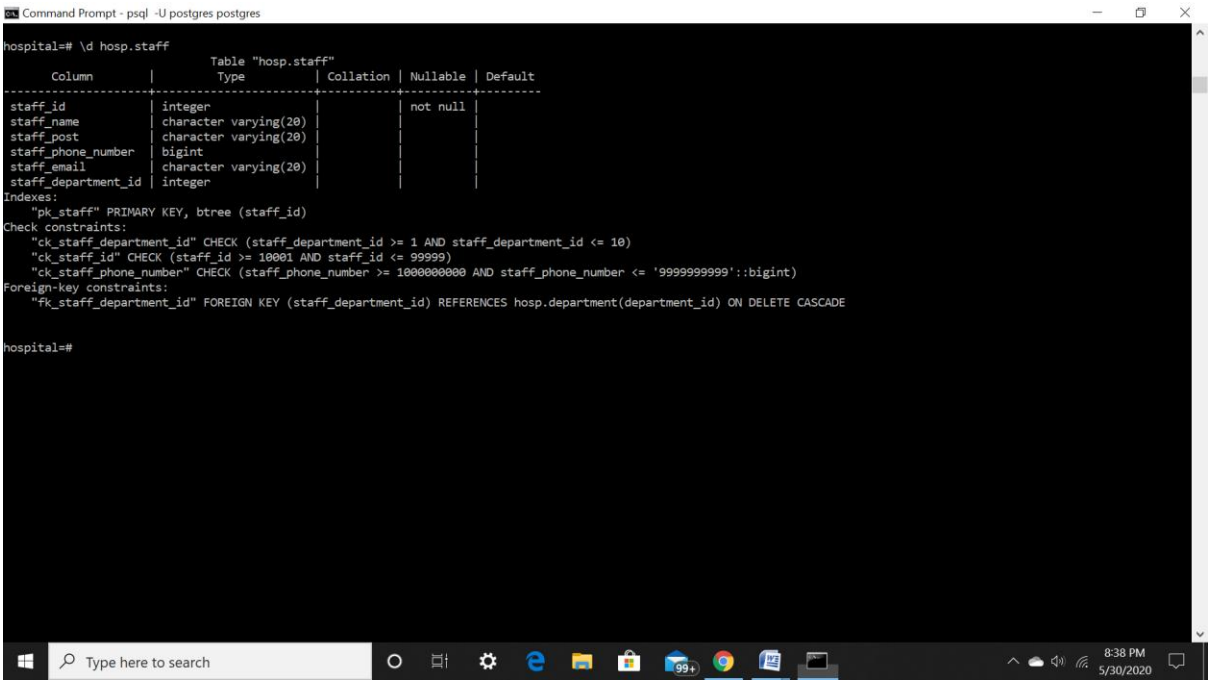


Create table hosp.patient(
Patient_id integer,
Patient_name varchar(20),
Patient_phone_name bigint,
Patient_email varchar(20),
Patient_room_number integer,

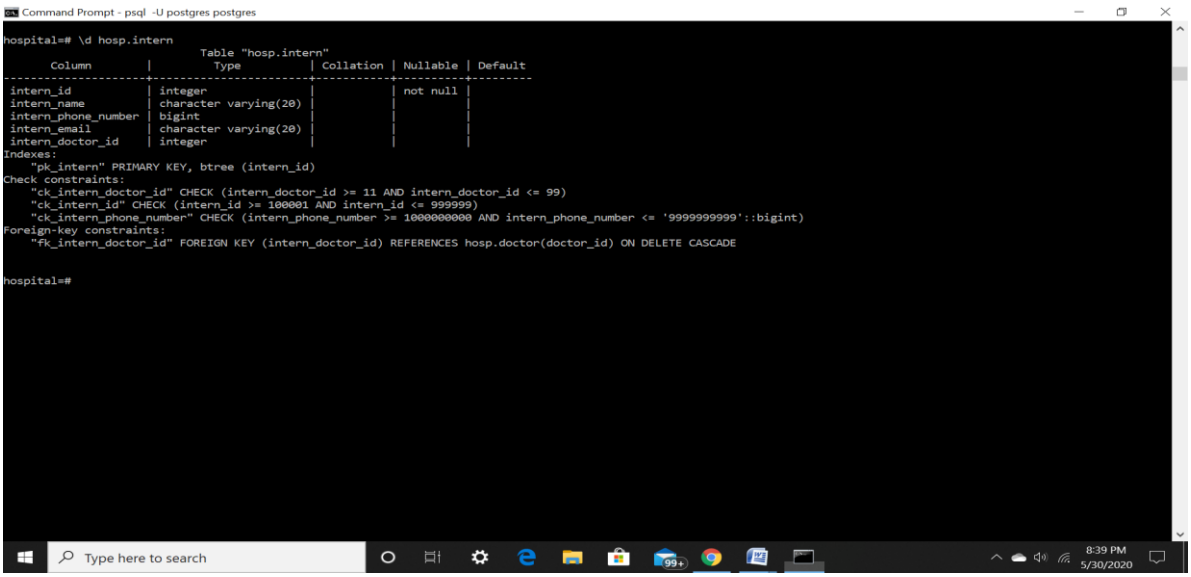
Patient_doctor_id integer,
Constraint pk_patient primary key(patient_id),
Constraint fk_room_number foreign key(patient_room_number) references hosp.room(room_number) on delete set null,
Constraint fk_doctor_id foreign key(patient_doctor_id) references hosp.doctor(doctor_id) on delete set null,
Constraint ck_patient_id check(patient_id between 1001 to 9999),
Constraint ck_patient_room_number check(patient_room_number between 101 and 999),
Constraint ck_patient_doctor_id check(patient_doctor_id between 11 and 99),
Constraint ck_patient_phone_number check(patient_phone_number between 1000000000 and 9999999999));



