**Winter Semester 2022**

**CSE250 Database Management System**

**Project Title: Blood Donation Record Database**

AU2040067 Astha Bhalodiya

AU2040076 Shreya Karia

AU2040103 Omikumar Makadia

# **a. Description of Project: Describe the project in detail. Explain each functionality that is included in the project. (System Requirement Specification)**

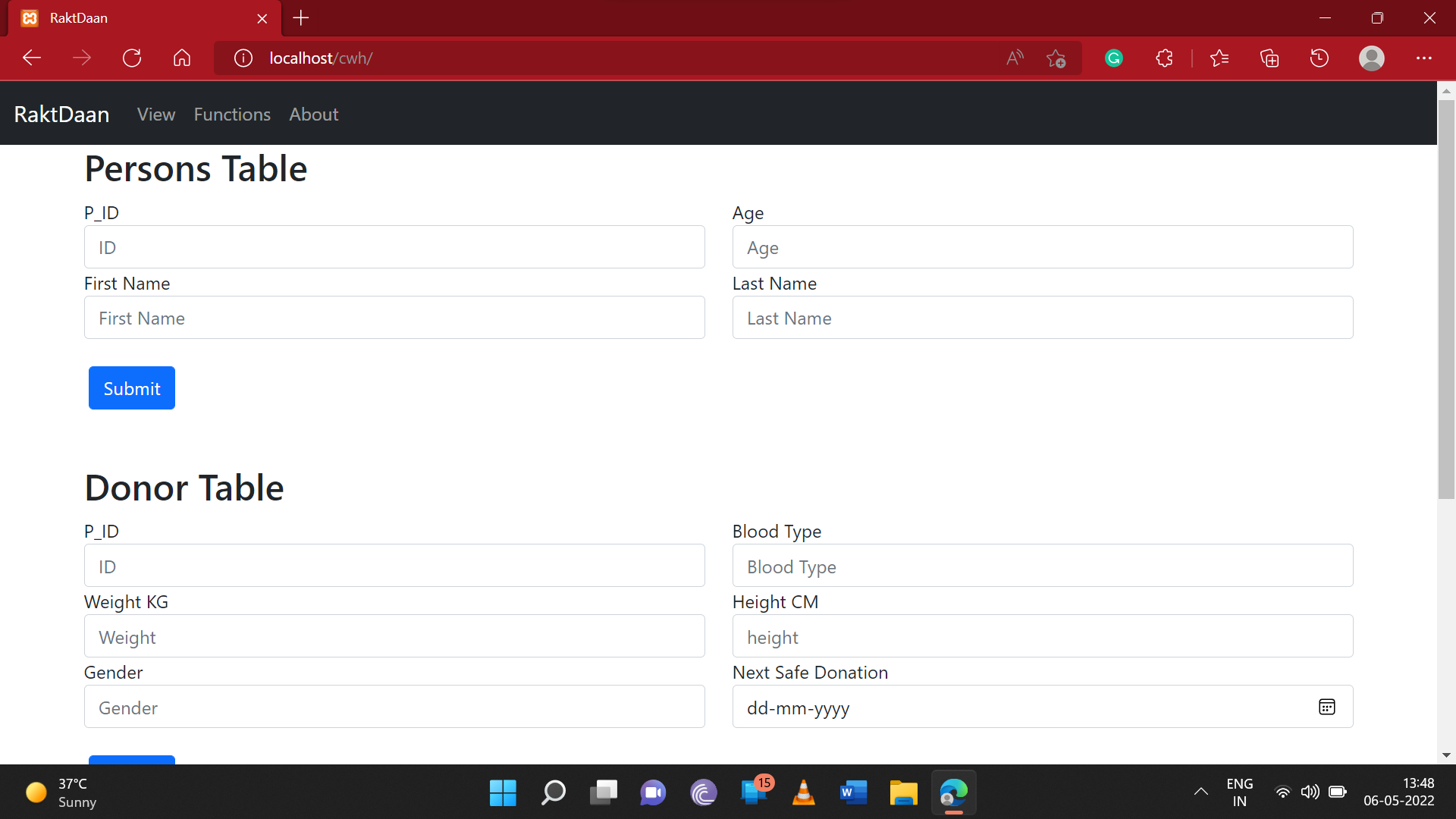
Our project is a Blood Donation System. Blood Donation is carried out by various NGOs, Governments, and hospitals worldwide. The blood, once collected, is stored and used when required. The process of registering a donor, collecting the blood, storing it, and making it available when required contributes to a lot of data. Our project is based on the same problem statement. The database is designed to make all the maintenance of the same data efficient. It makes it easy to retrieve the necessary data in accordance with need. It also keeps track of the history of a patient or donor. It also makes it easy to keep data on when one can donate and when one cannot.

The project is also based on our learnings from CSE250: Database Management System. We incorporated our learnings of database design, procedures, functions, and triggers with some frontend development to produce our final project. So the project is also aimed at conveying our learnings.

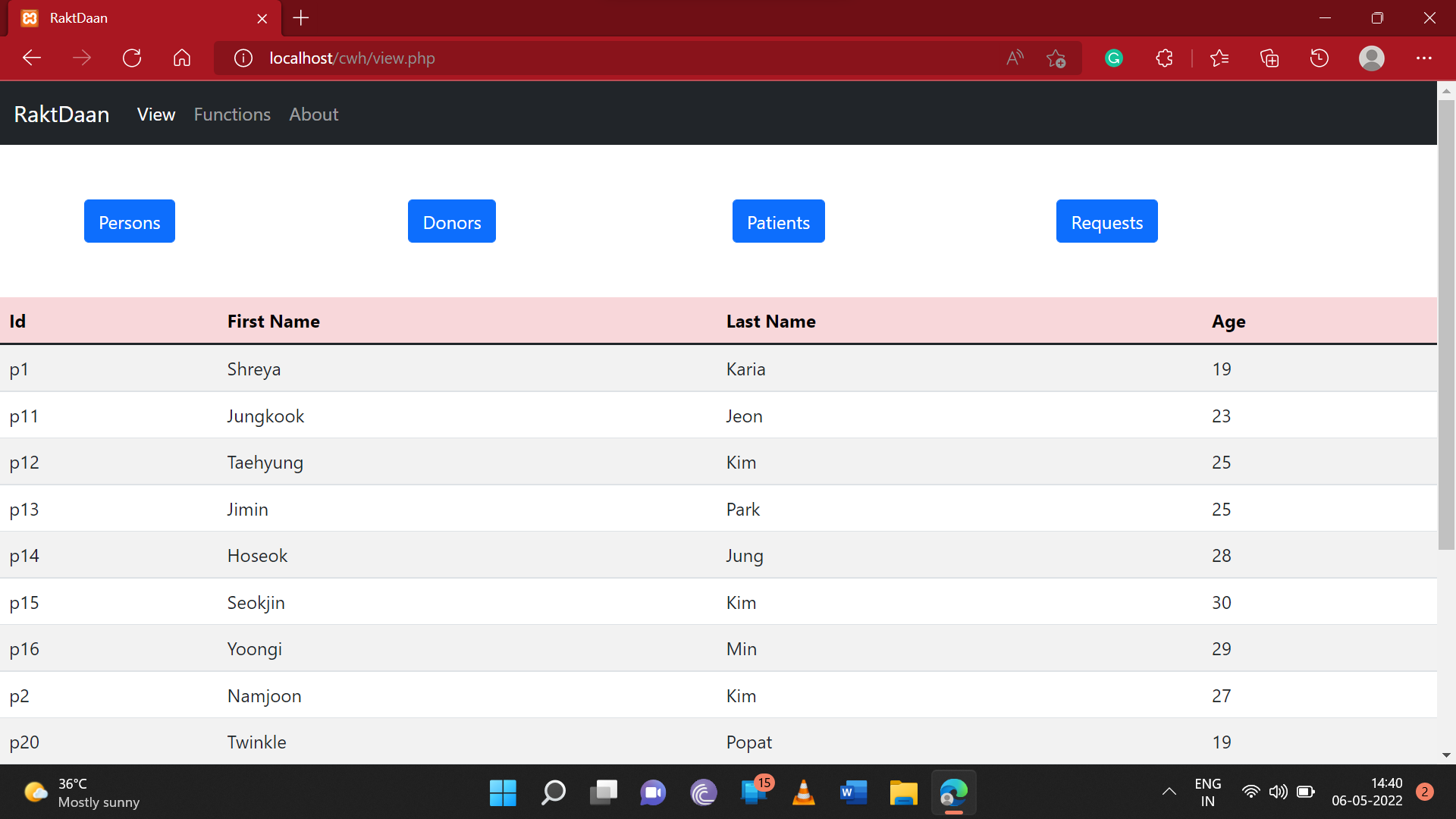
**Overview:**

Our frontend interface consists of 4 pages namely: Home, View, Functions, and About.

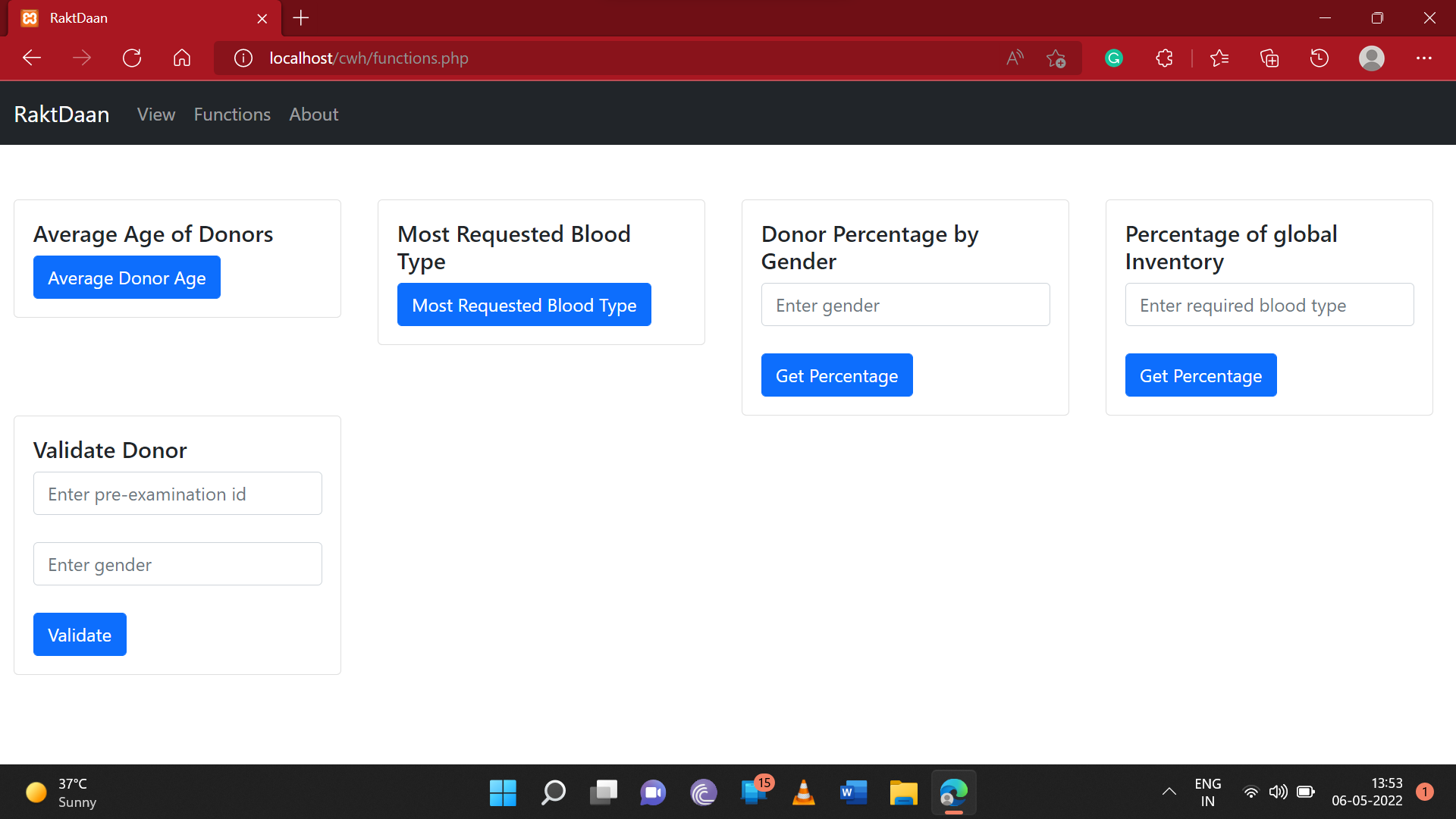
HOME PAGE: It is used for Insert operation and also to display the triggers regarding the same.

