# Hospital Management System

## 1. Design

**a) Explain what type of database architecture you recommend this type of database. For example, a centralized database, a distributed database, etc and why. (5 points)**

We recommended the use of traditional centralized relational database architecture for this project. Based on the project requirements, the use case is around a hospital management system, which appears to be for a single physical location and not with a very high volume. There will be a limited number of staff, limited number of doctors and even the number of patients they can see is small thus there is no clear requirement for a high transaction rate. This does not call for a more advanced architecture (like a Parallel or Distributed database architecture) as it will add additional complexity and performance overhead than necessary.

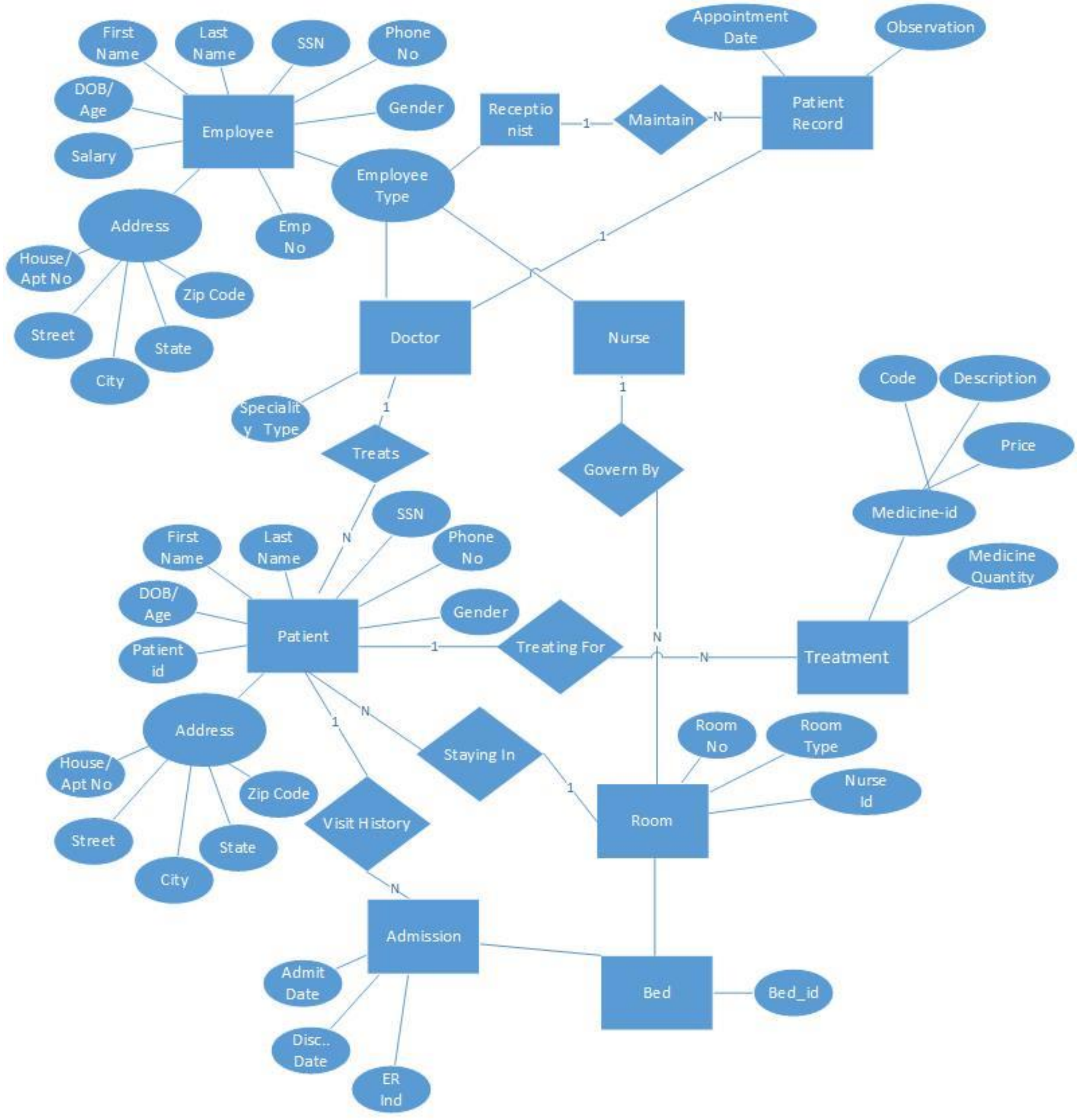
In addition, this database stores protected health information (PHI). Patient information comes under purview of Health Information Portability and Accountability Act (HIPAA). Therefore, data security is of prime importance. In a centralized relational database, it is relatively easy to secure the database using various mechanisms like Transparent Database encryption, Column level encryption to name a few. Since all the data is in one place, there can be stronger security measures around it. So, the centralized database can be much more secure (if controls are implemented correctly), instead of a distributed system where data is stored in multiple locations and more complex to manage and protect.

### b) Create a conceptual design for the hospital system database by using an ER diagram

As shown in the ER diagram below, here are some of the assumptions and reasoning behind the design