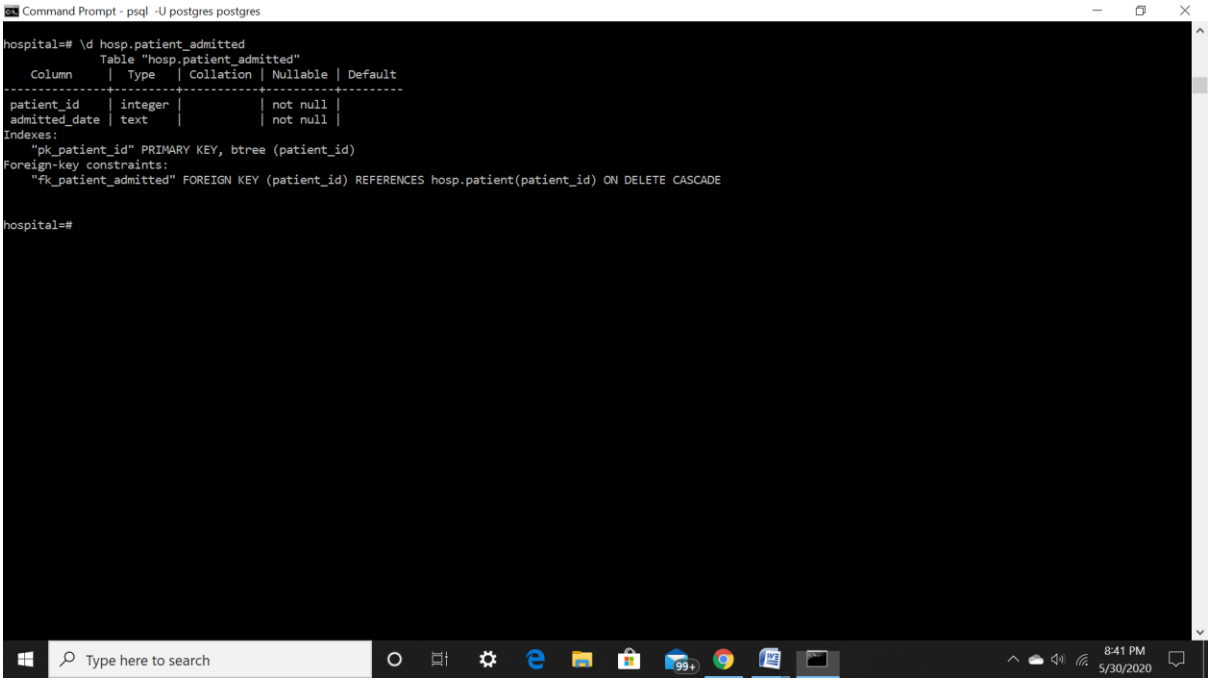
Create table hosp.staff(
Staff_id integer,
Staff_name varchar(20),
Staff_post varchar(20),
Staff_phone_number bigint,
Staff_email varchar(20),
Staff_department_id integer,
Constraint pk_staff primary key(staff_id),
Constraint fk_staff_department_id foreign key(staff_department_id) references hosp.department(department_id) on delete cascade,
Constraint ck_staff_id check(staff_id between 10001 and 99999),
Constraint ck_staff_department_id check(staff_department_id between 0 and 10),
Constraint ck_staff_phone_number check(staff_phone_number between 1000000000 and 9999999999));



Create table hosp.intern(
Intern_id integer,
Intern_name varchar(20),
Intern_phone_number bigint,
Intern email varchar(20),
Intern doctor_id integer,
Constraint pk_intern primary key (intern_id),
Constraint fk_intern_doctor_id foreign key(intern_doctor_id) references hosp.doctor(doctor_id) on delete cascade,
Constraint ck_intern_id check (intern_id between 100001 and 999999),
Constraint ck_intern_doctor_id check(intern_doctor_id between 11 and 99),
Constraint ck_intern_phone_number check(intern_phone_number between 1000000000 and 9999999999));



Create table patient_admitted(
Patient_id int not null,
Admitted_date text not null,
Constraint fk_patient_admitted foreign key(patient_id) references hosp.patient(patient_id) on delete cascade);



Insert values:

Insert into hosp.department values
(1, 'Emergency', 0),
(2, 'Cardiology',1),
(3, 'Neurology',2),
(4,'Gynaecology', 3),
(5, 'Hematology', 4);

Insert into hosp.doctor values
(11, 'Ramesh', 9341235687, 'ramesh27@gmail.com', 'Neurologist', 3),
(32, 'Suresh', 8314667890, 'suresh1928@gmail.com', 'Gynaecologist', 4),
(36, 'Naveen', 9314562349, 'naveenraja@gmail.com', 'Physician', 1),
(48,'Ramesh', 6123789012, 'rameshgoyy@gmail.com', 'Cardiologist', 2),
(50, 'Himaja', 7624156738, 'himajinath@gmail.com', 'Physician', 5),
(27, 'Manoj', 6622459072, 'manojkarik@gmail.com', 'Surgeon', 1);

Insert into hosp.room values
(102, 'General', 0)
(204, 'Delux', 1),
(105, 'General', 0),
(417, 'Special', 3),
(215, 'Delux ', 1),
(402, 'Special', 3),

(312, ‘General’, 2);