****

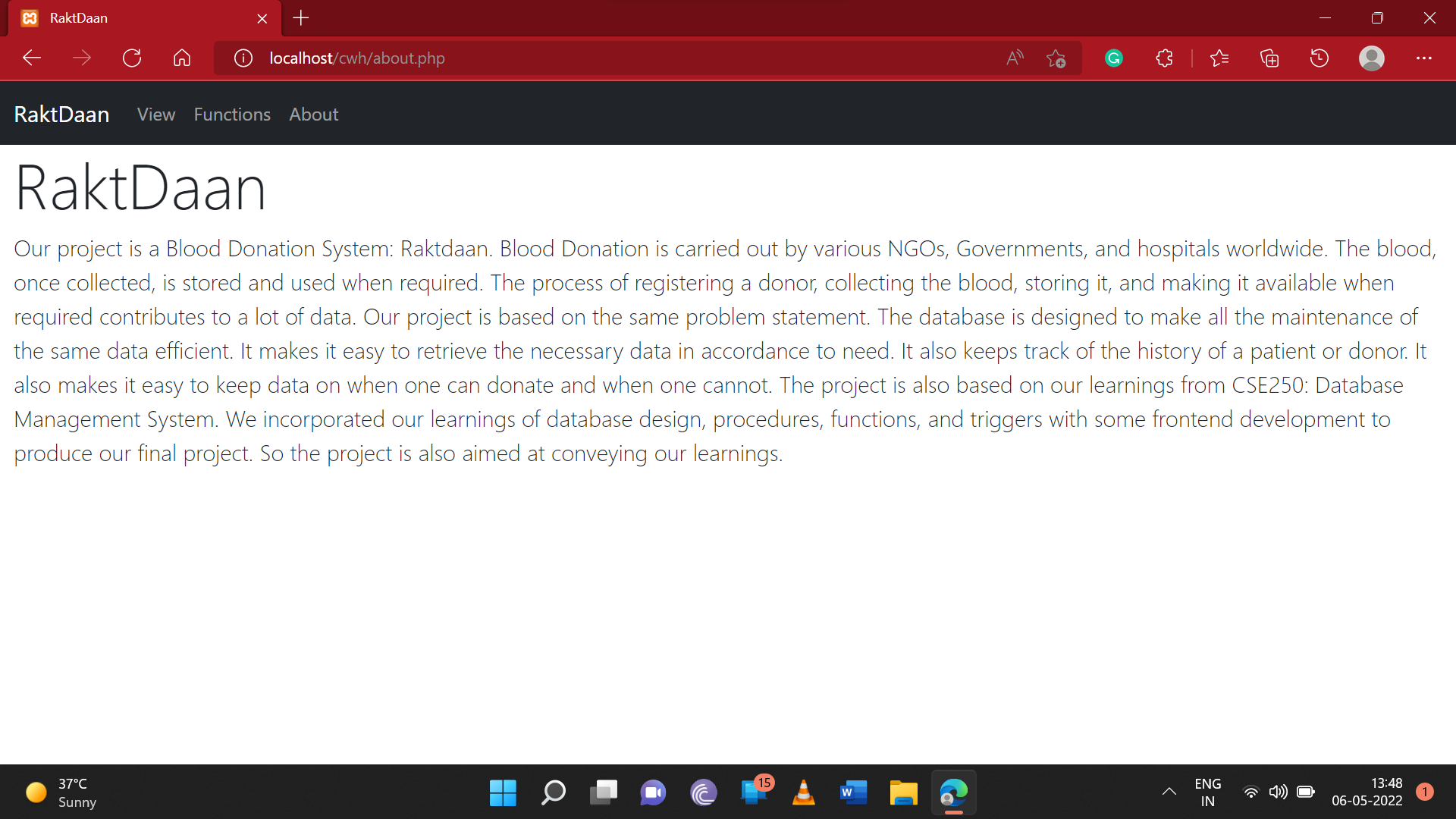
View Page: It is used for Read operations for various tables

****

FUNCTIONS PAGE: It is used to display the functionalities of the functions.



ABOUT PAGE: It has a brief insight into our project.



**System Requirement Specifications:**

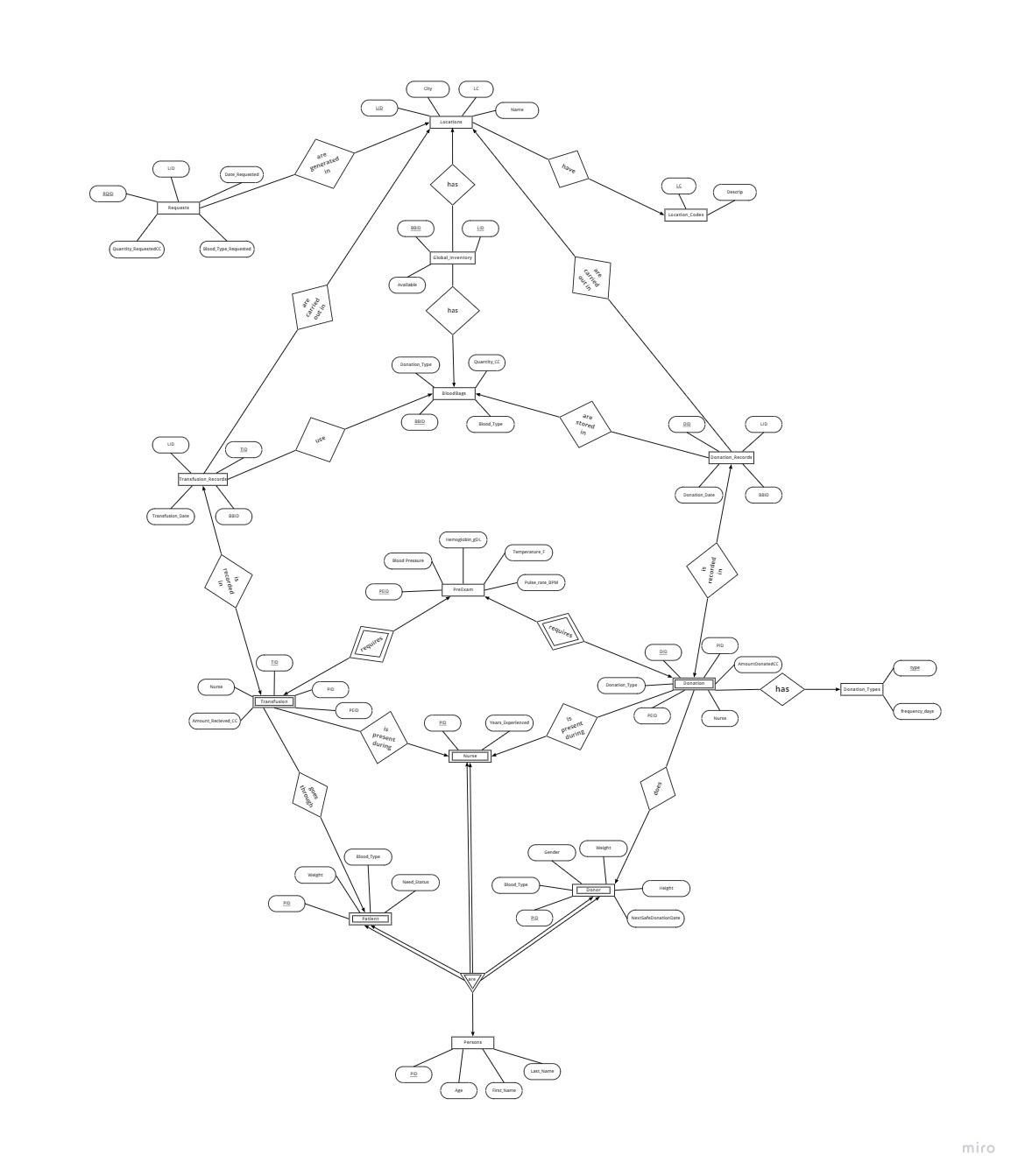
* BACKEND:

We have used MySQL database to build our database.

* FRONTEND:

We have used HTML CSS incorporated in PHP for our backend. We connected our front end with the backend through XAMPP on the local server.

