Database Project

Betty Tai

Table of Contents

Data and Functional Requirements.

This project is a database project of a blood donation database that seeks to classify the system used in a blood bank. The end result of this project will be open-source and available to use for the public.

The data that I use is fictionally created data, and I will have several different kinds of records. For the data, I will show the fictional database in the backend as well as show the patient views in the front end. I will be using the insert statement vis-a-vis SQL to create fictional data points.

The data is created in such a way that all corner cases such as a many-to-one relationships and foreign key constraints will be used to demonstrate novel views that could be part of a beta-testing if this application were a real live product.

In this project, there will be a creation of multiple tables with multiple attributes to store information about donor collection of blood as well as show who accessed the records and how much blood is in the blood banking system. Below, PK stands for Primary Key, and FK stands for Foreign Key.

For each donor email collected, an id will be generated. What will be collected is the first name of the donor, the last name of the donor, username for the donor and password for the donor. This is where the clientele will create their username and password. We are to have table with the information about the general information about donor:

- donor_email – PK
- donor_id
- donor_firstName
- donor_lastName
- donor_username
- donor_password

For each donor_id generated, each clientele needs to fill out the form stating their phone number, email, age, gender, bloodtype, and home address. This is a table about the details of the donor:

- donor_id – PK
- donor_number
- donor_email - FK
- donor_age
- donor_gender
- donor_bloodtype
- donor_address

For each email collected about the employee, the system also needs to collect information about the first name, last name, username and password. The id is automatically generated. This is a table about the medical personnel credentials accessing the information:

- empl_email - PK
- empl_id
- empl_firstName

- empl_lastName
- empl_username
- empl_password

For each blood id generated, there is a connected donor id and relevant information such as quantity donated, when the blood was donated and blood type will be collected from the medical personnel. This is a table about the blood donation and relevant information:

- blood_id - PK
- donor_id - FK
- quantityDonated
- blood_dateDonated
- blood_type

For each recipient id, first and last name will be collected as well as phone number, email, age, gender, bloodtype and address will be collected from the recipient. This is a table about the medical personnel information in which information gets stored:

- recipient_id - PK
- recipient_firstName
- recipient_lastName
- recipient_number
- recipient_email
- recipient_age
- recipient_gender
- recipient_bloodtype
- recipient_address

The medical personnel will collect the date out (when they decide to use the blood), quantity and recipient that they will be giving the blood to. This is a table about the blood transaction and relevant information:

- transaction_id - PK
- empl_id - FK
- blood_id
- dateOut
- quantity
- recipient_id - FK

This information is collected regarding pre-existing conditions about the donor and relates via foreign key via the donor's email. The seventh table is about the preexisting conditions some of the donors may have:

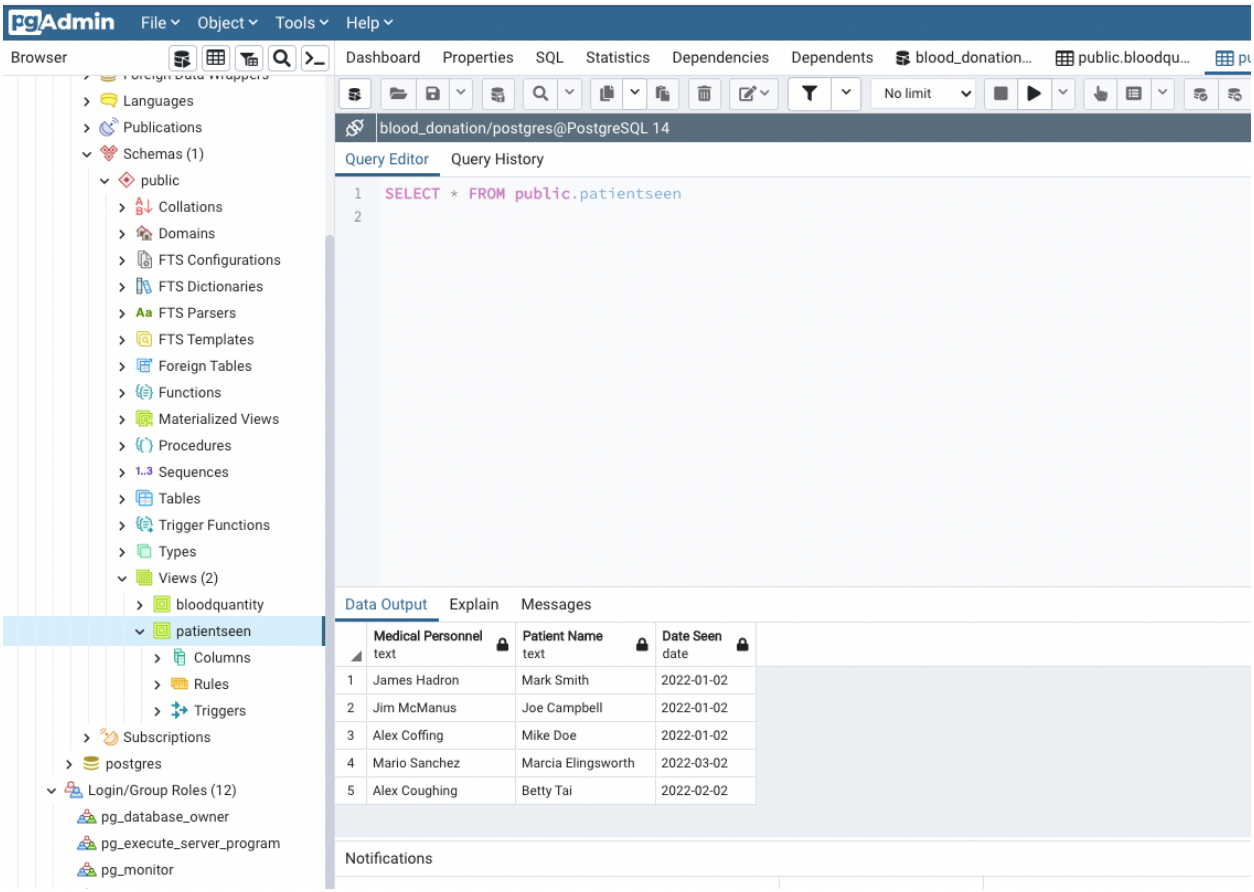
- donor_preexisting_conditions – PK
- donor_email - FK