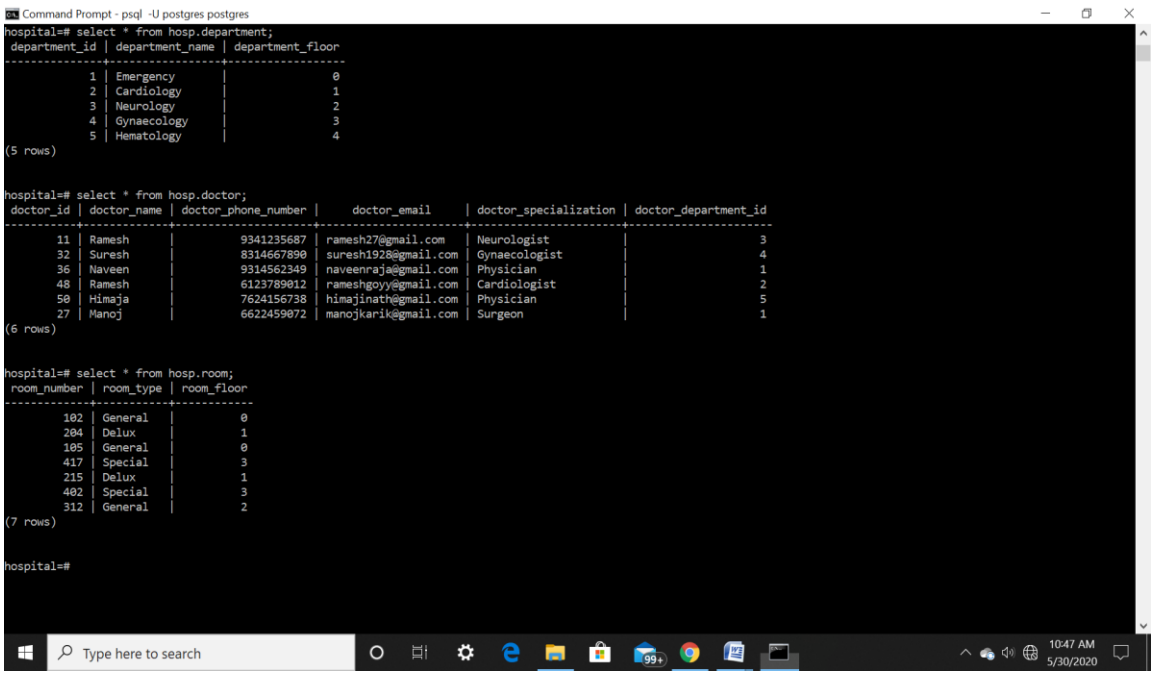
Insert into hosp.patient values
(2253, ‘Gaurav’, 6259262592, ’gau26@gmail.com’, 102, 36),
(3259, ’Meghana’, 7217700098, ‘megha01@gmail.com’, 417, 32),
(6934, ‘Sonali’, 8317678909,’megha00@gmail.com’, 105, 36),
(4203, ’Sonu’, 9342756499, ‘sonux2@gmail.com’, 312, 48),
(3542, ‘Sreekar’, 9845155770, ‘sreekar@gmail.com’, 402, 50),
(3341, ‘Sreejash’, 8313388833, ’sreej@gmail.com’, 215, 27);

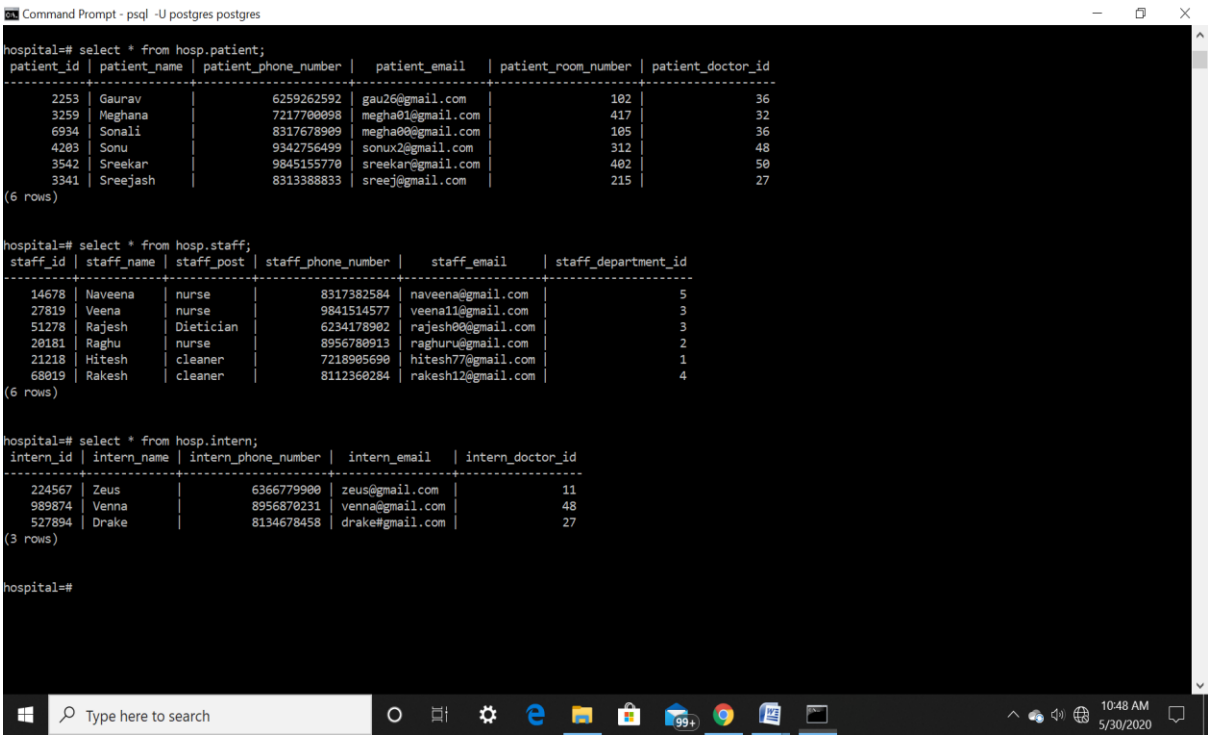
Insert into hosp.staff values
(14678, ‘Naveena’, ‘nurse’, 8317382584, ‘naveena@gmail.com’, 5),
(27819, ’Veena’, ‘nurse’, 9841514577, ‘veena11@gmail.com’, 3),
(51278, ‘Rajesh’, ‘Dietician’, 6234178902, ‘rajesh00@gmail.com’, 3),
(20181, ‘Raghu’, ‘nurse’, 8956780913, ’raghuru@gmail.com’, 2),
(21218, ‘Hitesh’, ‘cleaner’, 7218905690, ‘hitesh77@gmail.com’, 1),
(68019, ‘Rakesh’, ‘cleaner’, 8112360284, ‘rakesh12@gmail.com’, 4);