# **b. Entity-Relationship Diagram (Image)**



We can see the following relationships:

Transfusion M:1 Patient

Transfusion M:1 Nurse

Donor 1:M Donation

Nurse 1:M Donation

Donation M:1 Donation\_Types

Donation 1:1 PreExam

Transfusion 1:1 PreExam

Transfusion 1:1 Transfusion\_Records

Donation 1:1 Donation\_Records

BloodBags 1:M Transfusion\_Records

BloodBags 1:M Donation\_Records

Locations 1:M Transfusion\_Records

Locations 1:M Donation\_Records

BloodBags 1:M Global\_Inventory

Locations 1:M Global\_Inventory

Locations M:1 Location\_Codes

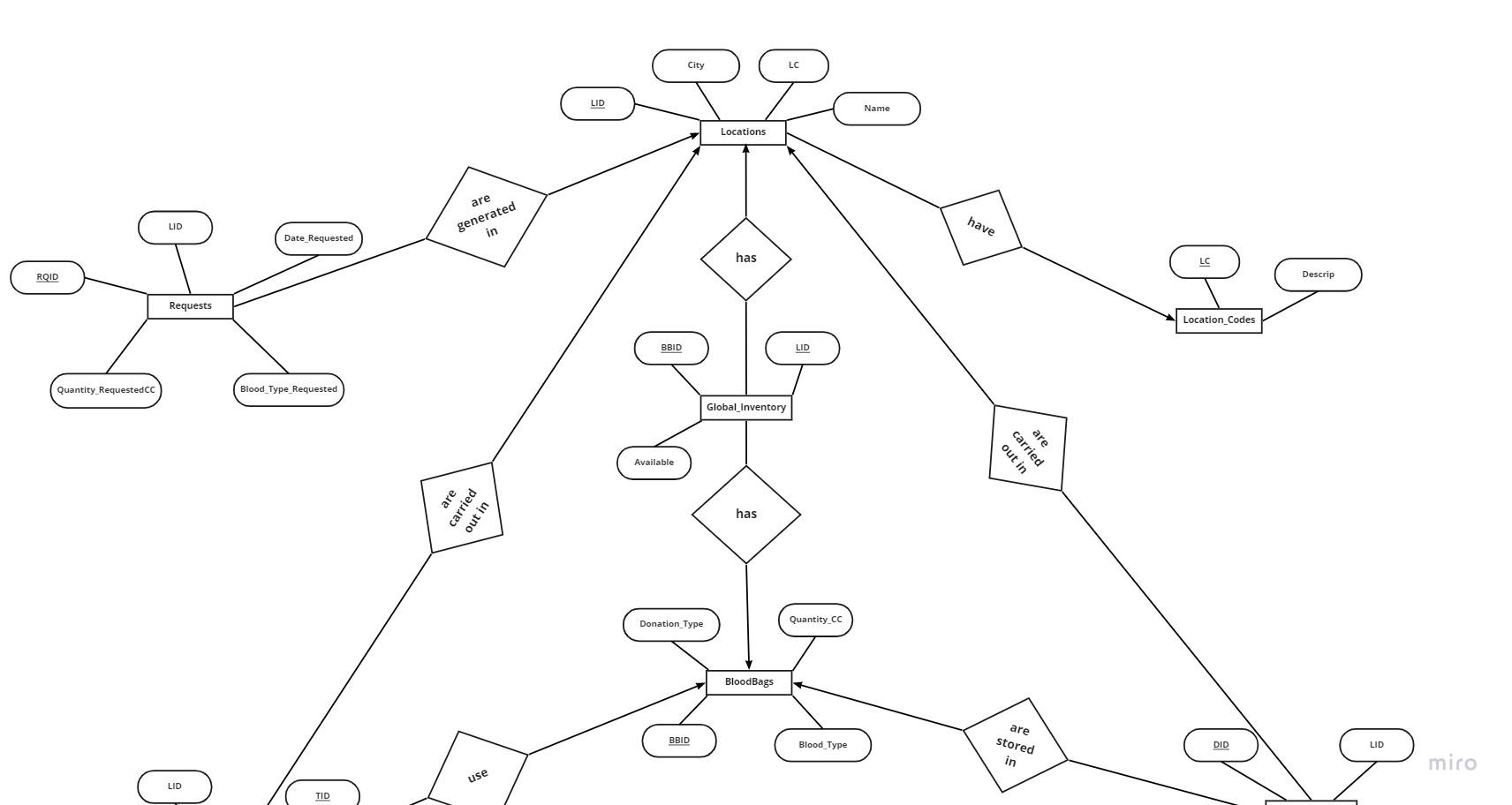
Locations 1:M Requests

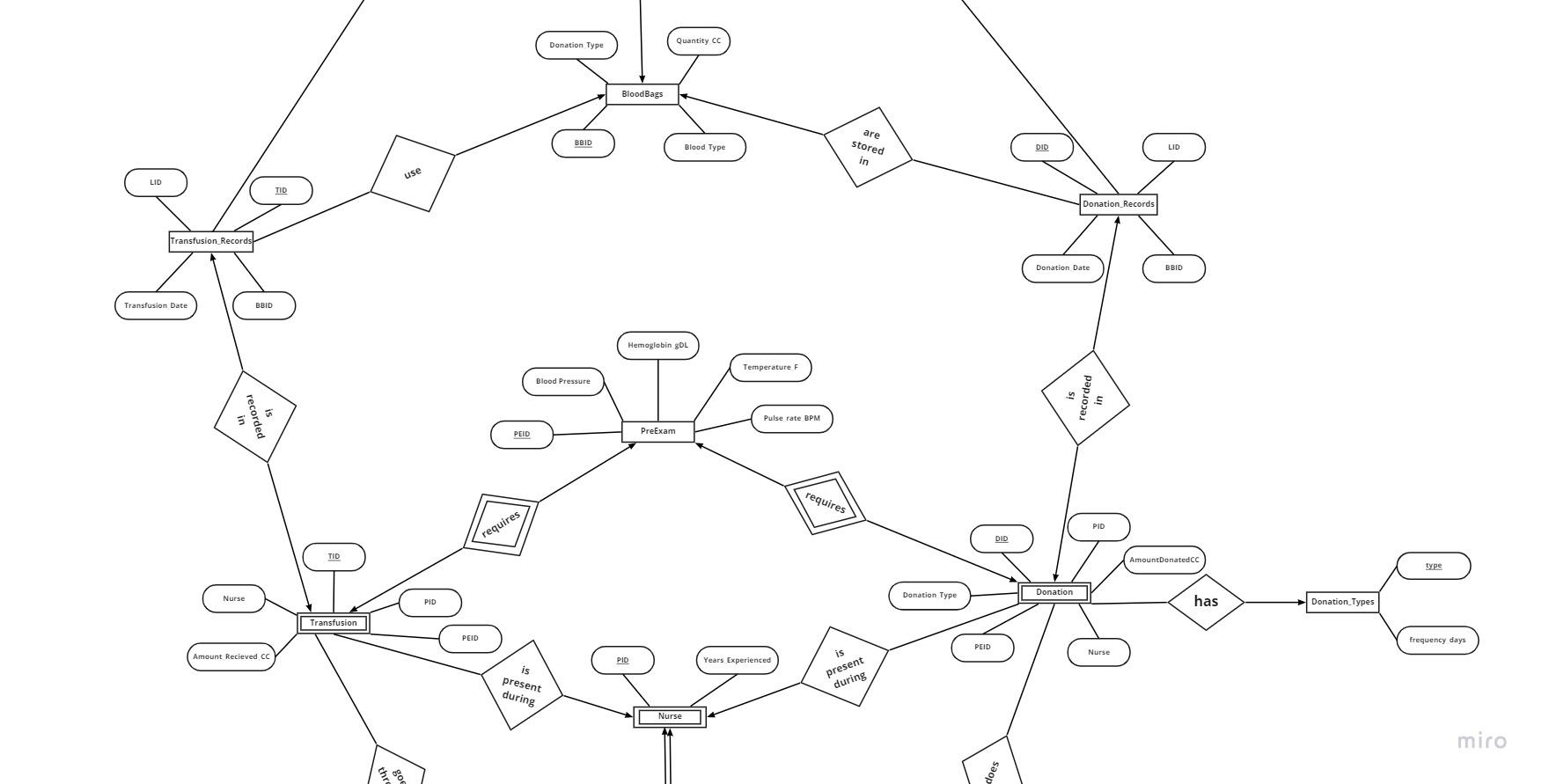
Donor 1:1 Persons

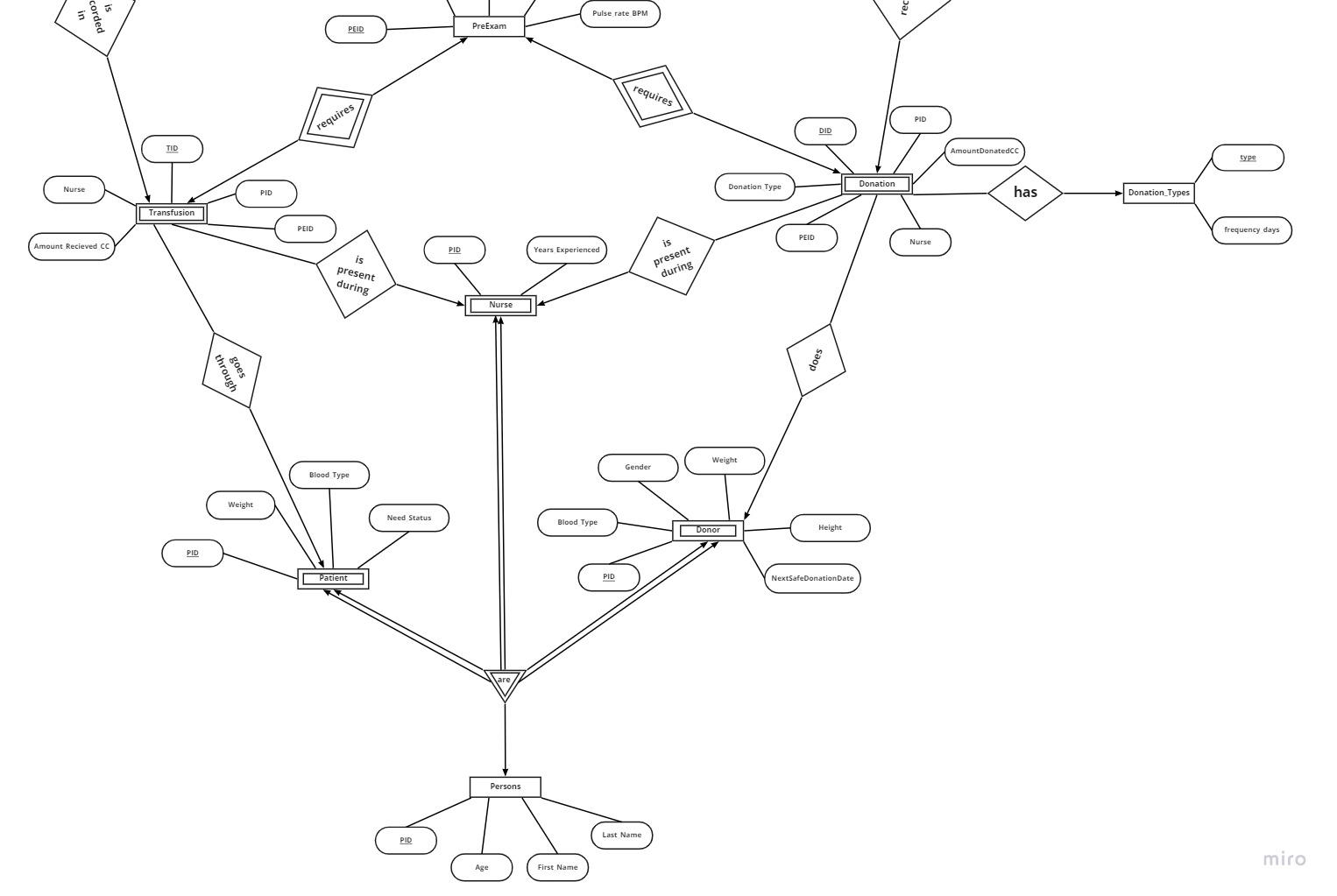
Patient 1:1 Persons

Nurse 1:1 Persons

A clearer view:



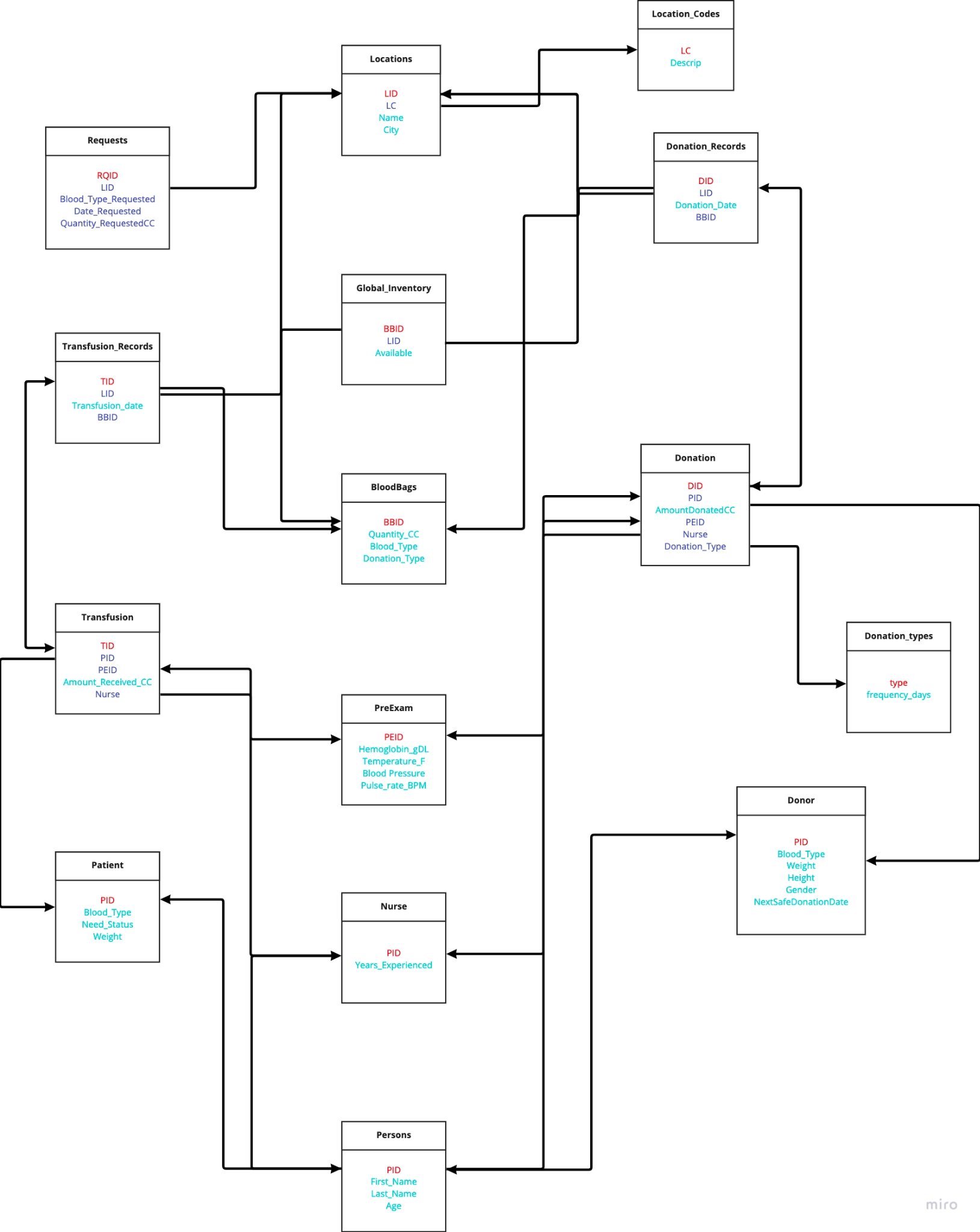




* Persons can be of three types: Donor, Patient, and Nurse.
* All Donors are Persons, but all Persons are not Donors, so the Donor table has Total Participation, and the Persons table has Partial Participation.
* All Nurses are Persons, but all Persons are not Nurses, so the Nurse table has Total Participation, and the Persons table has Partial Participation.
* All Patients are Persons, but all Persons are not Patients, so the Patient table has Total Participation, and the Persons table has Partial Participation.
* Double arrow connected to Donor, Patient, and Nurse shows their full participation in the relationship.
* Since Donors, Nurses, and Patients have total participation, they are weak entities and their corresponding relationship “are” is a strong relationship.
* Pre-Exam is compulsory before Donation and Transfusion, so it is a strong entity, and Donation and Transfusion are weak entities, and the corresponding relationship “requires” is a strong relationship.
* All the relations are in 3NF. There are no partial dependencies and no composite attributes.

<https://miro.com/welcomeonboard/dGczWXRlajZaajVXTWNXRW9MUldFR2kyNzJZZGpXVFJieGYxSko0WU9RbnhicExLNVhQUjhPQmYycjJTcGprZHwzNDU4NzY0NTIxMDAzMzUwMzg3?share_link_id=96282314787>

# **c. Table Design (Data Dictionary)**



<https://miro.com/app/board/uXjVO5p19xw=/>

**Tables:**

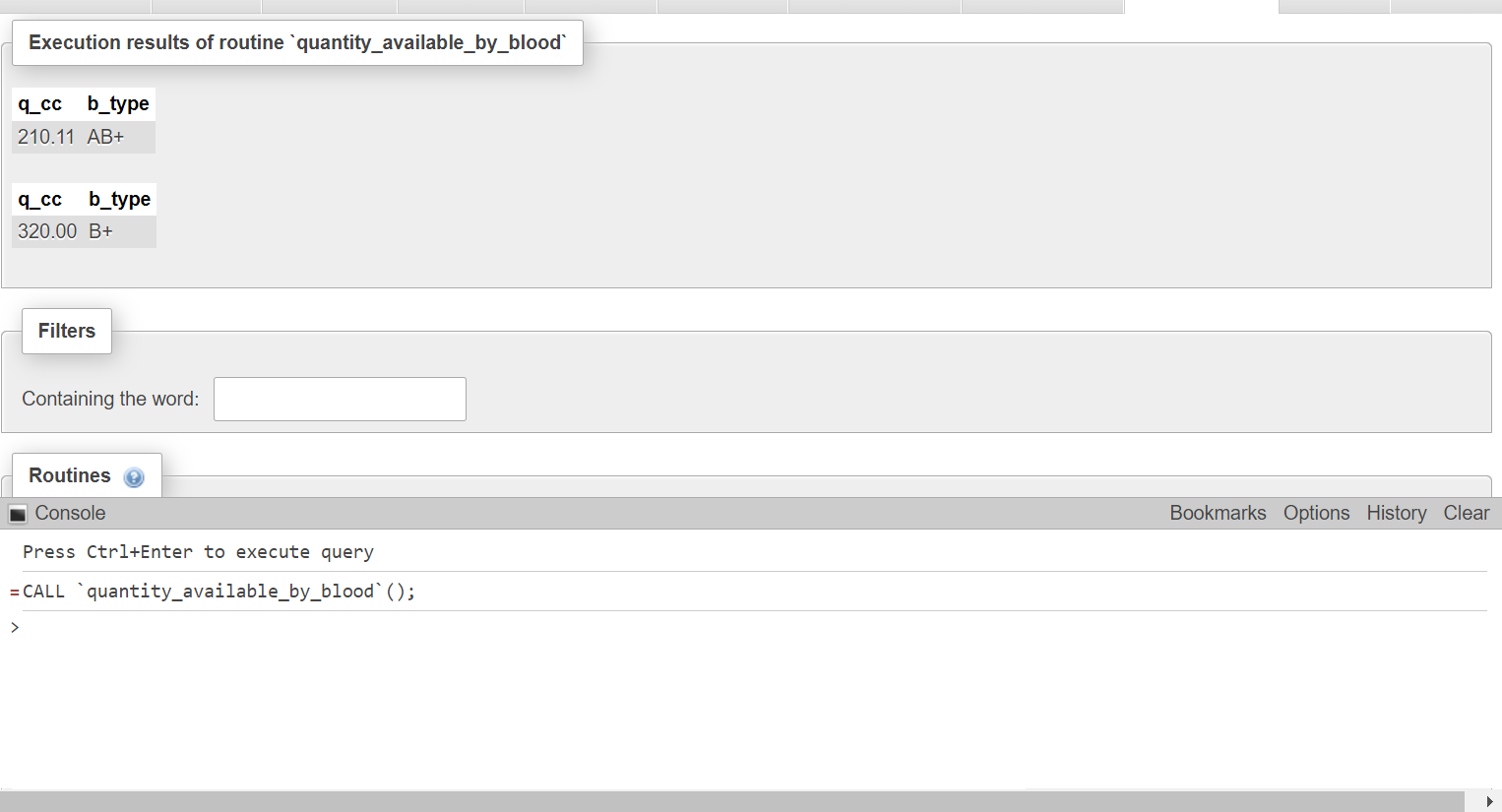
We have made a total of 15 Tables.

| Table name | Description |
| --- | --- |
| Persons | The Persons table contains all the people and their common attributes. There are three subtypes for the persons table: patient, donor, and nurse. |
| Donor | The Donor table contains the information required to be a donor. |
| Patient | The patient table contains all the patients and their information required before a blood transfusion. The need status field indicates whether they require blood on high or low priority. |
| Nurse | The nurse table contains all the nurses and their years of experience. |
| PreExam | The PreExam table contains the respective information about a donor before a donation, as well as a patient before a transfusion. |
| Donation | The Donation table contains the basic attributes about a blood donation. The Donation\_Type references the type of blood donation. |
| Donation\_types | The donation\_type table contains the four different types of blood donation types, as well as the frequency/wait time in which the donor must wait before donating that type again. |
| Transfusion | The Transfusion table contains the basic attributes about a blood transfusion. |
| BloodBags | The BloodBags table contains the basic attributes about each blood bag. Each blood bag is labeled with: the blood type, the quantity, and the type of donation. |
| Locations | The Locations table contains all the locations, as well as a code to describe the type of location. |
| Location\_Codes | The Location\_Codes table contains a four-character code describing the type of location. |
| Global\_Inventory | The Global\_Inventory table contains the global inventory of all blood bags with the location in which they are stored. |
| Requests | The Requests table contains attributes describing a “request” from a location. |
| Donation\_Records | The Donation\_Records table provides a more detailed record of all the donations. |
| Transfusion\_Records | The Transfusion\_Records table provides a more detailed record of all the transfusions. |

# **d. Stored Procedure, Functions, and Triggers (With code and statement to call procedure, a function written on front-end), Screenshots of results generated after procedure and function are called on the front end, and errors generated on the front-end when the trigger is fired.**

* **Procedures**

1. **quantity\_available\_by\_blood**

****

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `quantity\_available\_by\_blood`()

BEGIN

DECLARE b\_type CHAR(3);

DECLARE q\_cc DECIMAL(5,2);

DECLARE exit\_loop BOOLEAN DEFAULT FALSE;