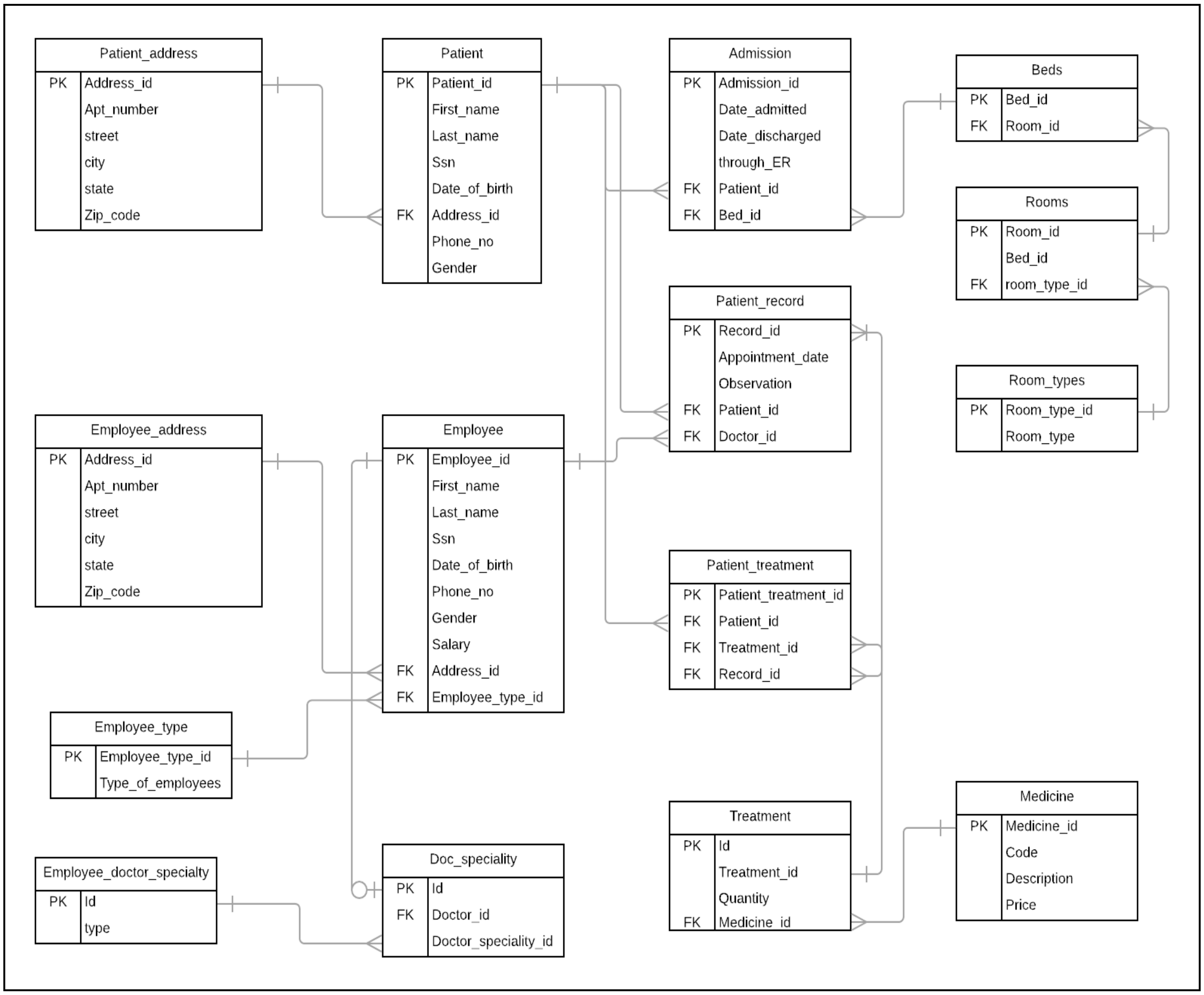
a) Employee_type – This table defines the types of employee - Receptionist, Nurse and Doctor.
b) Employee_doctor_speciality- This tables gives the details about the different specialties for doctors, 7 in our case.
c) Doc_Speciality - A doctor’s specialty is an attribute related to employee type Doctor, we store the specialty type for each doctor in this table.
d) Employee table – In this table, we maintain data associated with all employees of the hospital. Based on the given requirements, we are defining attributes for Name, date of birth, SSN, Phone number, Gender, Employee Type, Employee number, Address and Salary. Since address is a composite attribute, we created another table Employee_Address to store this data.
e) Patient – This table contains the patient information like name, ssn, date of birth, address, phone no and gender. For address, here also we created separate table.
f) Room_type - This table shows that there are two types of room- single room and double rooms are in the hospital.
g) Rooms - This table store the records of total rooms, type of the rooms, and nurse assigned to this room.
h) Beds – Since a single room will have 1 bed and double room will have 2 beds, we assigned each bed a unique id and associated it with the room. This way we can track down to a bed.
i) Medicine- This stores the information about the medicine i.e. its code, description and price.
j) Treatment - We assume that a treatment may consist of multiple medicines given to a patient. In order to accomplish this, we created this table to store treatment_id, medicine_id and its quantity.
k) Patient_record - This table contains the appointment date, observation, patient_id and doctor_id. Patient can visit multiple times and can be seen by different doctors. We also assumed, not all the patients seen by a doctor are admitted. Thus, separate table should be used to store admission related records.
l) Admission - If patient is admitted, here we track admission_id, whether admitted through ER, assigned bed, date when the patient was admitted and ultimately discharged.
m) Patient_treatment - We assume a patient might need multiple treatments and we track them here and map it to the patient record which reflect when the doctor prescribed this treatment.

# ER Diagram:


Appointment Date
Observation
First Name
Last Name
SSN
Phone No
DOB/ Age
Gender
Receptio nist
1
Maintain
N
Patient Record
Employee
Salary
Employee Type
Address
Emp No
1
House/ Apt No
Zip Code
Street
State
City
Doctor
Nurse
Specialit y Type
1
1
Treats
Govern By
Code
Description
Price
Medicine-id
Medicine Quantity
N
SSN
First Name
Last Name
Phone No
DOB/ Age
Gender
Patient
Patient id
1
Treating For
N
N
Treatment
Address
N
1
Staying In
Room No
Room Type
Nurse Id
House/ Apt No
Zip Code
1
Street
State
Visit History
Room
City
N
Admission
Bed
Bed_id
Admit Date
Disc.. Date
ER Ind

**c) Create a conceptual design for the hospital system database using UML class diagram.**

## UML Class Diagram:



**2. Transform the ER schema of database you get from step 1 into the corresponding relational database schema. (10 points)**

**a. Specify all the key attributes of relations and any referential integrity constraints.**

**b. Specify the data item format for each attribute in each relation schema.**

**c. Specify all the functional dependencies you could infer from the requirements.**

**Answer:** Please refer to the relational schema for the answers of Q. 2 a), b), and c).

## Relational Database Schema:

**Patient_address**

| Address_id | Apt_number | street | city | state | Zip_code |
|---|---|---|---|---|---|
| PK | | | | | |

**Patient**

| Patient | Patient_id | First_name | Last_name | Ssn | Date_of_birth | Address_id | Phone_no | Gender |
|---|---|---|---|---|---|---|---|---|
| PK | | | | | | FK | | |

**Emploee**

| Employee_id | First_name | Last_name | Ssn | Date_of_birth | Phone_no | Gender | Salary | Address_id | Employee_type_id |
|---|---|---|---|---|---|---|---|---|---|
| PK | | | | | | | | FK | FK |

**Employee_address**

| Address_id | Apt_number | street | city | state | Zip_code |
|---|---|---|---|---|---|
| PK | | | | | |

**Employee_type**

| Employee_type_id | Type_of_employees |
|---|---|
| PK | |

**Employee_doctor_specialty**

| Id | Type |
|---|---|
| PK | |

**Doc_speciality**

| Id | Doctor_id | Doctor_speciality_id |
|---|---|---|
| PK | | |

**Admission**

| Admission_id | Date_admitted | Date_discharged | through_ER | Patient_id | Bed_id |
|---|---|---|---|---|---|
| PK | | | | FK | FK |

**Patient_record**

| Record_id | Appointment_date | Observation | Patient_id | Doctor_id |
|---|---|---|---|---|
| PK | | | FK | FK |

**Patient_treatment**

| Patient_treatment_id | Patient_id | Treatment_id | Record_id |
|---|---|---|---|
| PK | FK | FK | FK |

**Beds**

| Bed_id | Room_id |
|---|---|
| PK | FK |

**Rooms**

| Room_id | Bed_id | Room_type_id |
|---|---|---|
| PK | | FK |

**Room_types**

| Room_type_id | Room_type |
|---|---|
| PK | |

**Treatment**

| Id | Treatment_id | Quantity | Medicine_id |
|---|---|---|---|
| PK | | | FK |

**Medicine**

| Medicine_id | Code | Description | Price |
|---|---|---|---|
| PK | | | |

**3. Normalize relation schema in the database design that you get from step 4 into either 3NF or BCNF if it is necessary. (10 points)**

Answer: Please refer to the relational schema for normalization in the database design.