Insert into hosp.intern values
(224567, ‘Zeus’, 6366779900, ’zeus@gmail.com’,11),
(989874, ‘Venna’, 8956870231, ‘venna@gmail.com’, 48),
(527894, ‘Drake’, 8134678458, ‘drake#gmail.com’, 27);

The data inserted:

1. Select * from hosp.department;
2. Select * from hosp.doctor;
3. Select * from hosp.room;
4. Select * from hosp.patient;
5. Select * from hosp.interns;
6. Select * from hosp.staff;




Command Prompt - psql -U postgres postgres
hospital=# select * from hosp.patient;
patient_id | patient_name | patient_phone_number | patient_email | patient_room_number | patient_doctor_id
2253 | Gaurav | 6259262592 | gau26@gmail.com | 102 | 36
3259 | Meghana | 7217700098 | megha01@gmail.com | 417 | 32
6934 | Sonali | 8317678909 | megha00@gmail.com | 105 | 36
4203 | Sonu | 9342756499 | sonux2@gmail.com | 312 | 48
3542 | Sreekar | 9845155770 | sreekar@gmail.com | 402 | 50
3341 | Sreejash | 8313388833 | sreej@gmail.com | 215 | 27
(6 rows)
hospital=# select * from hosp.staff;
staff_id | staff_name | staff_post | staff_phone_number | staff_email | staff_department_id
14678 | Naveena | nurse | 8317382584 | naveena@gmail.com | 5
27819 | Veena | nurse | 9841514577 | veena11@gmail.com | 3
51278 | Rajesh | Dietician | 6234178902 | rajesh00@gmail.com | 3
20181 | Raghu | nurse | 8956780913 | raghuru@gmail.com | 2
21218 | Hitesh | cleaner | 7218905690 | hitesh77@gmail.com | 1
68019 | Rakesh | cleaner | 8112360284 | rakesh12@gmail.com | 4
(6 rows)
hospital=# select * from hosp.intern;
intern_id | intern_name | intern_phone_number | intern_email | intern_doctor_id
224567 | Zeus | 6366779900 | zeus@gmail.com | 11
989874 | Venna | 8956870231 | venna@gmail.com | 48
527894 | Drake | 8134678458 | drake#gmail.com | 27
(3 rows)
hospital=#
Type here to search
10:48 AM
5/30/2020

Triggers

The created trigger helps in entering the data into the patient_admitted table, that is when a patient's data is been entered into the patient table the current time and patient_id is stored in the table.

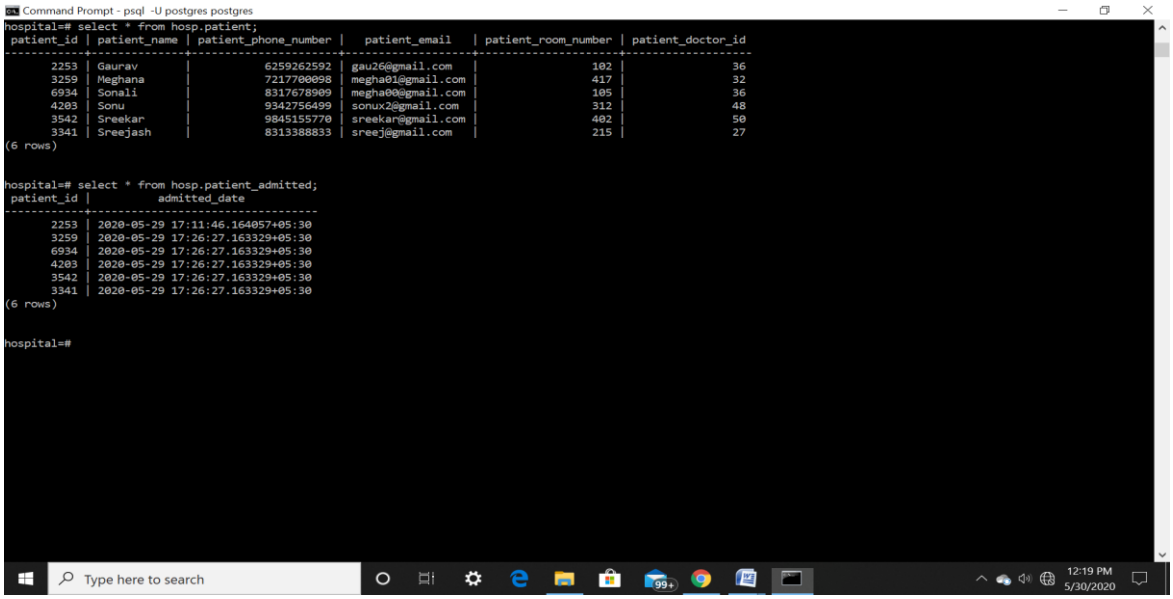
```
CREATE OR REPLACE FUNCTION admitlogfunc( ) RETURNS TRIGGER AS
$example_table$
        BEGIN
                INSERT INTO HOSP.PATIENT
_ADMITTED(PATIENT_ID,ADMITTED_DATE)
VALUES(new.PATIENT_ID,current_timestamp);
                RETURN NEW;
        END;
$example_table$ LANGUAGE plpgsql

CREATE TRIGGER patient_admit_trigger AFTER INSERT ON HOSP.PATIENT

FOR EACH ROW EXECUTE PROCEDURE admitlogfunc( );
```