DECLARE cur1 CURSOR FOR SELECT SUM(quantity\_CC), blood\_type FROM bloodbags WHERE bbid in (SELECT bbid FROM global\_inventory WHERE available = true) GROUP BY blood\_type;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit\_loop = TRUE;

OPEN cur1;

availability: LOOP

FETCH FROM cur1 INTO q\_cc,b\_type;

IF exit\_loop THEN

LEAVE availability;

END IF;

SELECT q\_cc,b\_type;

END LOOP availability;

CLOSE cur1;

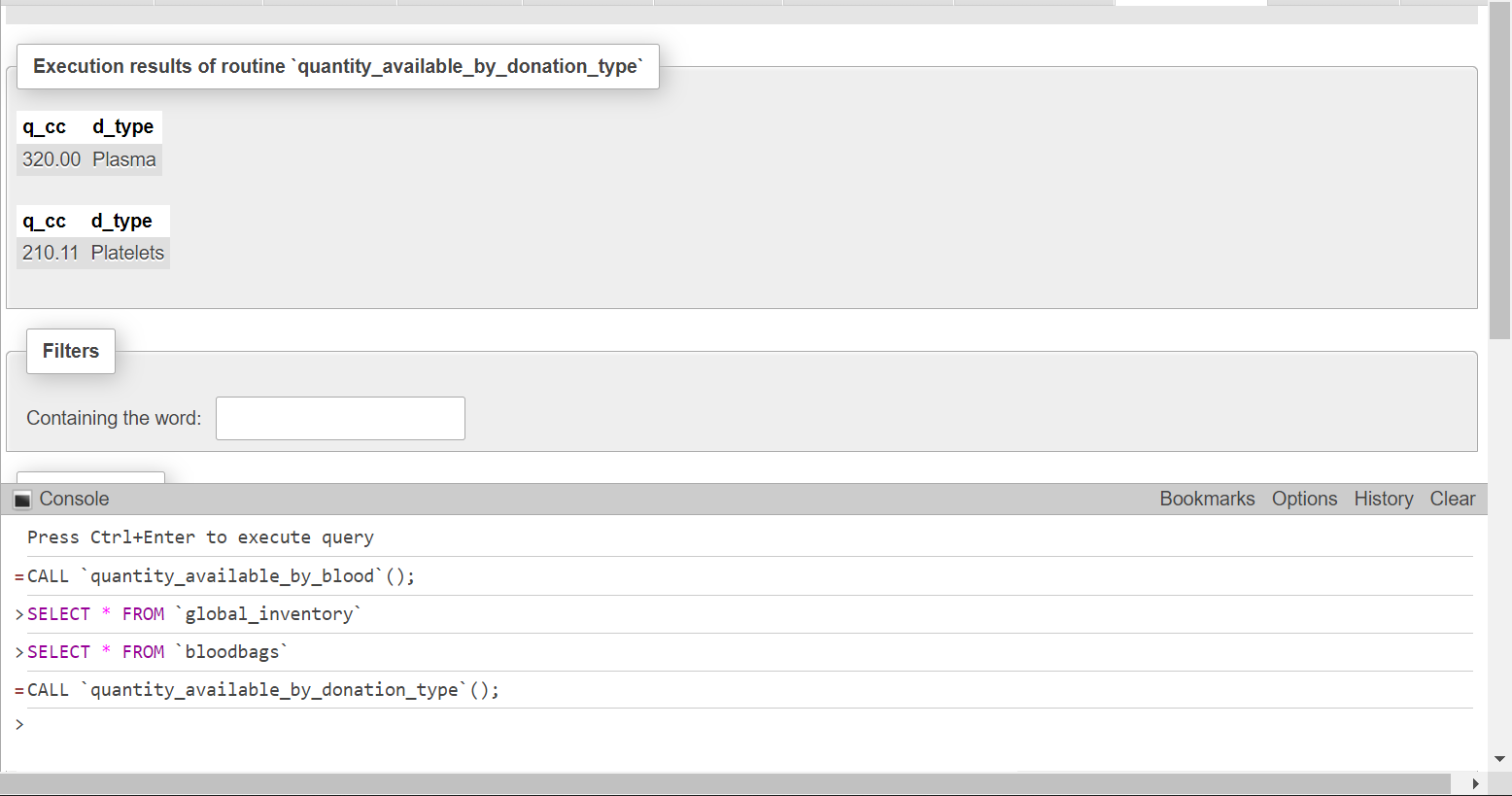
END$$

DELIMITER ;

**Explanation:**

The procedure provides details of the quantity of blood available for different blood types like O+, A+, etc.

1. **quantity\_available\_by\_donation\_type**

****

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `quantity\_available\_by\_donation\_type`()

BEGIN

DECLARE d\_type CHAR(20);

DECLARE q\_cc DECIMAL(5,2);

DECLARE exit\_loop BOOLEAN DEFAULT FALSE;

DECLARE cur1 CURSOR FOR SELECT SUM(quantity\_CC), donation\_type FROM bloodbags WHERE bbid in (SELECT bbid FROM global\_inventory WHERE available = true) GROUP BY donation\_type;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit\_loop = TRUE;

OPEN cur1;

availability: LOOP

FETCH FROM cur1 INTO q\_cc,d\_type;

IF exit\_loop THEN

LEAVE availability;

END IF;

SELECT q\_cc,d\_type;

END LOOP availability;

CLOSE cur1;

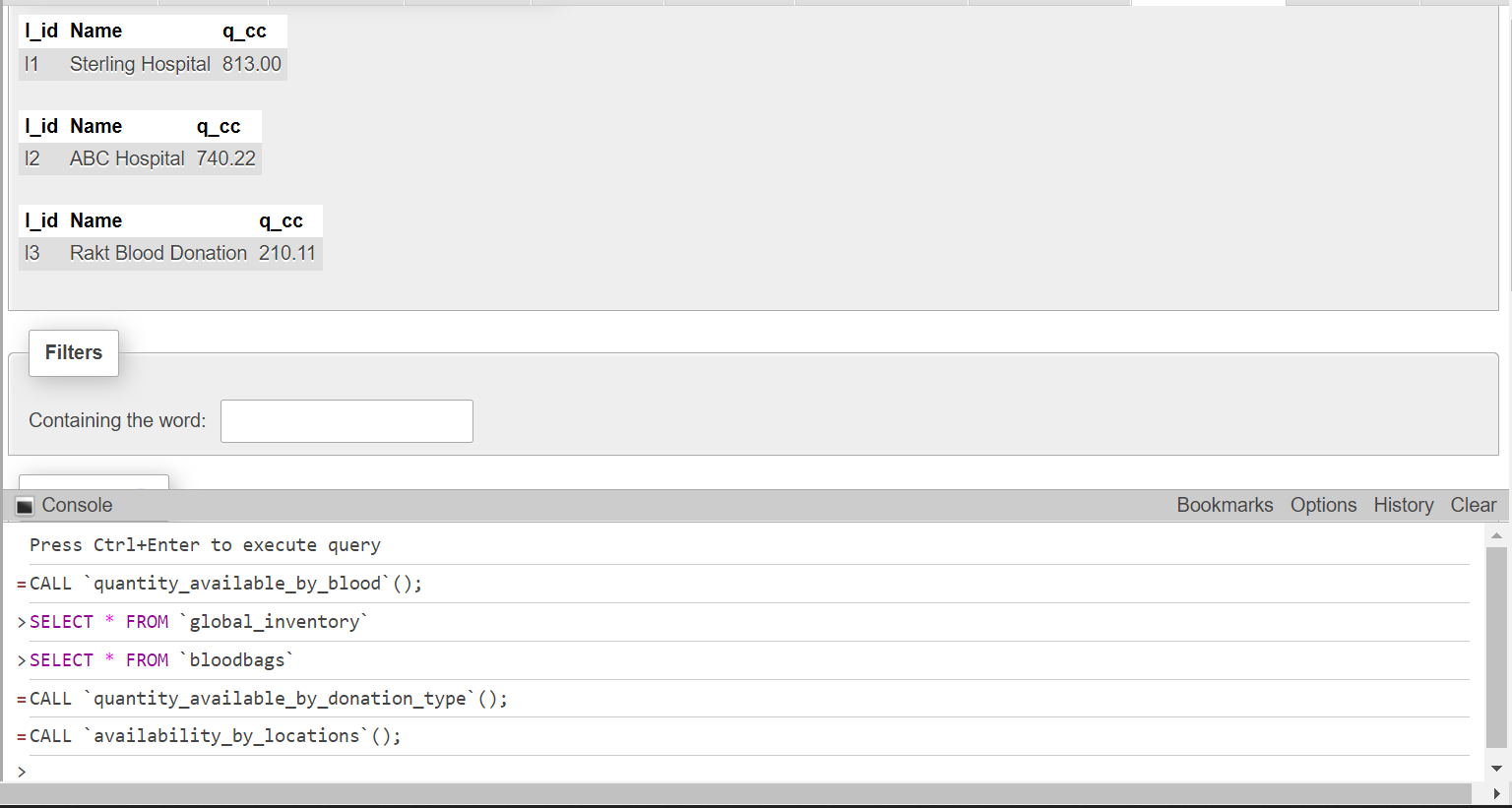
END$$

DELIMITER ;

**Explanation:**

The procedure provides details of the quantity of blood, plasma, platelets, etc available after the donation.

1. **availability\_by\_locations**



DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `availability\_by\_locations`()

BEGIN

DECLARE l\_id CHAR(4);

DECLARE q\_cc DECIMAL(5,2);

DECLARE Name TEXT;

DECLARE exit\_loop BOOLEAN DEFAULT FALSE;

DECLARE cur1 CURSOR FOR SELECT global\_inventory.lid, locations.name, SUM(bloodbags.quantity\_CC) FROM locations INNER JOIN global\_inventory ON locations.lid = global\_inventory.lid INNER JOIN bloodbags on global\_inventory.bbid = bloodbags.bbid GROUP BY lid;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit\_loop = TRUE;

OPEN cur1;

top\_loc: LOOP

FETCH FROM cur1 INTO l\_id,Name,q\_cc;

IF exit\_loop THEN

LEAVE top\_loc;

END IF;

SELECT l\_id,Name,q\_cc;

END LOOP top\_loc;

CLOSE cur1;