## 4. Implement the relational database:

### Database for Hospital Management System:

```
CREATE DATABASE HospitalManagementSystem;
```

### 1. employee table:

```
CREATE TABLE public.employee (
    employee_id integer NOT NULL,
    first_name character varying (30) NOT NULL,
    last_name character varying (30) NOT NULL,
    ssn character varying (15) NOT NULL,
    date_of_birth character varying (15) NOT NULL,
    phone_no character varying (20) NOT NULL,
    gender character varying (10) NOT NULL,
    salary integer NOT NULL,
    address_id integer,
    employee_type_id integer,
    CONSTRAINT employee_pkey PRIMARY KEY (employee_id),
    CONSTRAINT employee_address_id_fkey FOREIGN KEY (address_id)
    REFERENCES public.employee_address (address_id) ON DELETE SET NULL,
    CONSTRAINT employee_employee_type_id_fkey FOREIGN KEY (employee_type_id)
    REFERENCES public.employee_type (id) ON DELETE SET NULL);
```

### 2. employee_address table:

```
CREATE TABLE public.employee_address(
    address_id integer NOT NULL,
    apt_number character varying (10) NOT NULL,
    street character varying (50) NOT NULL,
    city character varying (20) NOT NULL,
    state character varying (20) NOT NULL,
    zipcode character varying (5) NOT NULL,
    CONSTRAINT address_employee_pkey PRIMARY KEY (address_id));
```

### 3. employee_type table:

```
CREATE TABLE public.employee_type(
    id integer NOT NULL,
    types_of_employees character varying (20) NOT NULL,
    CONSTRAINT employee_type_pkey PRIMARY KEY (id));
```

### 4. employee_doctor_specialty table:

```
CREATE TABLE public.employee_doctor_speciality(
    id integer NOT NULL,
    type character varying (20) NOT NULL,
    CONSTRAINT employee_doctor_speciality_pkey PRIMARY KEY (id));
```

### 5. doc_speciality table:

```
CREATE TABLE public.doc_speciality(
    id integer NOT NULL DEFAULT nextval('doc_speciality_id_seq'::regclass),
    doctor_id integer NOT NULL,
    doctor_speciality_id integer NOT NULL,
    CONSTRAINT doc_speciality_pkey PRIMARY KEY (id),
    CONSTRAINT doc_speciality_doctor_id_fkey FOREIGN KEY (doctor_id)
    REFERENCES public.employee (employee_id) ON DELETE SET NULL,
    CONSTRAINT doc_speciality_doctor_speciality_id_fkey FOREIGN KEY (doctor_speciality_id)
    REFERENCES public.employee_doctor_speciality (id) ON DELETE SET NULL);
```

### 6. patient_record table:

```
CREATE TABLE public.patient_record (
    record_no integer NOT NULL,
    appointment_date date NOT NULL,
    observation character varying (200) NOT NULL,
    patient_id integer NOT NULL,
    doctor_id integer NOT NULL,
    CONSTRAINT patient_record_pkey PRIMARY KEY (record_no),
    CONSTRAINT patient_record_doctor_id_fkey FOREIGN KEY (doctor_id)
    REFERENCES public.employee (employee_id) ON DELETE SET NULL,
    CONSTRAINT patient_record_patient_id_fkey FOREIGN KEY (patient_id)
    REFERENCES public.patient (patient_id) ON DELETE SET NULL);
```

### 7. patient table:

```
CREATE TABLE public.patient(
    patient_id integer NOT NULL,
    first_name character varying (30) NOT NULL,
    last_name character varying (30) NOT NULL,
    ssn character varying (15) NOT NULL,
    date_of_birth date NOT NULL,
    address_id integer,
    phone_no character varying (15) NOT NULL,
    gender character varying (10) NOT NULL,
    CONSTRAINT patient_pkey PRIMARY KEY (patient_id),
    CONSTRAINT patient_address_id_fkey FOREIGN KEY (address_id)
    REFERENCES public.patient_address (address_id) ON DELETE SET NULL);
```

### 8. patient_address table:

```
CREATE TABLE public.patient_address (
    address_id integer NOT NULL,
    apt_number character varying (10) NOT NULL,
    street character varying (50) NOT NULL,
    city character varying (20) COLLATE NOT NULL,
    state character varying (20) COLLATE NOT NULL,
    zipcode character varying (5) COLLATE NOT NULL,
    CONSTRAINT address_pkey PRIMARY KEY (address_id));
```

### 9. beds tables:

```
CREATE TABLE public.beds(
    bed_id integer NOT NULL DEFAULT nextval('beds_bed_id_seq'::regclass),
    room_id integer NOT NULL,
    CONSTRAINT beds_pkey PRIMARY KEY (bed_id),
    CONSTRAINT beds_room_id_fkey FOREIGN KEY (room_id)
    REFERENCES public.rooms (room_id) ON DELETE SET NULL);
```

### 10. rooms table:

```
CREATE TABLE public.rooms (
    room_id integer NOT NULL DEFAULT nextval('rooms_room_id_seq'::regclass),
    room_types integer NOT NULL,
    nurse_id integer NOT NULL,
    CONSTRAINT rooms_pkey PRIMARY KEY (room_id),
    CONSTRAINT rooms_room_types_fkey FOREIGN KEY (room_types)
    REFERENCES public.room_types (room_type_id) ON DELETE SET NULL);
```

11. **room_types table:**

```
CREATE TABLE public.room_types (
    room_type_id integer NOT NULL,
    room_type character varying (20) NOT NULL,
    CONSTRAINT room_pkey PRIMARY KEY (room_type_id));
```

12. **medicine table:**

```
CREATE TABLE public.medicine(
    medicine_id integer NOT NULL,
    code character varying (20) NOT NULL,
    description character varying (200) NOT NULL,
    price integer NOT NULL,
    CONSTRAINT medicine_pkey PRIMARY KEY (medicine_id));
```

13. **treatment table:**

```
CREATE TABLE public.treatment (
    Id integer NOT NULL,
    treatment_id integer NOT NULL,
    quantity integer NOT NULL,
    medicine_id integer NOT NULL,
    CONSTRAINT id_pkey PRIMARY KEY (id),
    CONSTRAINT treatment_medicine_id_fkey FOREIGN KEY (medicine_id)
    REFERENCES public.medicine (medicine_id) ON DELETE SET NULL);
```

14. **patient_treatment table:**

```
CREATE TABLE public.patient_treatment (
    patient_treatment_id integer NOT NULL DEFAULT nextval('patient_treatment_patient_treatment_id_seq'::regclass),
    patient_id integer NOT NULL,
    treatment_id integer NOT NULL,
    record_id integer NOT NULL,
    CONSTRAINT patient_treatment_pkey PRIMARY KEY (patient_treatment_id),
    CONSTRAINT patient_treatment_patient_id_fkey FOREIGN KEY (patient_id)
    REFERENCES public.patient (patient_id) ON DELETE SET NULL,
    CONSTRAINT patient_treatment_record_id_fkey FOREIGN KEY (record_id)
    REFERENCES public.patient_record (record_no) ON DELETE SET NULL);
```