To check if the trigger function works and if it inserts data regarding the patient into the patient_admitted table.

```
        Select * from hosp.patient;
        Select * from hosp.patient_admited;
```

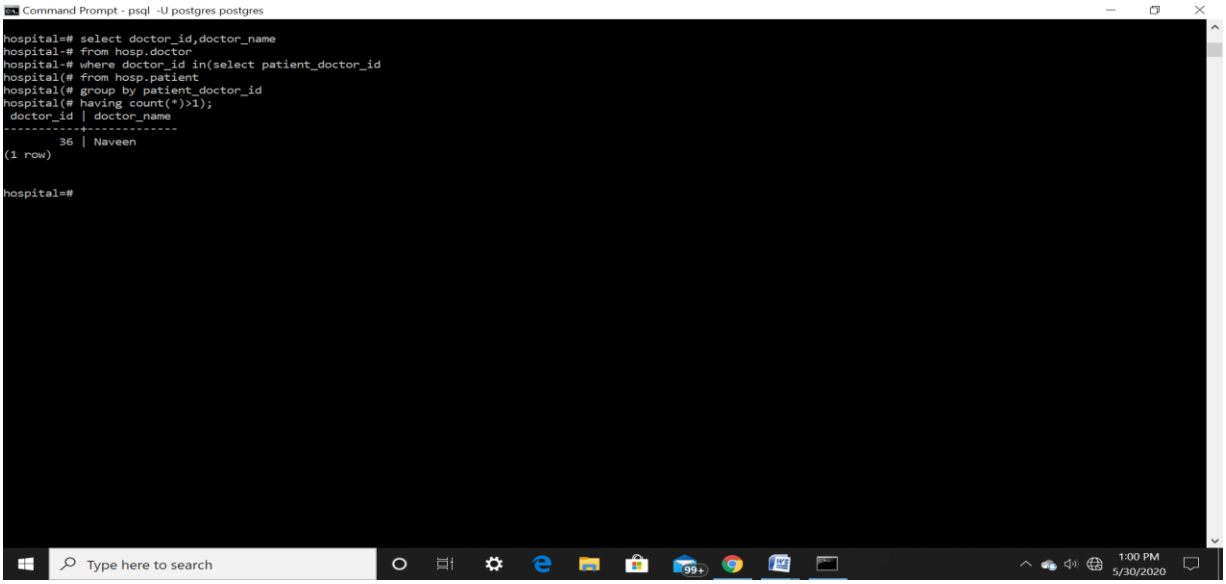


From this trigger we get to know the date and time of arrival of the patient in the hospital.

SQL Queries

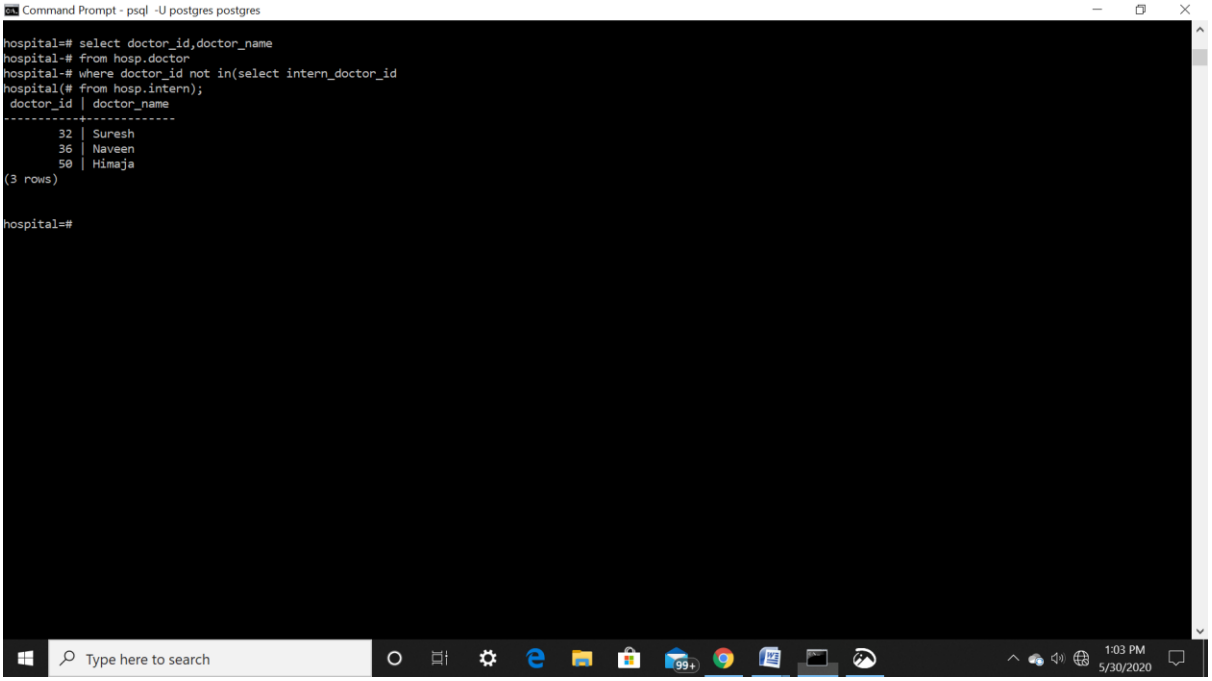
1.) Display all the doctors who have more than one patient?

```
Select doctor_id, doctor_name
from hosp.doctor
where doctor_id in(select patient_doctor_id
from hosp.patient
group by patient_doctor_id
having count(*)>1);
```