The purpose of this table is to demonstrate if there are changes in the last name of the employee. This table is used to demonstrate that there is a trigger and stored procedure with regards to the The eighth table is about the medical personnel changing their last name and auditing them for the last name changes:

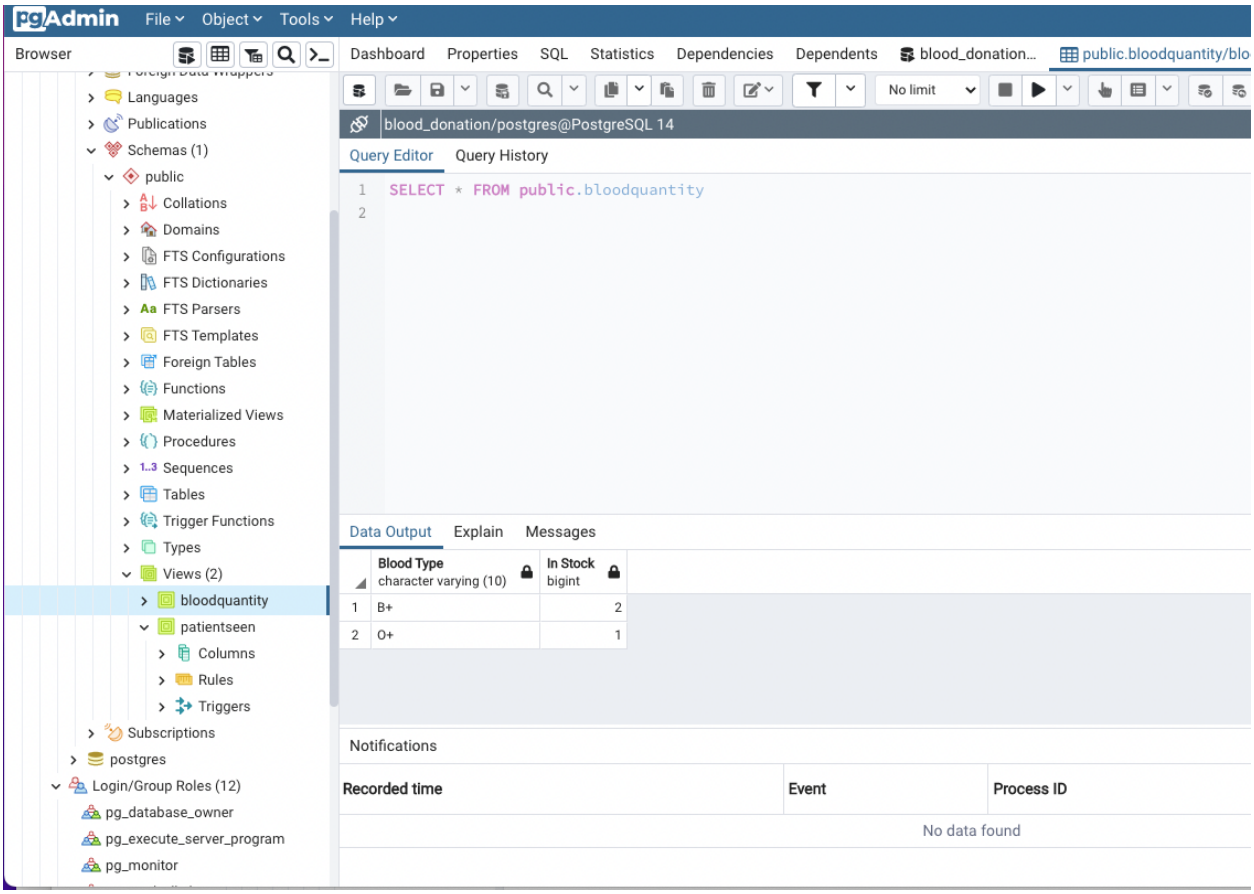
- empl_id FK
- empl_lastName

- changed_on

Here, the tables use foreign keys to link together in order to reinforce the relationships. There will be several kinds of different kinds of views: PatientSeen and BlockQuanitity. By segmenting the views to only two, we also add security of the views by not showing irrelevant data to unauthorized users. The views use joins to make columns from different tables.