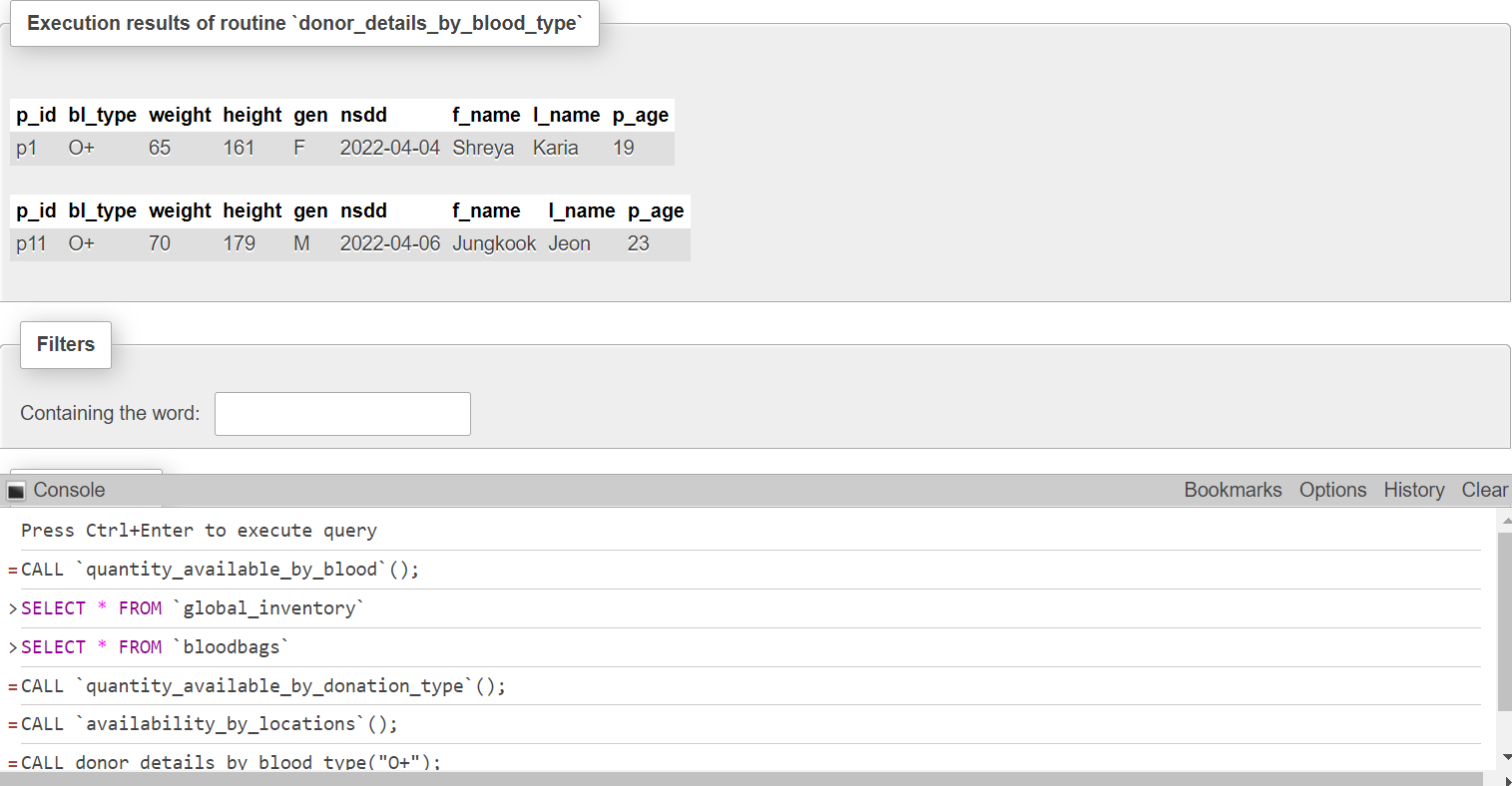
END$$

DELIMITER ;

**Explanation:**

The procedure provides details of the quantity of donation (it can be blood,plasma, platelets,etc and it can be any blood group) available at different locations.

1. **donor\_details\_by\_blood\_type**

****

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `donor\_details\_by\_blood\_type`(IN `bd` CHAR(3))

BEGIN

DECLARE p\_id CHAR(8);

DECLARE bl\_type CHAR(3);

DECLARE weight INT;

DECLARE height INT;

DECLARE gen CHAR(1);

DECLARE nsdd DATE;

DECLARE f\_name TEXT;

DECLARE l\_name TEXT;

DECLARE p\_age INT;

DECLARE exit\_loop BOOLEAN DEFAULT FALSE;

DECLARE cur1 CURSOR FOR SELECT persons.pid, blood\_type, donor.weight, donor.height, gender, nextSafeDonation, first\_name, last\_name, age FROM persons INNER JOIN donor ON persons.pid = donor.pid;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit\_loop = TRUE;

OPEN cur1;

bldonation\_type:LOOP

FETCH FROM cur1 INTO p\_id,bl\_type,weight,height,gen,nsdd,f\_name,l\_name,p\_age;

IF exit\_loop THEN

LEAVE bldonation\_type;

END IF;

IF bl\_type=bd THEN

SELECT p\_id,bl\_type,weight,height,gen,nsdd,f\_name,l\_name,p\_age;

END IF;

END LOOP bldonation\_type;

CLOSE cur1;

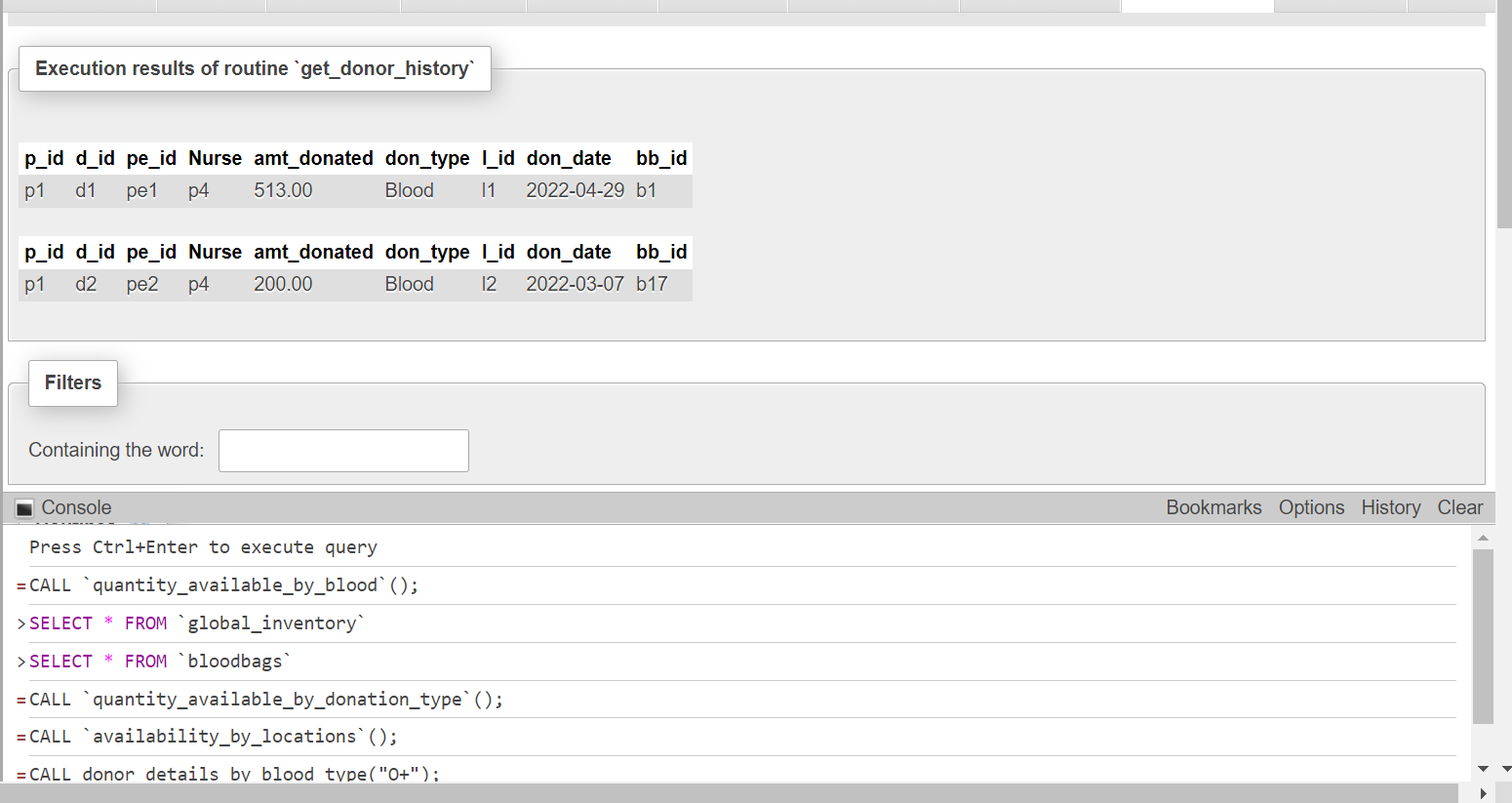
END$$

DELIMITER ;

**Explanation:**

The procedure provides details of all the donors for a specific blood type. Here we take input as blood type, i.e., O+, A+, etc and for that given blood type it displays all the details of the donor.

1. **get\_donor\_history**

****

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `get\_donor\_history`(IN `id` VARCHAR(8))

BEGIN

DECLARE p\_id CHAR(8);

DECLARE d\_id CHAR(8);

DECLARE pe\_id CHAR(8);

DECLARE Nurse CHAR(8);

DECLARE amt\_donated DECIMAL(5,2);

DECLARE don\_type char(20);

DECLARE l\_id CHAR(4);

DECLARE don\_date DATE;

DECLARE bb\_id CHAR(10);

DECLARE exit\_loop BOOLEAN DEFAULT FALSE;

DECLARE cur1 CURSOR FOR SELECT pid, donation.did, peid,donation.nurse,amount\_donated\_CC,donation\_type,lid,donation\_date,bbid FROM donation INNER JOIN donation\_records ON donation.did = donation\_records.did;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit\_loop = TRUE;

OPEN cur1;

donation\_type: LOOP

FETCH FROM cur1 INTO p\_id,d\_id,pe\_id,Nurse,amt\_donated,don\_type,l\_id,don\_date,bb\_id;

IF exit\_loop THEN

LEAVE donation\_type;

END IF;

IF p\_id=id THEN

SELECT p\_id,d\_id,pe\_id,Nurse,amt\_donated,don\_type,l\_id,don\_date,bb\_id;

END IF;

END LOOP donation\_type;

CLOSE cur1;