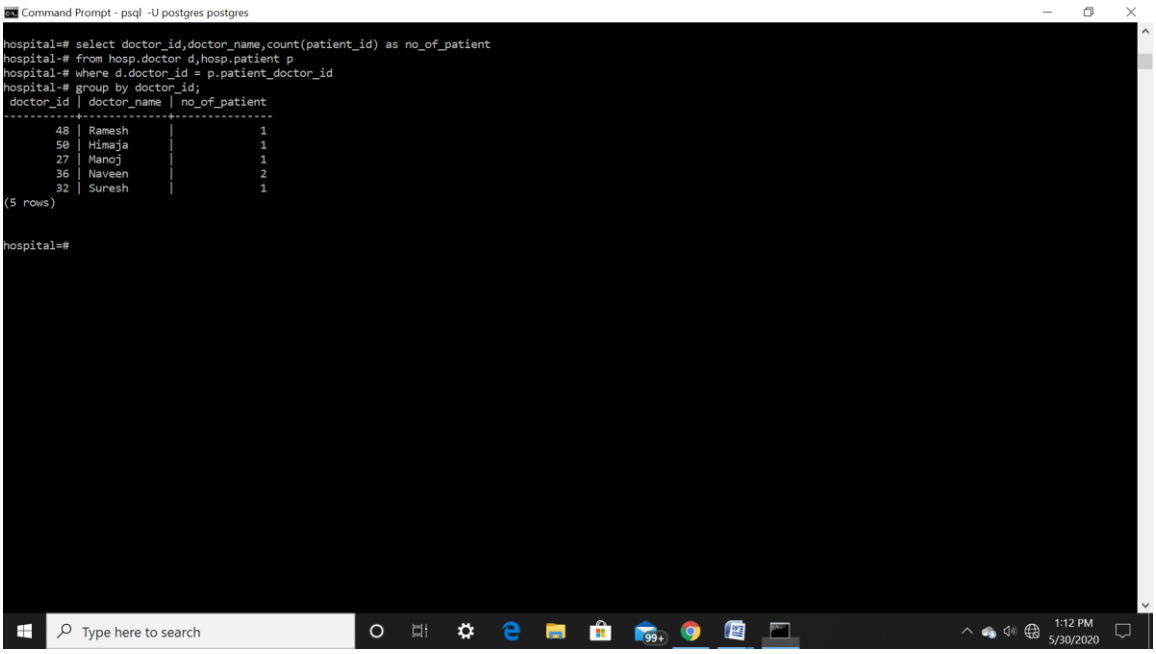
2.) Display the name of doctors who don't have interns under them?

```
Select doctor_id, doctor_name
from hosp.doctor
where doctor_id not in (select intern_doctor_id
from hosp.intern);
```



3.) Display the doctors with patients under their care?

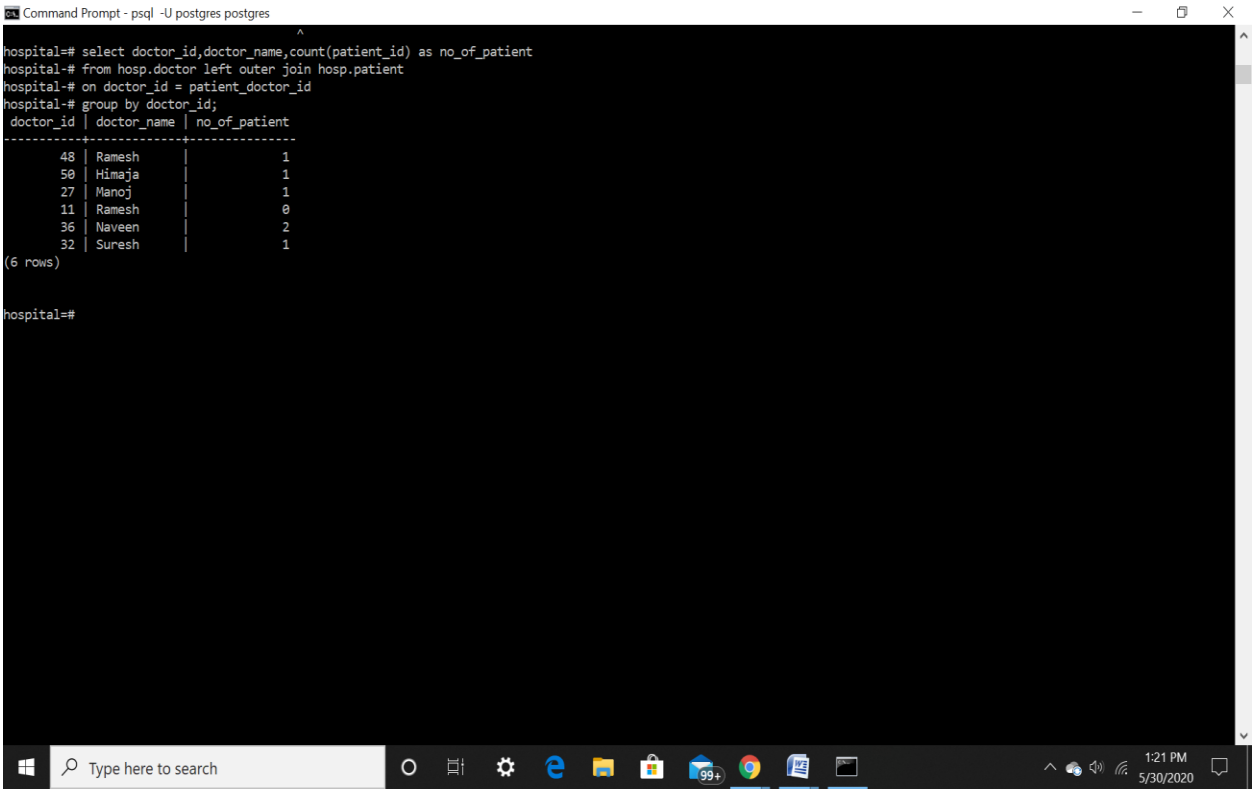
Select doctor_id, doctor_name, count(patient_id) as no_of_patient
from hosp.doctor d,hosp.patient p
where d.doctor_id = p.patient_doctor_id
group by doctor_id;