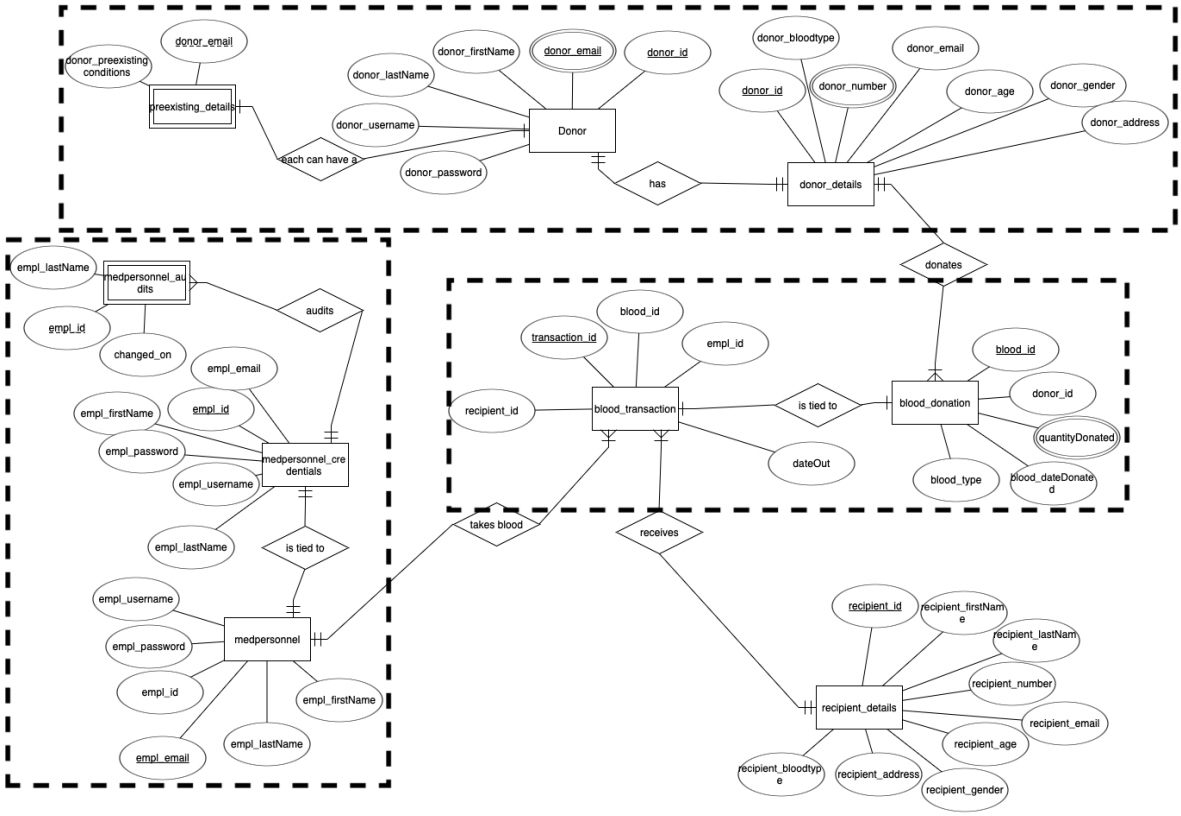
In the PatientSeen view, there is the concatenation of the patient seen and seen by whom as well as the date. This is useful for the database administrator because it shows who was seen by whom, and when in order to monitor the chain of custody regarding the blood donation.



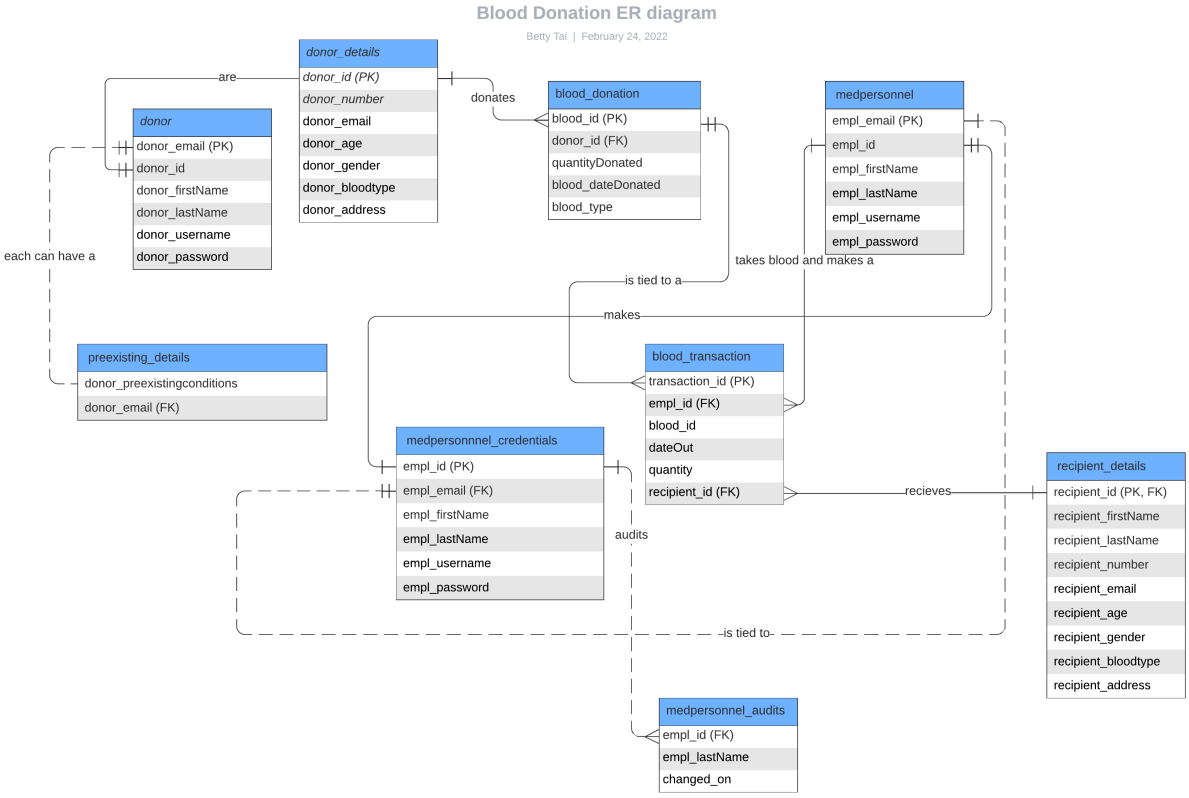
In the stock of blood view called BloodQuantity, the purpose of the view is to show how much of each blood and type of blood that is currently in the blood bank. This view aims to classify the kinds of blood joining the blood group table, and blood is categorized into the types of blood.