15. **admission table:**

```
CREATE TABLE public.admission (
    admission_id integer NOT NULL DEFAULT nextval('admission_admission_id_seq'::regclass),
    date_admitted date NOT NULL,
    date_discharged date NOT NULL,
    through_er boolean NOT NULL,
    patient_id integer NOT NULL,
    bed_id integer NOT NULL,
    CONSTRAINT admission_pkey PRIMARY KEY (admission_id),
    CONSTRAINT admission_bed_id_fkey FOREIGN KEY (bed_id)
    REFERENCES public.beds (bed_id) ON DELETE SET NULL,
    CONSTRAINT admission_patient_id_fkey FOREIGN KEY (patient_id)
    REFERENCES public.patient (patient_id) ON DELETE SET NULL);
```

## Data preparation:

## Queries to insert data into employee_doctor_speciality table:

```
insert into employee_doctor_speciality values(3,'pulmonologist');
insert into employee_doctor_speciality values(4,'nephrologist');
insert into employee_doctor_speciality values(5,'ENT');
insert into employee_doctor_speciality values(6,'neurologist');
insert into employee_doctor_speciality values(7,'endocrinologist');
```

## Queries to insert data into employee_type table:

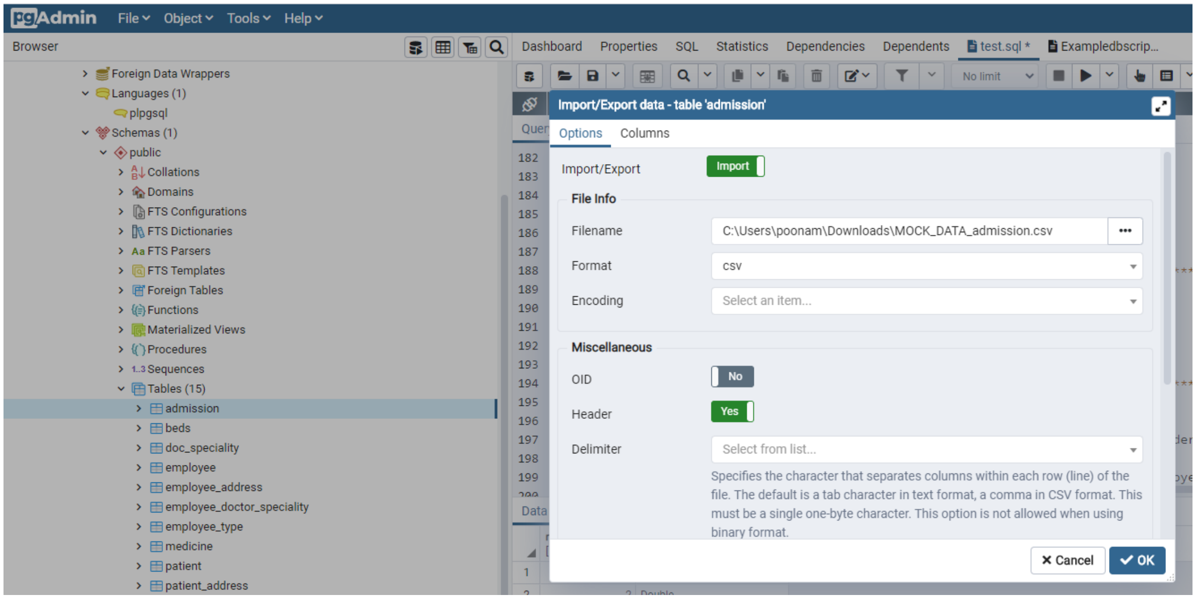
```
insert into employee_type values (1, 'doctor');
insert into employee_type values (2, 'nurse');
insert into employee_type values (3, 'receptionist');
```

## Queries to insert data into room_type table:

```
insert into employee_type values (1, 'Single');
insert into employee_type values (2, 'Double');
```

## To insert rest of the data into specific tables:

1. we used mockaroo data generator (https://www.mockaroo.com/) site to generate dummy data.
2. Save that data into csv files.
3. Then, Import the csv files into the database.



Import/Export data - table 'admission'

Import/Export: Import

File Info

Filename: C:\Users\poonam\Downloads\MOCK_DATA_admission.csv

Format: csv

Encoding: Select an item...

Miscellaneous

OID: No

Header: Yes

Delimiter: Select from list...

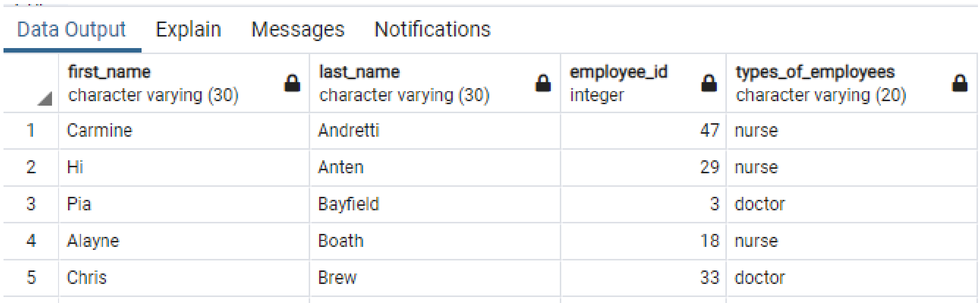
Specifies the character that separates columns within each row (line) of the file. The default is a tab character in text format, a comma in CSV format. This must be a single one-byte character. This option is not allowed when using binary format.

After importing all the files, we were able to run the queries.

## 5. Queries:

**1. List the last name, name, employee number, type of employee of all employees ordered by last name.**

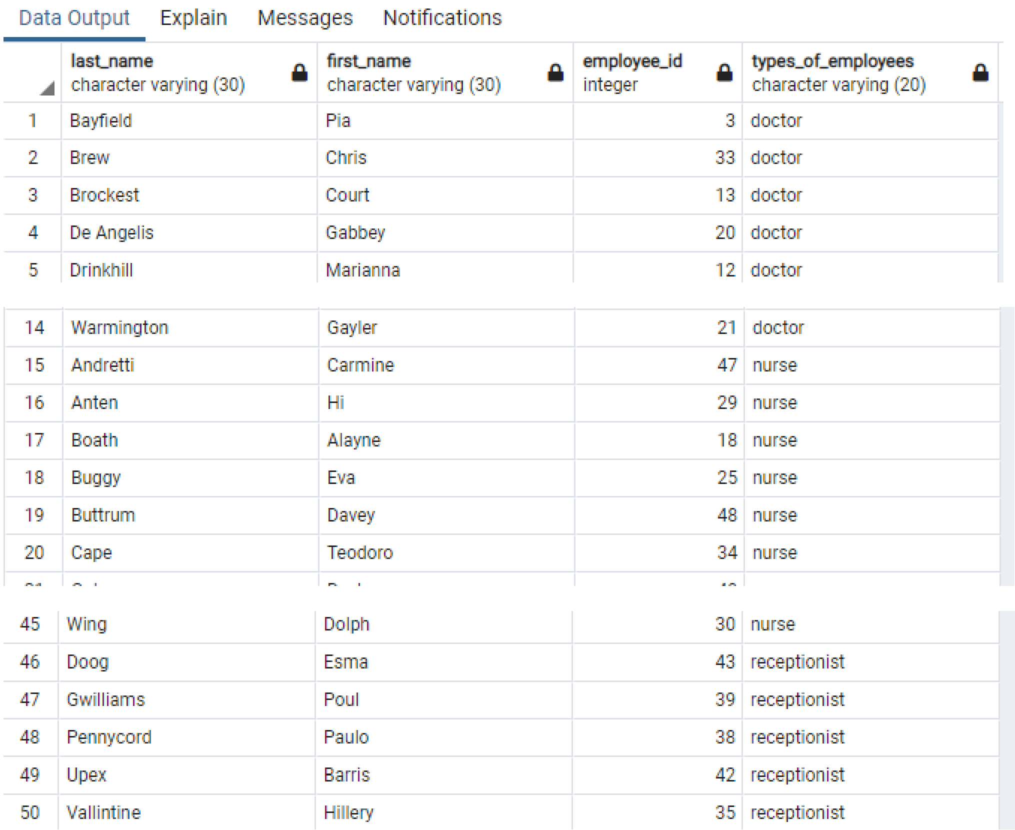
select first_name,last_name,employee_id,types_of_employees from employee e
join employee_type t on e.employee_type_id = t.id order by last_name;