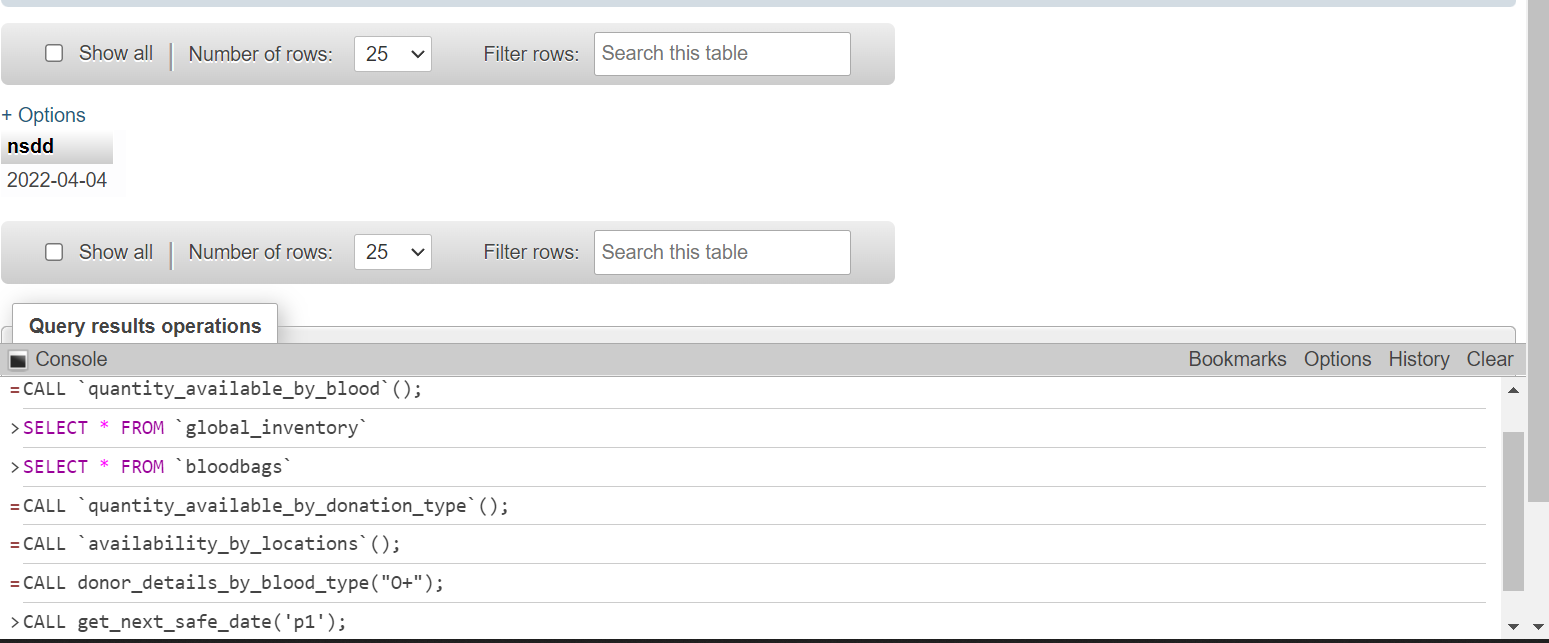
END$$

DELIMITER ;

**Explanation:**

The procedure provides the donation history of a specific donor. Here, we take pid as input and from that pid the donation history of that donor is displayed.

1. **get\_next\_safe\_date**



DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `get\_next\_safe\_date`(IN `id` CHAR(8))

BEGIN

DECLARE p\_id CHAR(8);

DECLARE bl\_type CHAR(3);

DECLARE gen CHAR(1);

DECLARE nsdd DATE;

DECLARE exit\_loop BOOLEAN DEFAULT FALSE;

DECLARE cur1 CURSOR FOR SELECT pid, nextSafeDonation FROM donor WHERE pid = id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit\_loop = TRUE;

OPEN cur1;

date\_type: LOOP

FETCH FROM cur1 INTO p\_id,nsdd;

IF exit\_loop THEN

LEAVE date\_type;

END IF;

IF p\_id=id THEN

SELECT nsdd;

END IF;

END LOOP date\_type;

CLOSE cur1;

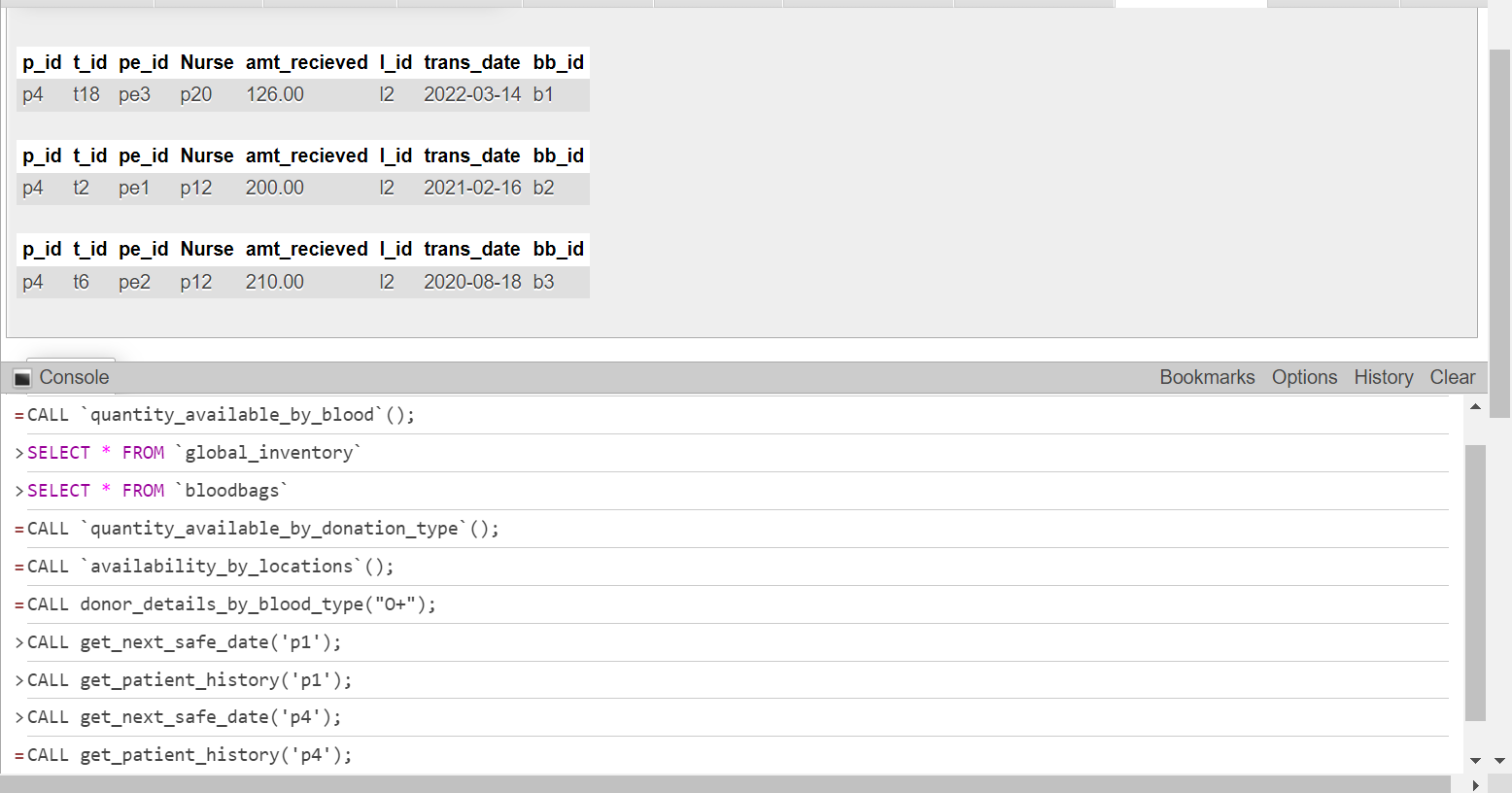
END$$

DELIMITER ;

**Explanation:**

The procedure provides details of the next safe date for donation for a specific donor. Here, we take pid as input and from that pid the next safe donation date for that donor is displayed.

1. **get\_patient\_history**

****

DELIMITER $$

CREATE DEFINER=`root`@`localhost` PROCEDURE `get\_patient\_history`(IN `id` VARCHAR(8))

BEGIN

DECLARE p\_id CHAR(8);

DECLARE t\_id CHAR(8);

DECLARE pe\_id CHAR(8);

DECLARE Nurse CHAR(8);

DECLARE amt\_recieved DECIMAL(5,2);

DECLARE l\_id CHAR(4);

DECLARE trans\_date DATE;

DECLARE bb\_id CHAR(10);

DECLARE exit\_loop BOOLEAN DEFAULT FALSE;

DECLARE cur1 CURSOR FOR SELECT pid, transfusion.tid, peid,transfusion.nurse,amount\_recieved\_CC,lid,transfusion\_date,bbid FROM transfusion INNER JOIN transfusion\_records ON transfusion.tid = transfusion\_records.tid;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit\_loop = TRUE;

OPEN cur1;

transfusion\_type:LOOP

FETCH FROM cur1 INTO p\_id,t\_id,pe\_id,Nurse,amt\_recieved,l\_id,trans\_date,bb\_id;

IF exit\_loop THEN

LEAVE transfusion\_type;

END IF;

IF p\_id=id THEN

SELECT p\_id,t\_id,pe\_id,Nurse,amt\_recieved,l\_id,trans\_date,bb\_id;

END IF;

END LOOP transfusion\_type;

CLOSE cur1;

END$$

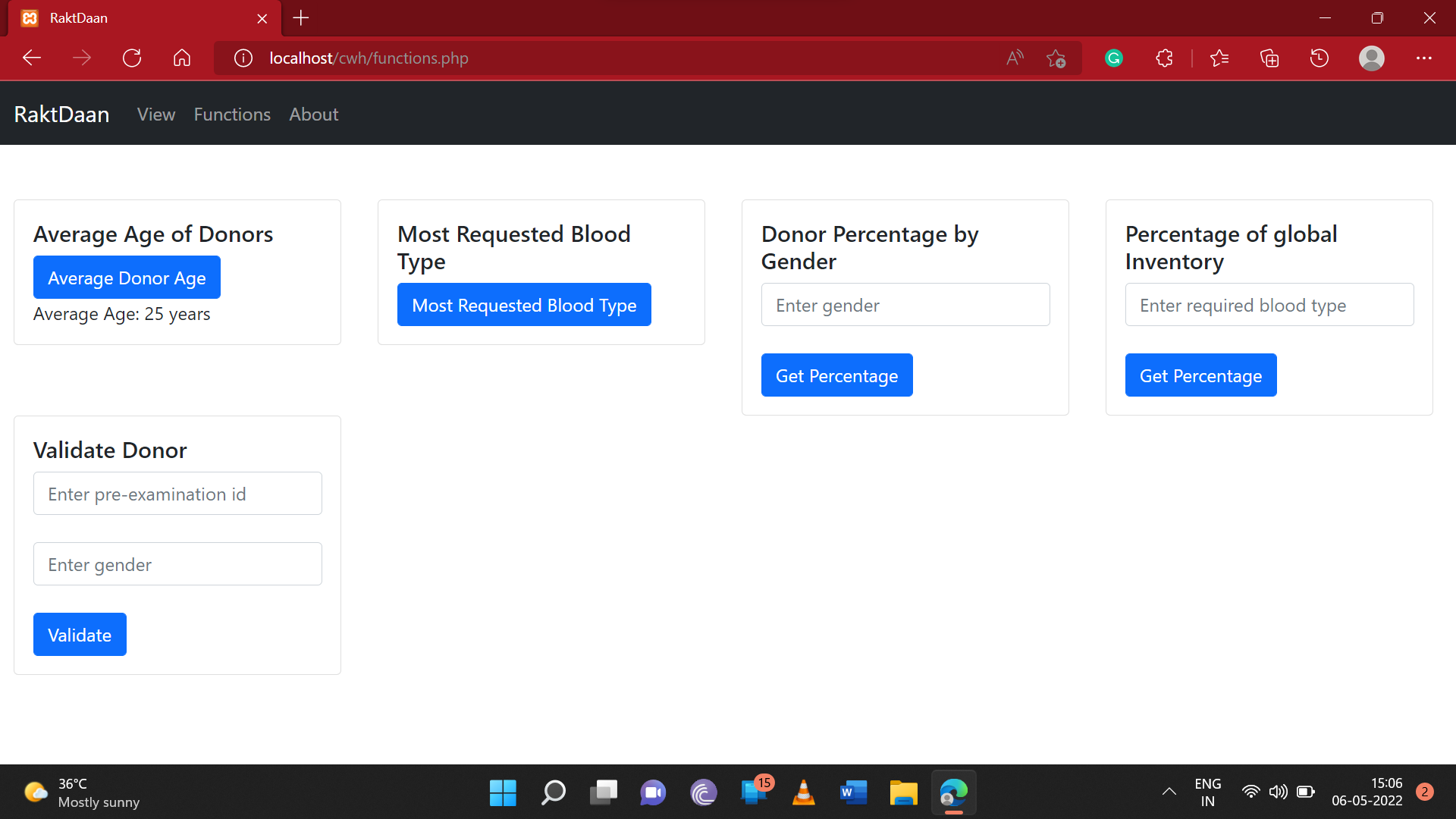
DELIMITER ;

**Explanation:**

The procedure provides the transfusion history of a specific patient. Here, we take pid as input and from that pid the transfusion history of that patient is displayed.