ER Model.

a) Entity and Relationship Sets
The entities and relationship sets are in the diagram.
b) Composite and Multivalued Attributes
A multivalued attribute is the donor's number (donor_number) and recipient's number (recipient_number). An example of a composite attribute is the address of the donor, which comprises of Street, City, State and Country as well as Zip Code.
c) Weak Entity Sets
An example of a weak entity set is the medpersonnel_audits because it is linked by the emails of the medical personnel employees. Another example of a weak entity set is preexisting_details because it is linked with the donor with the foreign key of the donor's email, and the entity set called preexisting_details does not have a primary key.
d) Cardinality Constraints
Generally, a donor can have multiple blood donations and blood transactions.
Medical personnel employees can participate in multiple blood transactions, especially when they are logged into their account via empl_username and empl_password.
A recipient can have several blood donations.
There can be many audits tied to one personnel if a personnel chooses to change their last name several times.
e) Participation Constraints
A partial participation constraint example would be not all blood_donations have a blood_transaction. There is more blood banked than there are transfused for recipients.
f) Specialization/Generalization
There is specialization and generalization in the ER diagram with the donor and the donor_details. The details of the donor are the specialization of the donor which is the general form.
g) Attributes on Relationship Sets
Donor_details is a good example of attributes; attributes for donor_details include the id, phone number, email, age, gender, blood type, and address for residence.
h) Aggregation
There is aggregation present in this blood bank database system. Aggregation occurs with medpersonnel and medpersonnel_credentials. Another example of aggregation occurs with donor and donor details.
i) N-ary Relationship Sets
There is a binary relationship for a donor to give blood via a blood donation. There are no unary relationship sets, and an n-ary relationship set example would be the blood_transaction. There can be multiple blood_donation per each blood_transaction (when the blood is given to the recipient) and a recipient can receive multiple quantities of blood_transactions in one transaction or multiple transactions.


donor_email
donor_preexisting conditions
preexisting_details
donor_firstName
donor_email
donor_id
donor_lastName
donor_username
donor_password
Donor
each can have a
has
donor_bloodtype
donor_email
donor_id
donor_number
donor_age
donor_gender
donor_address
donor_details
donates
empl_lastName
medpersonnel_audits
audits
empl_id
changed_on
empl_email
empl_firstName
empl_id
empl_password
empl_username
medpersonnel_credentials
empl_lastName
is tied to
empl_username
empl_password
empl_id
empl_firstName
empl_email
empl_lastName
medpersonnel
takes blood
blood_id
transaction_id
empl_id
recipient_id
blood_transaction
is tied to
dateOut
blood_id
donor_id
quantityDonated
blood_type
blood_dateDonated
blood_donation
receives
recipient_id
recipient_firstName
recipient_lastName
recipient_number
recipient_email
recipient_age
recipient_gender
recipient_address
recipient_bloodtype
recipient_details

Relational Database Schema Diagram.