4.) Display the list of doctors with number of patients?(outer join)

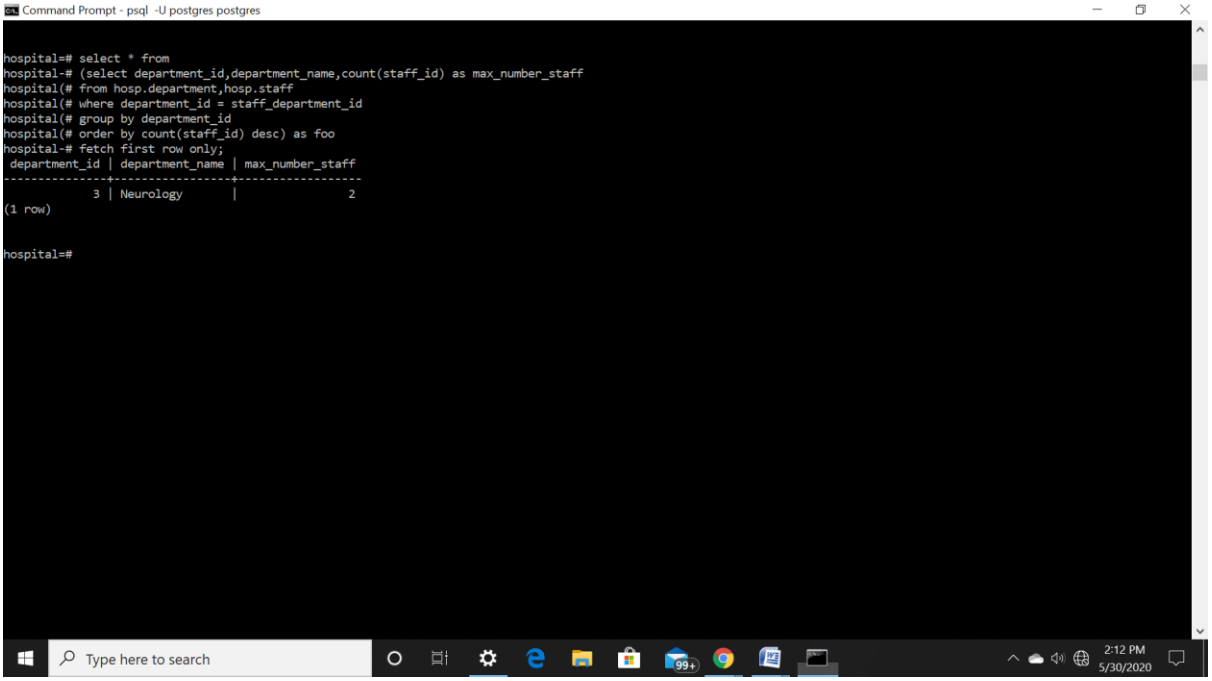
Select doctor_id,doctor_name,count(patient_id) as no_of_patient
from hosp.doctor left outer join hosp.patient
on doctor_id = patient_doctor_id

group by doctor_id;