| | first_name<br>character varying (30) | last_name<br>character varying (30) | employee_id<br>integer | types_of_employees<br>character varying (20) |
|---|---|---|---|---|
| 1 | Carmine | Andretti | 47 | nurse |
| 2 | Hi | Anten | 29 | nurse |
| 3 | Pia | Bayfield | 3 | doctor |
| 4 | Alayne | Boath | 18 | nurse |
| 5 | Chris | Brew | 33 | doctor |

**2. List the last name, name, employee number, type of employee of all employees ordered by last name grouped by employee type.**

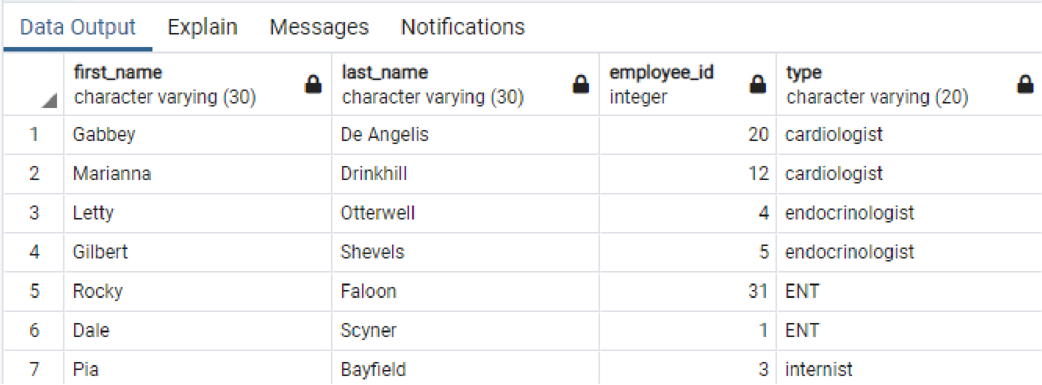
select employee.last_name, first_name, employee_id, employee_type.types_of_employees from employee
inner join public.employee_type  on employee.employee_type_id = employee_type.id
order by employee_type_id, last_name;



| | last_name<br>character varying (30) | first_name<br>character varying (30) | employee_id<br>integer | types_of_employees<br>character varying (20) |
|---|---|---|---|---|
| 1 | Bayfield | Pia | 3 | doctor |
| 2 | Brew | Chris | 33 | doctor |
| 3 | Brockest | Court | 13 | doctor |
| 4 | De Angelis | Gabbey | 20 | doctor |
| 5 | Drinkhill | Marianna | 12 | doctor |
| 14 | Warmington | Gayler | 21 | doctor |
| 15 | Andretti | Carmine | 47 | nurse |
| 16 | Anten | Hi | 29 | nurse |
| 17 | Boath | Alayne | 18 | nurse |
| 18 | Buggy | Eva | 25 | nurse |
| 19 | Buttrum | Davey | 48 | nurse |
| 20 | Cape | Teodoro | 34 | nurse |
| 45 | Wing | Dolph | 30 | nurse |
| 46 | Doog | Esma | 43 | receptionist |
| 47 | Gwilliams | Poul | 39 | receptionist |
| 48 | Pennycord | Paulo | 38 | receptionist |
| 49 | Upex | Barris | 42 | receptionist |
| 50 | Vallintine | Hillery | 35 | receptionist |

**3. List the name, last name, employee number, Specialty of doctors. Group by specialty and order by last name.**

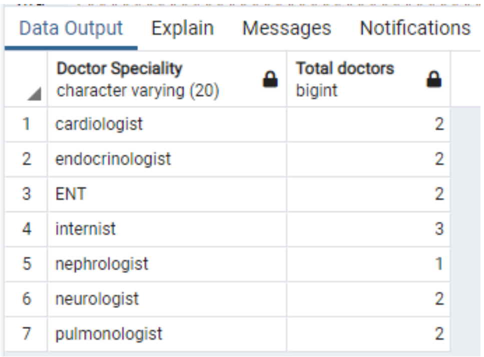
```
select emp.first_name,emp.last_name,emp.employee_id,eds.type from doc_speciality ds
inner join employee_doctor_speciality eds on eds.id = ds.doctor_speciality_id
inner join public.employee emp on ds.doctor_id = emp.employee_id
order by eds.type, emp.last_name;
```



| | first_name<br>character varying (30) | last_name<br>character varying (30) | employee_id<br>integer | type<br>character varying (20) |
|---|---|---|---|---|
| 1 | Gabbey | De Angelis | 20 | cardiologist |
| 2 | Marianna | Drinkhill | 12 | cardiologist |
| 3 | Letty | Otterwell | 4 | endocrinologist |
| 4 | Gilbert | Shevels | 5 | endocrinologist |
| 5 | Rocky | Faloon | 31 | ENT |
| 6 | Dale | Scyner | 1 | ENT |
| 7 | Pia | Bayfield | 3 | internist |

### 4. List the count of doctors per specialty order the list by specialty name.

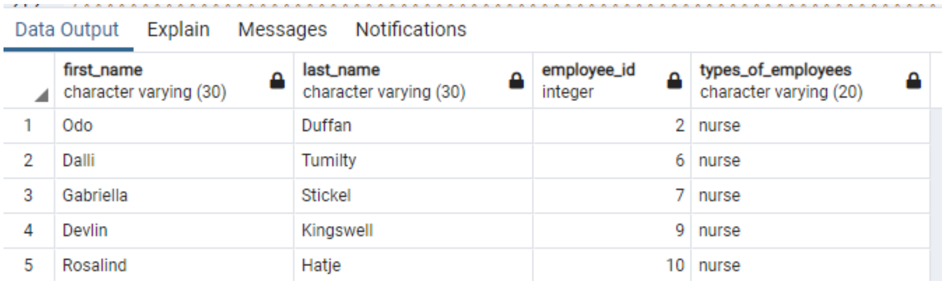
```
select eds.type as "Doctor Speciality", count(eds.type) as "Total doctors" from doc_speciality ds
join employee_doctor_speciality eds on eds.id=ds.doctor_speciality_id
join public.employee emp on ds.doctor_id=emp.employee_id
group by eds.type
order by eds.type;
```



| | Doctor Speciality<br>character varying (20) | Total doctors<br>bigint |
|---|---|---|
| 1 | cardiologist | 2 |
| 2 | endocrinologist | 2 |
| 3 | ENT | 2 |
| 4 | internist | 3 |
| 5 | nephrologist | 1 |
| 6 | neurologist | 2 |
| 7 | pulmonologist | 2 |