SQL DDL.

```
create database blood_donation_db;


create table donor(
donor_email varchar(50) NOT NULL UNIQUE,
donor_id int NOT NULL,
donor_firstName varchar(50) NOT NULL,
donor_lastName varchar(50) NOT NULL,
donor_username varchar(50) NOT NULL,
donor_password varchar(50) NOT NULL,
primary key(donor_email)
);

/*create table donor_details in which all donor information gets stored.*/
create table donor_details(
donor_id int NOT NULL,
donor_number varchar(12) NOT NULL,
donor_email varchar(50),
donor_age int NOT NULL,
donor_gender varchar(10) NOT NULL,
donor_bloodtype varchar(10) NOT NULL,
donor_address varchar(100) NOT NULL,
Primary key(donor_id),
foreign key(donor_email) references donor(donor_email)
);

create table medpersonnel(
empl_email varchar(50) NOT NULL UNIQUE,
empl_id int NOT NULL,
empl_firstName varchar(50) NOT NULL,
empl_lastName varchar(50) NOT NULL,
empl_username varchar(50) NOT NULL,
empl_password varchar(50) NOT NULL,
primary key(empl_email)
);

/* create a table with all information about medical personnel gets stored*/
create table medpersonnel_credentials(
empl_id int NOT NULL UNIQUE,
empl_email varchar(50) NOT NULL,
empl_firstName varchar(50) NOT NULL,
empl_lastName varchar(50) NOT NULL,
empl_username varchar(50) NOT NULL,
empl_password varchar(50) NOT NULL,
primary key(empl_id),
foreign key(empl_email) references medpersonnel(empl_email)
```

```
);


/*create table blood in which all blood group is stored.*/
create table blood_donation(
blood_id SERIAL Not Null,
donor_id int Not Null,
quantityDonated INT NOT NULL,
blood_dateDonated DATE NOT NULL,
blood_type varchar(10) NOT NULL,
primary key(blood_id),
foreign key(donor_id) references donor_details(donor_id)
);

/*create table donor_details in which all donor information gets stored.*/
create table recipient_details(
recipient_id int NOT NULL,
recipient_firstName varchar(50) NOT NULL,
recipient_lastName varchar(50) NOT NULL,
recipient_number varchar(12) NOT NULL,
recipient_email varchar(50),
recipient_age int NOT NULL,
recipient_gender varchar(10) NOT NULL,
recipient_bloodtype varchar(10) NOT NULL,
recipient_address varchar(100) NOT NULL,
Primary key(recipient_id)
);

create table blood_transaction(
transaction_id int,
empl_id int NOT NULL,
blood_id SERIAL NOT NULL,
dateOut DATE NOT NULL,
quantity int NOT NULL,
recipient_id int NOT NULL,
primary key(transaction_id),
foreign key(recipient_id) references recipient_details(recipient_id),
foreign key(blood_id) references blood_donation(blood_id),
foreign key(empl_id) references medpersonnel_credentials(empl_id)
);

create table preexisting_details(
donor_preexistingconditions varchar(50) NOT NULL,
donor_email varchar(50) NOT NULL UNIQUE,
foreign key(donor_email) references donor(donor_email)
);

create table medpersonnel_audits(
```

```
  empl_id int NOT NULL,
  empl_lastName varchar(50) NOT NULL,
  changed_on TIMESTAMP(6) NOT NULL,
  foreign key(empl_id) references medpersonnel_credentials(empl_id)
)

Create view PatientSeen as Select
concat_ws(' ', m.empl_firstName, m.empl_lastName) as "Medical Personnel",
concat_ws(' ', r.recipient_firstName, r.recipient_lastName) as "Patient Name",
b.dateOut as "Date Seen"
From medpersonnel m, blood_transaction b, recipient_details r
WHERE m.empl_id = b.empl_id AND r.recipient_id = b.recipient_id

Create view BloodQuantity as Select
donor_details.donor_bloodtype as "Blood Type", sum(blood_donation.quantityDonated) as
"In Stock"
FROM blood_donation join donor_details
on blood_donation.donor_id = donor_details.donor_id
group by donor_details.donor_bloodtype

create table medpersonnel_audits(
  empl_id int NOT NULL,
  empl_lastName varchar(50) NOT NULL,
  changed_on TIMESTAMP(6) NOT NULL,
  foreign key(empl_id) references medpersonnel_credentials(empl_id)
)

CREATE OR REPLACE FUNCTION log_name_changes()
  RETURNS TRIGGER
  LANGUAGE PLPGSQL
  AS
$$
BEGIN
  IF NEW.empl_lastName <> OLD.empl_lastName THEN
     INSERT INTO medpersonnel_audits(empl_id,empl_lastName,changed_on)
     VALUES(OLD.empl_id,OLD.empl_lastName,now());
  END IF;

  RETURN NEW;
END;
$$

CREATE TRIGGER empl_lastName_changes
  BEFORE UPDATE
  ON medpersonnel
  FOR EACH ROW
  EXECUTE PROCEDURE log_name_changes();
```

```sql
UPDATE medpersonnel
SET empl_lastName = 'Brown'
WHERE empl_id = 1003;
```