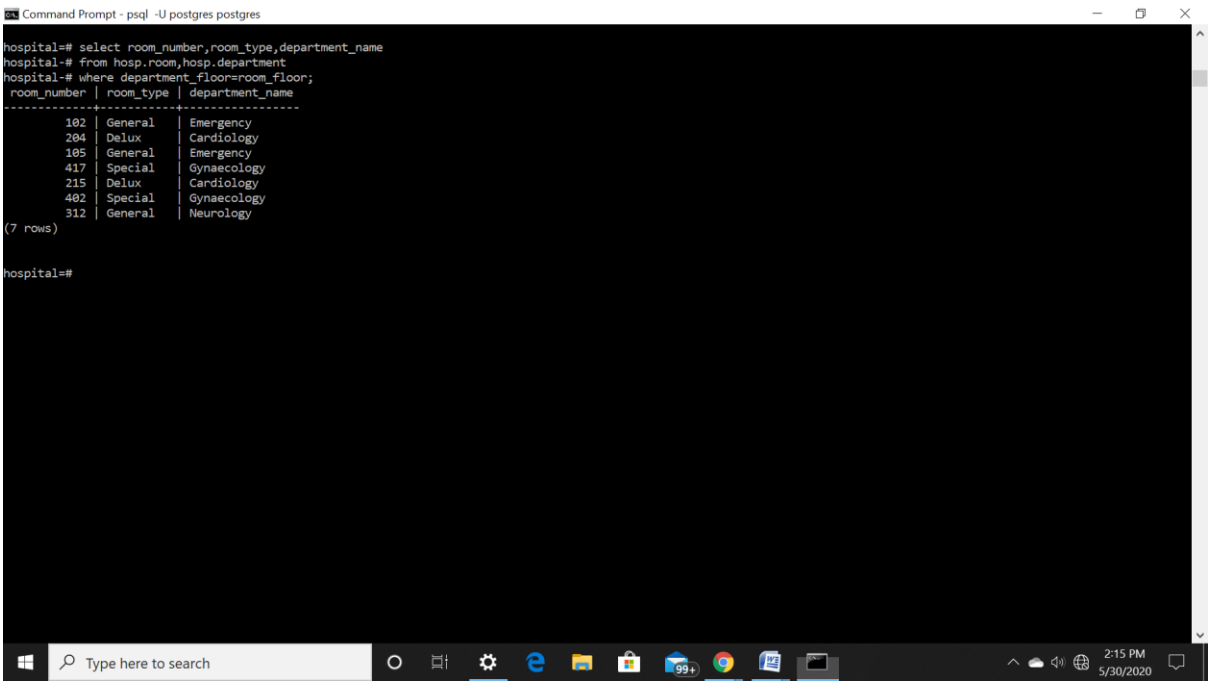
5.) Display the department with the maximum number of staff in it?

```
Select * from
(select department_id, department_name, count(staff_id) as max_number_staff
from hosp.department, hosp.staff
where department_id = staff_department_id
group by department_id
order by count(staff_id) desc) as foo
fetch first row only;
```



6.) Display the list of rooms with department name?

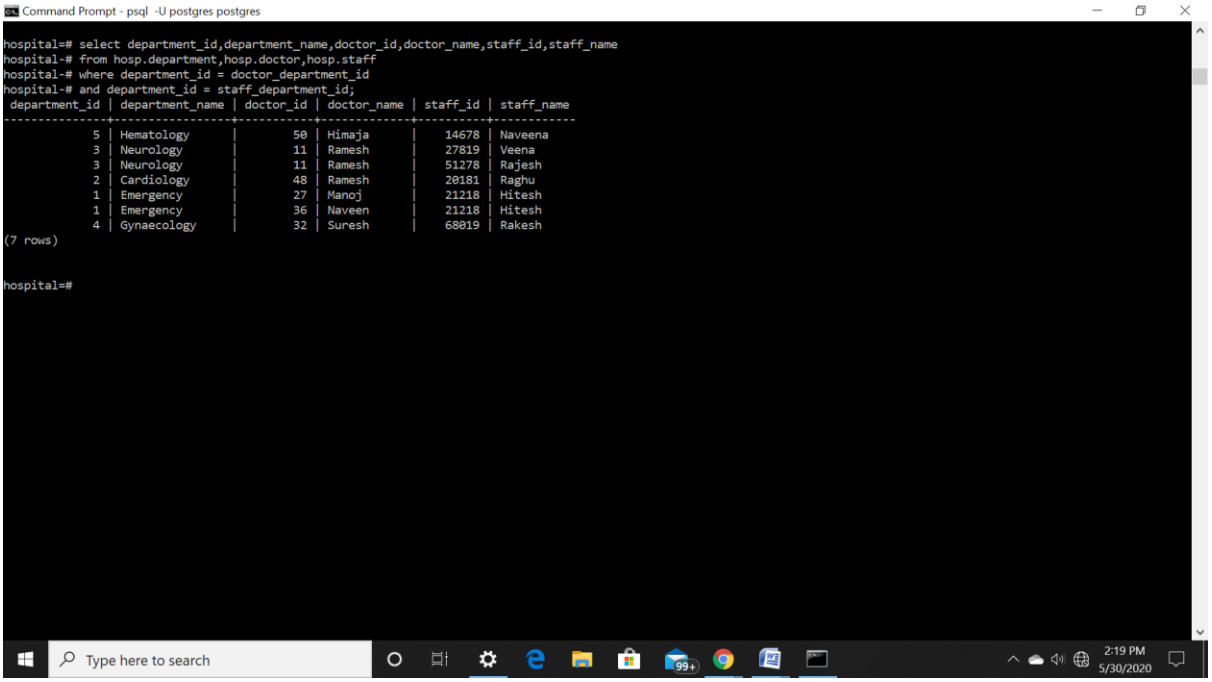
Select room_number, room_type,department_name
from hosp.room, hosp.department
where department_floor = room_floor;



7.) Display the list of all the doctors and staff working for the department?

Select department_id, department_name, doctor_id, doctor_name, staff_id, staff_name
from hosp.department, hosp.doctor, hosp.staff

where department_id = doctor_department_id
and department_id = staff_department_id;



Conclusion

Capabilities-

This system can record all important data regarding the hospital and is capable of getting a better understanding of the people who are related to the hospital. It can also be used to store details related to the patient like the admit time and room in which the patient has been admitted.

Limitations-

However, it only gives an overall view about the patient and the people working in the hospital but doesn't give any information regarding the patient's condition or the doctors working period .

It also only allows us to perform very basic comparisons.

Future work-

Attributes like patient's duration in the hospital, patient's condition and doctors working time can also be added to the database to make it better.

New tables which record data of every patient in details (like patient's admission date ,duration in the hospital and patient's condition) can be added which can be used to predict much more complex things like rebound duration a patient and compare a patient's condition(duration in the hospital) .

With better information about the hospital, the management would be easier.