### 5. List the name, last name, employee number of all the nurses.

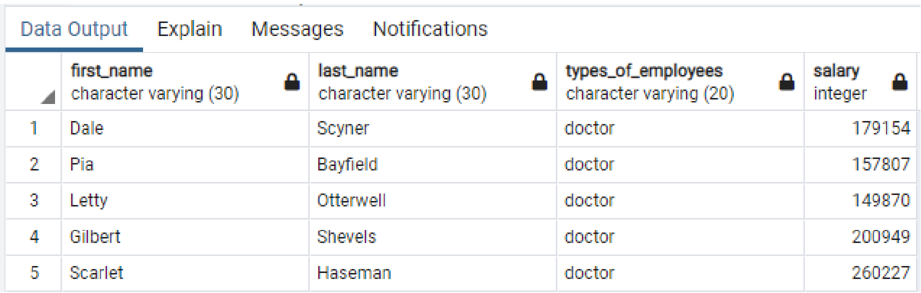
```
select first_name,last_name,employee_id,types_of_employees from employee
join employee_type  on employee.employee_type_id = employee_type.id
where types_of_employees = 'nurse';
```



| | first_name<br>character varying (30) | last_name<br>character varying (30) | employee_id<br>integer | types_of_employees<br>character varying (20) |
|---|---|---|---|---|
| 1 | Odo | Duffan | 2 | nurse |
| 2 | Dalli | Tumilty | 6 | nurse |
| 3 | Gabriella | Stickel | 7 | nurse |
| 4 | Devlin | Kingswell | 9 | nurse |
| 5 | Rosalind | Hatje | 10 | nurse |

### 6. List the employees name, last name, employee type, salary with salaries greater than 85K.

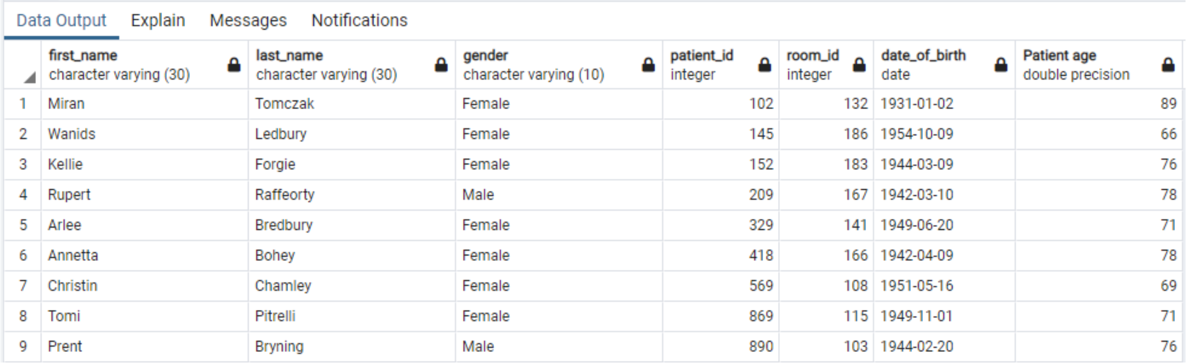
```
select first_name,last_name,types_of_employees,salary from employee
join employee_type  on employee.employee_type_id = employee_type.id
where employee.salary>85000;
```



|  | first_name<br>character varying (30) | last_name<br>character varying (30) | types_of_employees<br>character varying (20) | salary<br>integer |
|---|---|---|---|---|
| 1 | Dale | Scyner | doctor | 179154 |
| 2 | Pia | Bayfield | doctor | 157807 |
| 3 | Letty | Otterwell | doctor | 149870 |
| 4 | Gilbert | Shevels | doctor | 200949 |
| 5 | Scarlet | Haseman | doctor | 260227 |

### 7. List the name, last name, sex, patient id and room number of all the patients not discharged yet and who are older than 65 years old.

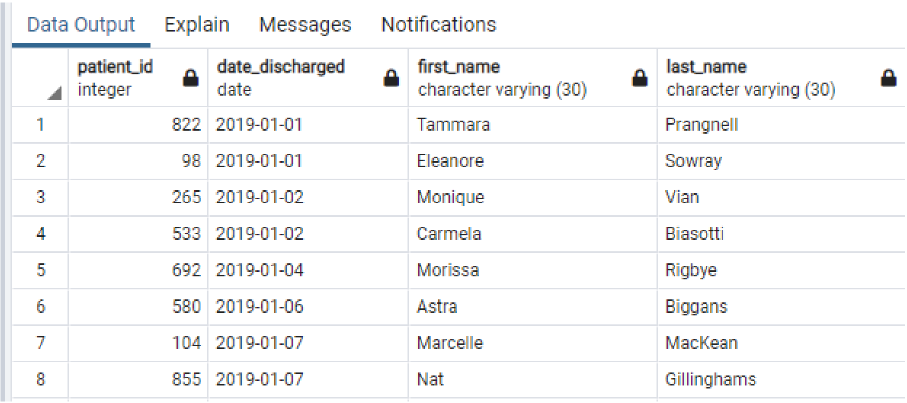
```
select patient.first_name, patient.last_name, patient.gender, patient.patient_id, beds.room_id, patient.date_of_birth,
date_part('year',CURRENT_TIMESTAMP) - date_part('year',patient.date_of_birth) as "Patient age"
from admission ad
inner join beds on beds.bed_id = ad.bed_id
inner join patient on ad.patient_id=patient.patient_id
where date_part('year',CURRENT_TIMESTAMP) - date_part('year',patient.date_of_birth) > 65 and ad.date_discharged is null;
```



|  | first_name<br>character varying (30) | last_name<br>character varying (30) | gender<br>character varying (10) | patient_id<br>integer | room_id<br>integer | date_of_birth<br>date | Patient age<br>double precision |
|---|---|---|---|---|---|---|---|
| 1 | Miran | Tomczak | Female | 102 | 132 | 1931-01-02 | 89 |
| 2 | Wanids | Ledbury | Female | 145 | 186 | 1954-10-09 | 66 |
| 3 | Kellie | Forgie | Female | 152 | 183 | 1944-03-09 | 76 |
| 4 | Rupert | Raffeorty | Male | 209 | 167 | 1942-03-10 | 78 |
| 5 | Arlee | Bredbury | Female | 329 | 141 | 1949-06-20 | 71 |
| 6 | Annetta | Bohey | Female | 418 | 166 | 1942-04-09 | 78 |
| 7 | Christin | Chamley | Female | 569 | 108 | 1951-05-16 | 69 |
| 8 | Tomi | Pitrelli | Female | 869 | 115 | 1949-11-01 | 71 |
| 9 | Prent | Bryning | Male | 890 | 103 | 1944-02-20 | 76 |

### 8. List the patient name, last name, patient id of patients discharged in one specific month (specified the month based on the data you used to populate the database).