SQL DML.

```
insert into donor(donor_email, donor_id, donor_firstName, donor_lastName,
donor_username, donor_password)
values('Sai@gmail.com', 2001,'Sia','Ai','Sai@gmail.com','Sai213');

insert into donor(donor_email, donor_id, donor_firstName, donor_lastName,
donor_username, donor_password)
values('hollyasher@gmail.com',
2002,'Holly','Asher','hollyasher@gmail.com','hollyasher123');

insert into donor(donor_email, donor_id, donor_firstName, donor_lastName,
donor_username, donor_password)
values('hollya2003@gmail.com', 2003,'Holly','Asher','hollya2003@gmail.com','123');

insert into donor(donor_email, donor_id, donor_firstName, donor_lastName,
donor_username, donor_password)
values('Rachel.mane@gmail.com',
2004,'Rachel','Mane','rachelmane@gmail.com','rachelmane123');

insert into donor(donor_email, donor_id, donor_firstName, donor_lastName,
donor_username, donor_password)
values('Mimi.kim@gmail.com', 2005,'Mimi','Kim','Mimi.kim@gmail.com','mimikim123');

insert into donor(donor_email, donor_id, donor_firstName, donor_lastName,
donor_username, donor_password)
values('Mary.kent@gmail.com', 2006,'Mary','Kent','Mary.Kent@gmail.com','marykent123');

UPDATE donor SET donor_email = 'Mimikim@gmail.com' WHERE donor_email =
'Mimi.kim@gmail.com';

insert into donor_details(donor_id, donor_number, donor_email, donor_age,
donor_gender, donor_bloodtype, donor_address)
values(2001, '236-144-3655', 'Sai@gmail.com','29', 'Female', 'B+', '8441 Rym Ave');

insert into donor_details(donor_id, donor_number, donor_email, donor_age,
donor_gender, donor_bloodtype, donor_address)
values(2002, '313-291-9392', 'hollyasher@gmail.com','36', 'Female', 'B+', '5427 Ocean
Drive Ave');

insert into donor_details(donor_id, donor_number, donor_email, donor_age,
donor_gender, donor_bloodtype, donor_address)
values(2003, '983-293-9302', 'hollya2003@gmail.com','18', 'Female', 'O+', '283 Rhine
Road');

insert into donor_details(donor_id, donor_number, donor_email, donor_age,
donor_gender, donor_bloodtype, donor_address)
```

```
values(2004, '002-039-2811', 'Rachel.mane@gmail.com','40', 'Female','O+','299 Main
Street');

insert into donor_details(donor_id, donor_number, donor_email, donor_age,
donor_gender, donor_bloodtype, donor_address)
values(2005, '646-293-2918', 'Mimikim@gmail.com', '39', 'Female','B+', '1440 Main
Street');

insert into donor_details(donor_id, donor_number, donor_email, donor_age,
donor_gender, donor_bloodtype, donor_address)
values(2006, '293-299-1929', 'Mary.kent@gmail.com', '38', 'Female', 'B+', '33 Claw
Street');

UPDATE donor_details SET donor_address = '29 Main Street' WHERE donor_id = 2004;


insert into medpersonnel(empl_email, empl_id, empl_firstName, empl_lastName,
empl_username, empl_password)
values('Jim.Mcmanus@donationbank.com',
1001,'Jim','McManus','Jim.Mcmanus@donationbank.com','jimmcmanus123');

insert into medpersonnel(empl_email, empl_id, empl_firstName, empl_lastName,
empl_username, empl_password)
values('James.Hadron@donationbank.com',
1002,'James','Hadron','James.Hadron@donationbank.com','JamesHadron123');

insert into medpersonnel(empl_email, empl_id, empl_firstName, empl_lastName,
empl_username, empl_password)
values('Alex.Coffing@donationbank.com',
1003,'Alex','Coffing','Alex.Coffing@donationbank.com','AlexCoffing123');

insert into medpersonnel(empl_email, empl_id, empl_firstName, empl_lastName,
empl_username, empl_password)
values('Alex.Coughing@donationbank.com',
1004,'Alex','Coughing','Alex.Coughing@donationbank.com','AlexCoughing123');

insert into medpersonnel(empl_email, empl_id, empl_firstName, empl_lastName,
empl_username, empl_password)
values('Mario.Sanchez@donationbank.com',
1005,'Mario','Sanchez','Mario.Sanchez@donationbank.com','MarioSanchez123');

insert into medpersonnel(empl_email, empl_id, empl_firstName, empl_lastName,
empl_username, empl_password)
values('test@donationbank.com', 1006,'test','test','test@donationbank.com','test123');


UPDATE medpersonnel SET empl_password = '123AlexCoughing' WHERE empl_id = 1004;
```

```sql
/*  insert medpersonnel data into medpersonnel_credential table*/
insert into medpersonnel_credentials(empl_id, empl_email, empl_firstName,
empl_lastName, empl_username, empl_password)
values(1001, 'Jim.Mcmanus@donationbank.com', 'John', 'Doe', 'administrator1','1234');

insert into medpersonnel_credentials(empl_id, empl_email, empl_firstName,
empl_lastName, empl_username, empl_password)
values(1002, 'James.Hadron@donationbank.com', 'James',
'Smith','administrator2','1234');

insert into medpersonnel_credentials(empl_id, empl_email, empl_firstName,
empl_lastName, empl_username, empl_password)
values(1003,'Alex.Coffing@donationbank.com',
'Alex','Coffing','administrator3','AlexCoffing123');

insert into medpersonnel_credentials(empl_id, empl_email, empl_firstName,
empl_lastName, empl_username, empl_password)
values(1004,'Alex.Coughing@donationbank.com',
'Alex','Coughing','administrator4','AlexCoughing123');

insert into medpersonnel_credentials(empl_id, empl_email, empl_firstName,
empl_lastName, empl_username, empl_password)
values(1005,'Mario.Sanchez@donationbank.com',
'Mario','Sanchez','administrator4','MarioSanchez123');

insert into medpersonnel_credentials(empl_id, empl_email, empl_firstName,
empl_lastName, empl_username, empl_password)
values(1006,'test@donationbank.com', 'test','test','test@donationbank.com','test123');

UPDATE medpersonnel_credentials SET empl_password = '123AlexCoughing' WHERE empl_id =
1004;

insert into blood_donation(blood_id, donor_id, quantityDonated, blood_dateDonated,
blood_type)
values(1, 2001, 1, '01-01-2022','B+');

insert into blood_donation(blood_id, donor_id, quantityDonated, blood_dateDonated,
blood_type)
values(2, 2002, 1, '01-03-2022','B+');

insert into blood_donation(blood_id, donor_id, quantityDonated, blood_dateDonated,
blood_type)
values(3, 2003, 1, '01-02-2022','O+');

insert into blood_donation(blood_id, donor_id, quantityDonated, blood_dateDonated,
blood_type)
values(4, 2004, 1, '04-01-2022','O+');
```

```sql
insert into blood_donation(blood_id, donor_id, quantityDonated, blood_dateDonated,
blood_type)
values(5, 2003, 1, '10-01-2022','O+');

insert into blood_donation(blood_id, donor_id, quantityDonated, blood_dateDonated,
blood_type)
values(6, 2002, 1, '02-28-2022','B+');

insert into blood_donation(blood_id, donor_id, quantityDonated, blood_dateDonated,
blood_type)
values(7, 2004, 1, '01-28-2022','B+');

insert into blood_donation(blood_id, donor_id, quantityDonated, blood_dateDonated,
blood_type)
values(8, 2005, 1, '12-02-2022','B+');

insert into blood_donation(blood_id, donor_id, quantityDonated, blood_dateDonated,
blood_type)
values(9, 2006, 1, '12-02-2022','O+');

UPDATE blood_donation SET blood_type = 'O+' WHERE donor_id = 2005;

insert into recipient_details(recipient_id, recipient_firstName, recipient_lastName,
recipient_number, recipient_email, recipient_age, recipient_gender,
recipient_bloodtype, recipient_address)
values(3001,'Mark', 'Smith','212-928-0392','marksmith@gmail.com',27,'Male','B+','15
Marks Place');

insert into recipient_details(recipient_id, recipient_firstName, recipient_lastName,
recipient_number, recipient_email, recipient_age, recipient_gender,
recipient_bloodtype, recipient_address)
values(3002,'Mike','Doe','646-927-0392','mikedoe@gmail.com',35,'Male','B+','36 Saint
George Road');

insert into recipient_details(recipient_id, recipient_firstName, recipient_lastName,
recipient_number, recipient_email, recipient_age, recipient_gender,
recipient_bloodtype, recipient_address)
values(3003,'Joe','Campbell','516-293-0293','johnnylee@gmail.com',32,'Male','O+','225
E 34th Street Apt 21G');

insert into recipient_details(recipient_id, recipient_firstName, recipient_lastName,
recipient_number, recipient_email, recipient_age, recipient_gender,
recipient_bloodtype, recipient_address)
values(3004,'Betty','Tai','574-343-0293','bettytai@gmail.com',32,'Female','O+','13675
37 Ave Apt 1B');
```

```sql
insert into recipient_details(recipient_id, recipient_firstName, recipient_lastName,
recipient_number, recipient_email, recipient_age, recipient_gender,
recipient_bloodtype, recipient_address)
values(3005,'Marcia','Elingsworth','212-002-
0002','bettytai@gmail.com',32,'Female','O+','136 Main Street');
/* delete the duplicate registration*/
insert into recipient_details(recipient_id, recipient_firstName, recipient_lastName,
recipient_number, recipient_email, recipient_age, recipient_gender,
recipient_bloodtype, recipient_address)
values(3006,'Maria','Vasquez','212-002-
0002','mariavasquez@gmail.com',32,'Female','O+','13675 37 Ave Apt 1B');

Update recipient_details SET recipient_email = 'MarciaElingsworth@gmail.com' WHERE
recipient_id = 3005;
DELETE FROM recipient_details WHERE recipient_id = 3006;

Update blood_transaction SET dateOut = '02-02-2022' WHERE recipient_id = 3004;
DELETE FROM blood_transaction WHERE transaction_id = 4006;
Delete FROM blood_donation WHERE blood_id = 1;
Delete FROM blood_donation WHERE blood_id = 2;
Delete FROM blood_donation WHERE blood_id = 3;
Delete FROM blood_donation WHERE blood_id = 4;
Delete FROM blood_donation WHERE blood_id = 5;

DELETE FROM medpersonnel_credentials where empl_id =1006;
DELETE FROM medpersonnel where empl_email ='test@donationbank.com';

insert into preexisting_details(donor_preexistingconditions, donor_email)
values('None', 'Sai@gmail.com');

insert into preexisting_details(donor_preexistingconditions, donor_email)
values('Pregnant', 'hollyasher@gmail.com');

insert into preexisting_details(donor_preexistingconditions, donor_email)
values('Diabetic', 'hollya2003@gmail.com');

insert into preexisting_details(donor_preexistingconditions, donor_email)
values('Unknown', 'Rachel.mane@gmail.com');

insert into preexisting_details(donor_preexistingconditions, donor_email)
values('Unknown', 'Mimikim@gmail.com');

insert into preexisting_details(donor_preexistingconditions, donor_email)
values('HIV', 'Mary.kent@gmail.com');

UPDATE preexisting_details SET donor_preexistingconditions = 'None' WHERE donor_email
= 'Mimikim@gmail.com';
DELETE FROM blood_donation where donor_id = 2006;
```

```sql
DELETE FROM preexisting_details WHERE donor_email = 'Mary.kent@gmail.com';
DELETE FROM donor_details where donor_email = 'Mary.kent@gmail.com';
DELETE FROM donor where donor_email = 'Mary.kent@gmail.com';
```

Queries.

```
/* What is the patient list of information for the donors? */
Select donor.donor_firstName, donor.donor_lastName, donor_details.donor_number,
donor_details.donor_email, donor_details.donor_age, donor_details.donor_gender,
donor_details.donor_bloodtype, donor_details.donor_address
From donor INNER JOIN donor_details ON donor.donor_id = donor_details.donor_id


/* What are the donor id's of the people named Holly?*/
SELECT donor.donor_id
FROM donor
WHERE donor.donor_firstName='Holly' AND donor.donor_lastName='Asher';

/* What are the first names and last names of the people who have donated? */
select r.recipient_firstName, r.recipient_lastName, r.recipient_bloodtype, b.dateOut
FROM recipient_details r, blood_transaction b
WHERE b.recipient_id = r.recipient_id

/*What are the medical personnel credentials?*/
select medpersonnel.empl_username, medpersonnel.empl_password
FROM medpersonnel
INNER JOIN  medpersonnel_credentials on medpersonnel.empl_id =
medpersonnel_credentials.empl_id

/*Find the credentials of the medical personnel who are not null.*/
select medpersonnel_credentials.empl_username, medpersonnel_credentials.empl_password
FROM  medpersonnel_credentials
WHERE medpersonnel_credentials.empl_id is not NULL

/* What are the pre-exisitng conditions of the people in the donor list?*/
select donor.donor_firstName, donor.donor_lastName,
preexisting_details.donor_preexistingconditions
FROM  donor, preexisting_details
WHERE donor.donor_email = preexisting_details.donor_email

/*What are the usernames of the donors who do not have "None" as their pre-exisiting
condition?*/
select donor.donor_username
FROM  donor, preexisting_details
WHERE donor.donor_email = preexisting_details.donor_email AND NOT
preexisting_details.donor_preexistingconditions='None'

/*What are the dates in and out of the blood relative to the transaction_id and
recipient_id?*/
select blood_transaction.transaction_id, blood_transaction.recipient_id,
blood_donation.blood_dateDonated, blood_transaction.dateOut
FROM  blood_donation INNER JOIN blood_transaction
```

```
ON blood_donation.blood_id = blood_transaction.blood_id
```