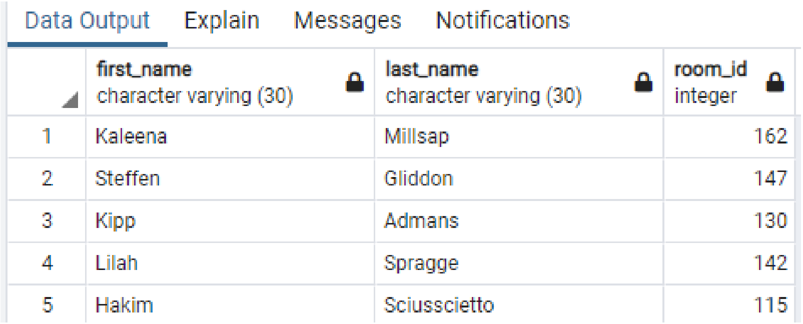
```
select ad.patient_id, ad.date_discharged,patient.first_name,patient.last_name from admission ad
inner join patient on ad.patient_id=patient.patient_id
where date_discharged is not null and date_part('month',date_discharged) =1
order by date_discharged;
```



| | patient_id<br>integer | date_discharged<br>date | first_name<br>character varying (30) | last_name<br>character varying (30) |
|---|---|---|---|---|
| 1 | 822 | 2019-01-01 | Tammara | Prangnell |
| 2 | 98 | 2019-01-01 | Eleanore | Sowray |
| 3 | 265 | 2019-01-02 | Monique | Vian |
| 4 | 533 | 2019-01-02 | Carmela | Biasotti |
| 5 | 692 | 2019-01-04 | Morissa | Rigbye |
| 6 | 580 | 2019-01-06 | Astra | Biggans |
| 7 | 104 | 2019-01-07 | Marcelle | MacKean |
| 8 | 855 | 2019-01-07 | Nat | Gillinghams |

**9. List the name and last name and room number of patients admitted through the ER and already discharged**

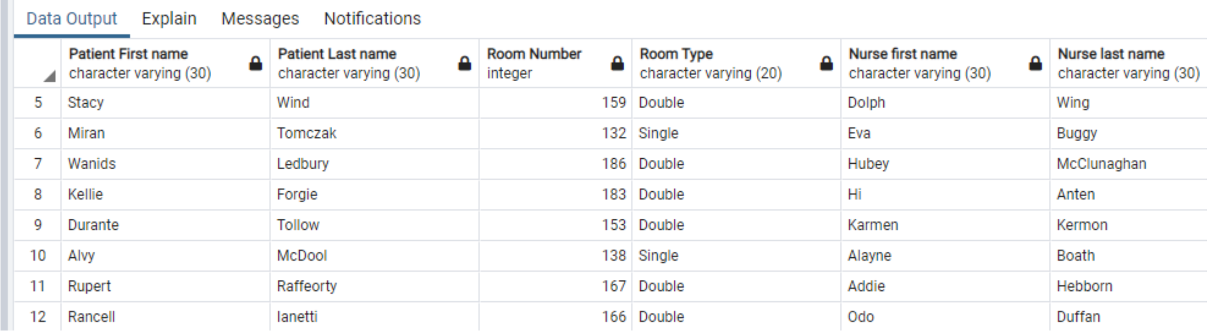
```
select patient.first_name, patient.last_name, beds.room_id from admission ad
inner join beds on beds.bed_id = ad.bed_id
inner join patient on ad.patient_id=patient.patient_id
where ad.date_discharged is not null and ad.through_er is true;
```



| | first_name<br>character varying (30) | last_name<br>character varying (30) | room_id<br>integer |
|---|---|---|---|
| 1 | Kaleena | Millsap | 162 |
| 2 | Steffen | Gliddon | 147 |
| 3 | Kipp | Admans | 130 |
| 4 | Lilah | Spragge | 142 |
| 5 | Hakim | Sciusscietto | 115 |

**10. List the name and last name, assigned room, room type and assigned nurse name and last name of patients not discharged yet.**

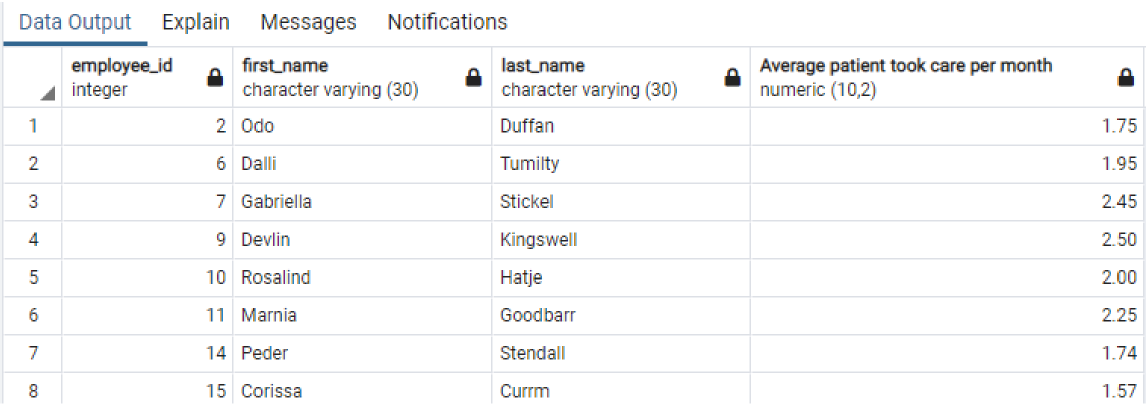
```
select patient.first_name as "Patient First name", patient.last_name as "Patient Last name", beds.room_id as "Room Number",
room_types.room_type as "Room Type", employee.first_name " Nurse first name", employee.last_name  as " Nurse last name"
from admission ad join beds on beds.bed_id = ad.bed_id join patient on ad.patient_id=patient.patient_id
join rooms on rooms.room_id=beds.room_id join room_types on rooms.room_types = room_types.room_type_id
join employee on employee.employee_id = rooms.nurse_id where ad.date_discharged is null;
```



| | Patient First name<br>character varying (30) | Patient Last name<br>character varying (30) | Room Number<br>integer | Room Type<br>character varying (20) | Nurse first name<br>character varying (30) | Nurse last name<br>character varying (30) |
|---|---|---|---|---|---|---|
| 5 | Stacy | Wind | 159 | Double | Dolph | Wing |
| 6 | Miran | Tomczak | 132 | Single | Eva | Buggy |
| 7 | Wanids | Ledbury | 186 | Double | Hubey | McClunaghan |
| 8 | Kellie | Forgie | 183 | Double | Hi | Anten |
| 9 | Durante | Tollow | 153 | Double | Karmen | Kermon |
| 10 | Alvy | McDool | 138 | Single | Alayne | Boath |
| 11 | Rupert | Raffeorty | 167 | Double | Addie | Hebborn |
| 12 | Rancell | Ianetti | 166 | Double | Odo | Duffan |