Application Code.

CapstoneProject

February 7, 2022

[5]:
```
pip install psycopg2-binary --no-cache-dir
```

```
Requirement already satisfied: psycopg2-binary in
/Users/flushingmacmini/opt/anaconda3/lib/python3.9/site-packages (2.9.2)
Note: you may need to restart the kernel to use updated packages.
```

[1]:
```
import psycopg2
```

[27]:
```
conn = psycopg2.connect(
    host="localhost",
    database="blood_donation_db",
    user="postgres",
    password="NasaDog2020@")
```

[28]:
```
cur = conn.cursor()
```

[29]:
```
cur.execute("SELECT * FROM donor;")
cur.fetchall()
```

[29]:
```
[('Sai@gmail.com', 2001, 'Sia', 'Ai', 'Sai@gmail.com', 'Sai213'),
 ('hollyasher@gmail.com',
  2002,
  'Holly',
  'Asher',
  'hollyasher@gmail.com',
  'hollyasher123'),
 ('hollya2003@gmail.com',
  2003,
  'Holly',
  'Asher',
  'hollya2003@gmail.com',
  '123'),
 ('Rachel.mane@gmail.com',
  2004,
  'Rachel',
  'Mane',
  'rachelmane@gmail.com',
  'rachelmane123')]
```

[30]:
```
##Application Code demonstrating the insert
cur.execute("""
    INSERT INTO recipient_details(recipient_id, recipient_firstName,␣
↪recipient_lastName, recipient_number, recipient_email, recipient_age,␣
↪recipient_gender, recipient_bloodtype, recipient_address)
    values(%s, %s, %s, %s, %s, %s, %s, %s, %s);
    """,
    ("3004","Betty","Tai","5743430293","bettytai@gmail.
↪com",32,"Female","O+","13675 37 Ave Apt 1B"))
```

[31]:
```
##Application Code demonstrating that the insert worked.
cur.execute("SELECT * FROM recipient_details;")
cur.fetchall()
```

[31]:
```
[(3001,
  'Mark',
  'Smith',
  '2129280392',
  'marksmith@gmail.com',
  27,
  'Male',
  'B+',
  '15 Marks Place'),
 (3002,
  'Mike',
  'Doe',
  '6469270392',
  'mikedoe@gmail.com',
  35,
  'Male',
  'B+',
  '36 Saint George Road'),
 (3003,
  'Joe',
  'Campbell',
  '5162930293',
  'johnnylee@gmail.com',
  32,
  'Male',
  'O+',
  '225 E 34th Street Apt 21G'),
 (3004,
  'Betty',
  'Tai',
  '5743430293',
  'bettytai@gmail.com',
  32,
```

```
  'Female',
  'O+',
  '13675 37 Ave Apt 1B')]
```

[36]:
```
##This triggers the stored procedure of auditing the last name of the employee.
cur.execute(
  "UPDATE medpersonnel SET empl_lastName=(%s)"
  " WHERE empl_id = (%s)",
  ("Black","1003",));
```

[37]:
```
cur.execute("SELECT * FROM medpersonnel_audits;")
cur.fetchall()
```

[37]:
```
[(1003, 'Coffing', datetime.datetime(2022, 2, 7, 14, 0, 12, 574132)),
 (1003, 'Brown', datetime.datetime(2022, 2, 7, 15, 24, 8, 781071))]
```

[ ]:

EagleMail

Betty Tai <betty.tai@eastern.edu>

DTSC691 Project Proposal

Michael Morabito <mmorabit@eastern.edu> Tue, Feb 1, 2022 at 6:08 PM
To: Betty Tai <betty.tai@eastern.edu>
Cc: Greg Longo <gregory.longo@eastern.edu>, Amy Berrios <amy.berrios@eastern.edu>

Hi Betty,

Great.

Pick an enterprise to design a database for and specify who are the intended users of the database.
Create a list of entities and relationship sets.
Create a sample ER diagram and/or relational database schema diagram.

You have permission to use the psycopg2 package to connect to the database, which I have disallowed for all other students.
If you can simply satisfy the requirements outlined in that document using the psycopg2 package to connect to the database from a python script, then you can pass this class.

Be sure to resubmit a new proposal in the coming days with the information herein requested.

- Dr. Morabito

[Quoted text hidden]