### 11. List the nurse name, nurse last name and average patient they took care per month.

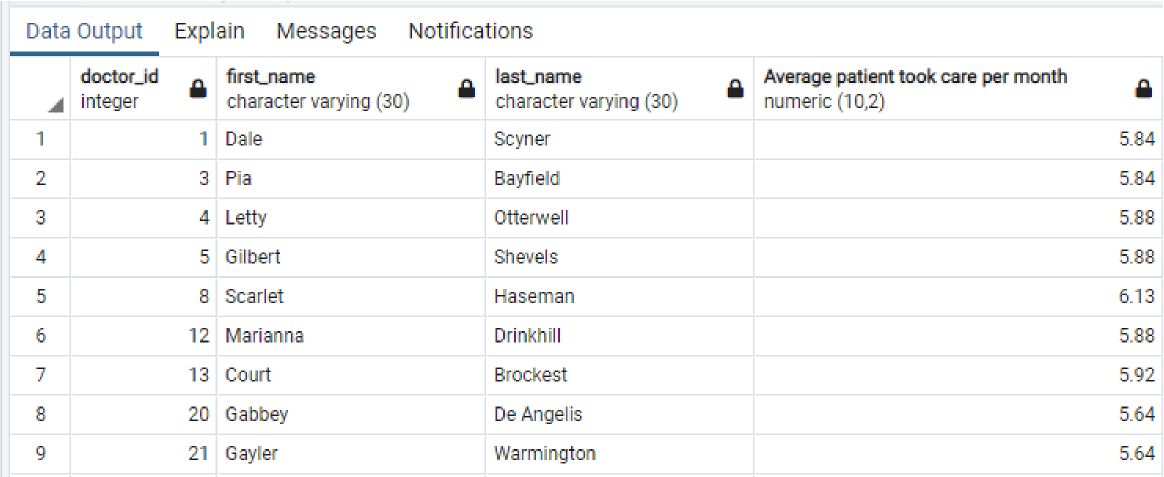
```
create temp table nurse_table as
select date_trunc('month',ad.date_admitted), count(employee.employee_id), employee.employee_id, employee.first_name,
employee.last_name from admission ad
inner join beds on beds.bed_id = ad.bed_id
inner join rooms on rooms.room_id=beds.room_id
inner join employee on employee.employee_id = rooms.nurse_id group by
date_trunc('month',ad.date_admitted),employee.employee_id
order by employee.employee_id;

select  employee_id, first_name,last_name,avg(count)::numeric(10,2) as "Average patient took care per month"
from nurse_table group by employee_id,first_name,last_name order by employee_id;
```



| | employee_id<br>integer | first_name<br>character varying (30) | last_name<br>character varying (30) | Average patient took care per month<br>numeric (10,2) |
|---|---|---|---|---|
| 1 | 2 | Odo | Duffan | 1.75 |
| 2 | 6 | Dalli | Tumilty | 1.95 |
| 3 | 7 | Gabriella | Stickel | 2.45 |
| 4 | 9 | Devlin | Kingswell | 2.50 |
| 5 | 10 | Rosalind | Hatje | 2.00 |
| 6 | 11 | Marnia | Goodbarr | 2.25 |
| 7 | 14 | Peder | Stendall | 1.74 |
| 8 | 15 | Corissa | Currm | 1.57 |

### 12. List the doctor name and last name and average patient attended per month for all the doctors.

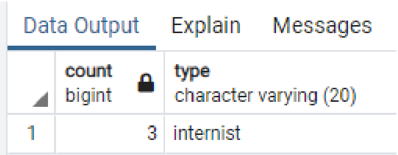
```
create temp table doc_by_month  as
select DATE_TRUNC('month',appointment_date) as appointmentsinamonth, count(doctor_id) as records,
doctor_id,e.first_name,e.last_name from patient_record join employee e on patient_record.doctor_id = e.employee_id
group by DATE_TRUNC('month',appointment_date), doctor_id,e.first_name,e.last_name order by doctor_id,
appointmentsinamonth;
select doctor_id,first_name,last_name,avg(records)::numeric(10,2) as "Average patient took care per month"
from doc_by_month group by doctor_id,first_name,last_name order by doctor_id;
```



| | doctor_id<br>integer | first_name<br>character varying (30) | last_name<br>character varying (30) | Average patient took care per month<br>numeric (10,2) |
|---|---|---|---|---|
| 1 | 1 | Dale | Scyner | 5.84 |
| 2 | 3 | Pia | Bayfield | 5.84 |
| 3 | 4 | Letty | Otterwell | 5.88 |
| 4 | 5 | Gilbert | Shevels | 5.88 |
| 5 | 8 | Scarlet | Haseman | 6.13 |
| 6 | 12 | Marianna | Drinkhill | 5.88 |
| 7 | 13 | Court | Brockest | 5.92 |
| 8 | 20 | Gabbey | De Angelis | 5.64 |
| 9 | 21 | Gayler | Warmington | 5.64 |

### 13. List the specialty and number of doctors for the specialty that has more than 3 doctors that make more than 100K a year.

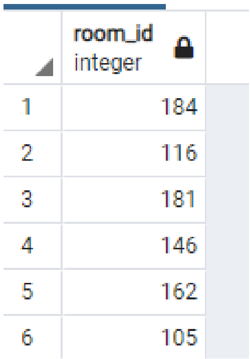
```
select Count (eds.type), eds.type from doc_speciality ds
join employee_doctor_speciality eds on eds.id=ds.doctor_speciality_id
join public.employee emp on ds.doctor_id=emp.employee_id
where emp.salary > 100000 group by eds.id having count(eds.type) > 2;
```



| | count<br>bigint | type<br>character varying (20) |
|---|---|---|
| 1 | 3 | internist |

### 14. List the room number of any empty room.

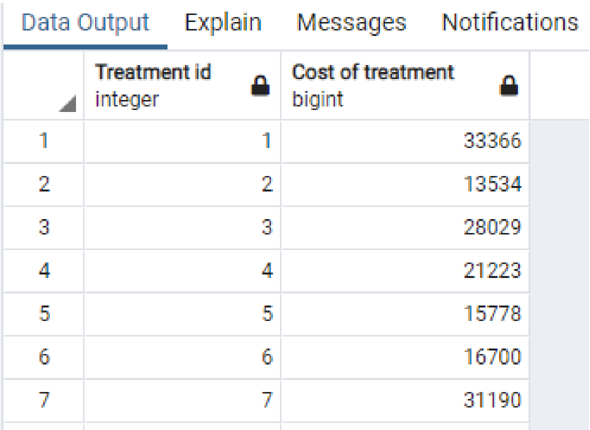
```
select x.room_id from beds x EXCEPT select occupied.room_id from beds  occupied
inner join admission ad on ad.bed_id = occupied.bed_id  where ad.date_discharged is null;
```



| | room_id<br>integer |
|---|---|
| 1 | 184 |
| 2 | 116 |
| 3 | 181 |
| 4 | 146 |
| 5 | 162 |
| 6 | 105 |

### 15. List the average cost of a treatment.

```
select treatment_id as " Treatment id", sum (treatment.quantity * medicine.price) as "Cost of treatment"
from treatment inner join medicine on treatment.medicine_id = medicine.medicine_id
group by treatment_id order by treatment_id asc;
```



| | Treatment id<br>integer | Cost of treatment<br>bigint |
|---|---|---|
| 1 | 1 | 33366 |
| 2 | 2 | 13534 |
| 3 | 3 | 28029 |
| 4 | 4 | 21223 |
| 5 | 5 | 15778 |
| 6 | 6 | 16700 |
| 7 | 7 | 31190 |