* **Functions**

1. **avg\_age\_donor**

****

DELIMITER $$

CREATE DEFINER=`root`@`localhost` FUNCTION `avg\_age\_donor`() RETURNS int(11)

BEGIN

DECLARE avg\_age int;

SET avg\_age = 0;

SELECT avg(persons.age) into avg\_age from donor INNER JOIN persons ON donor.pid = persons.pid;

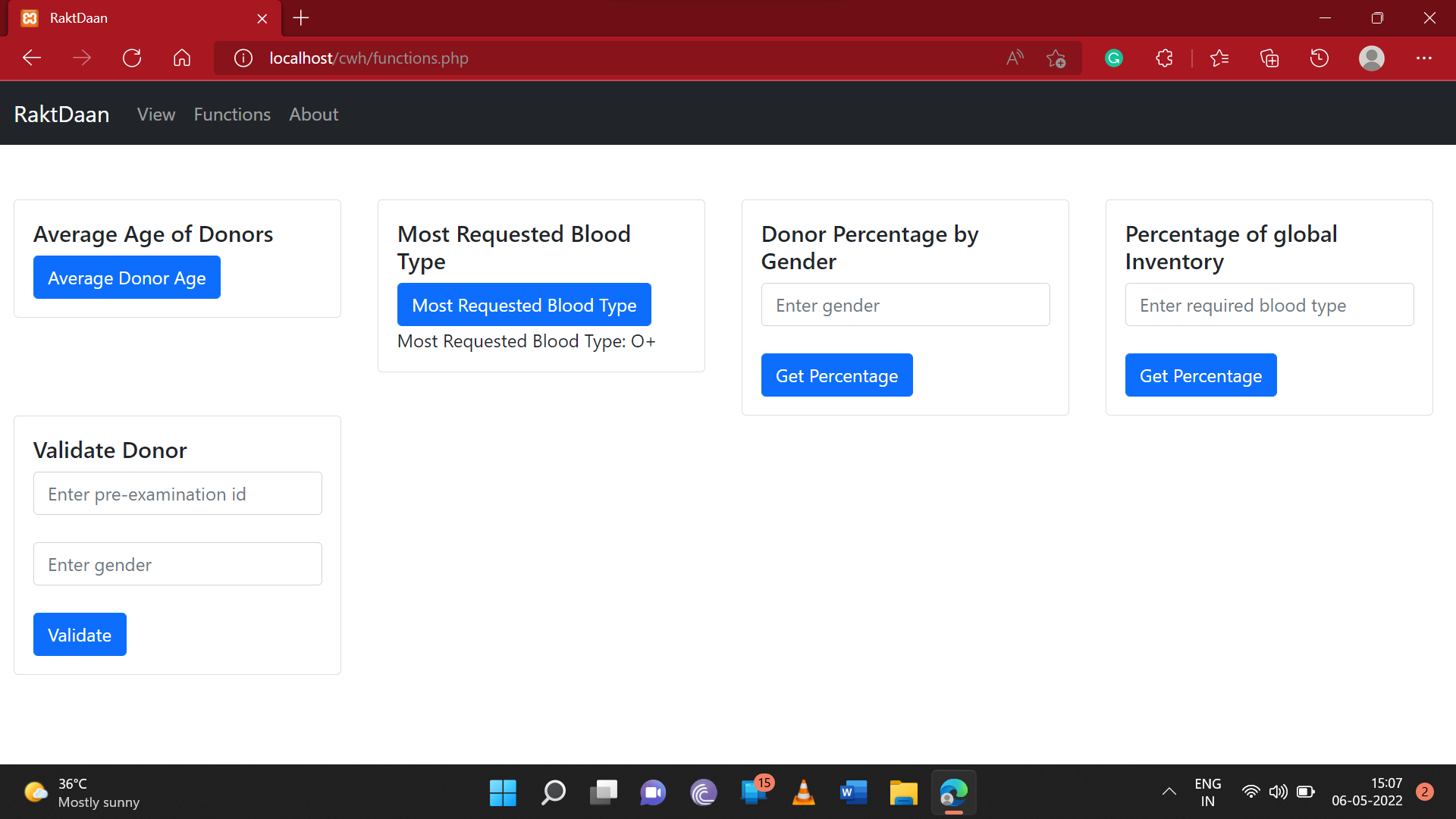
RETURN avg\_age;

END$$

DELIMITER ;

**Explanation:** This function provides the average age of the donor that came for donation.

1. **most\_requested\_bloodType**



DELIMITER $$

CREATE DEFINER=`root`@`localhost` FUNCTION `most\_requested\_bloodType`() RETURNS varchar(3) CHARSET utf8mb4

BEGIN

DECLARE b\_type TEXT;

DECLARE num INT;

SELECT blood\_type\_requested, COUNT(rqid) INTO b\_type,num FROM requests GROUP BY blood\_type\_requested DESC limit 1;

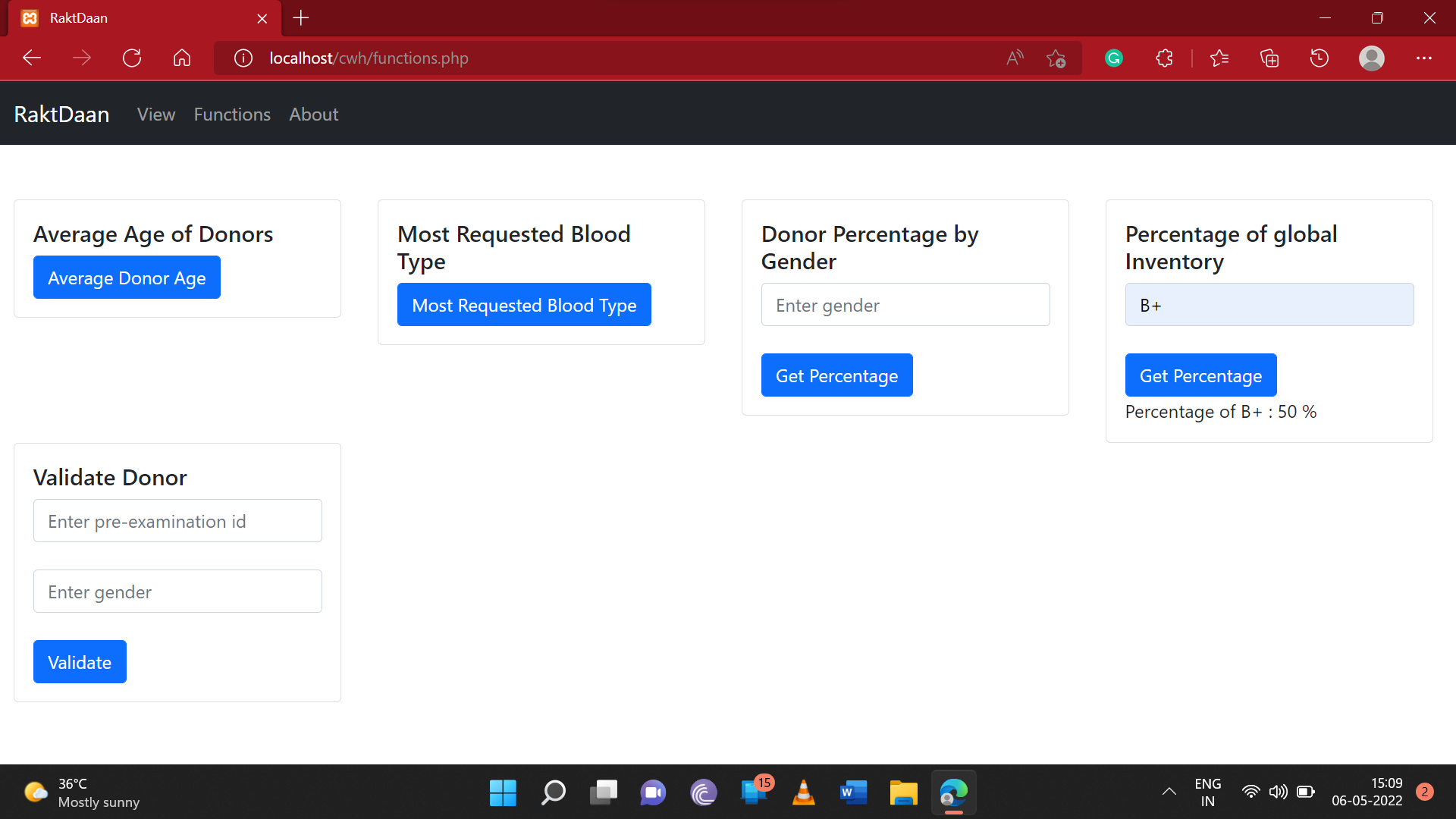
RETURN b\_type;

END$$

DELIMITER ;

**Explanation:** This function provides the most requested blood type for transfusion for patients.

1. **requested\_blood\_type\_global\_percentage**



DELIMITER $$

CREATE DEFINER=`root`@`localhost` FUNCTION `requested\_blood\_type\_global\_percentage`(`reqType` VARCHAR(3)) RETURNS float

BEGIN

DECLARE type\_c float;

DECLARE total\_c float;

DECLARE per float;

SET per = 0;

SELECT COUNT(gi.bbid) INTO type\_c FROM global\_inventory gi INNER JOIN bloodbags bb ON gi.bbid = bb.bbid WHERE bb.blood\_type = reqType

AND gi.available = TRUE;

SELECT COUNT(gi.bbid) INTO total\_c

FROM global\_inventory gi

WHERE gi.available = TRUE;

SET per = (type\_c/total\_c)\*100;

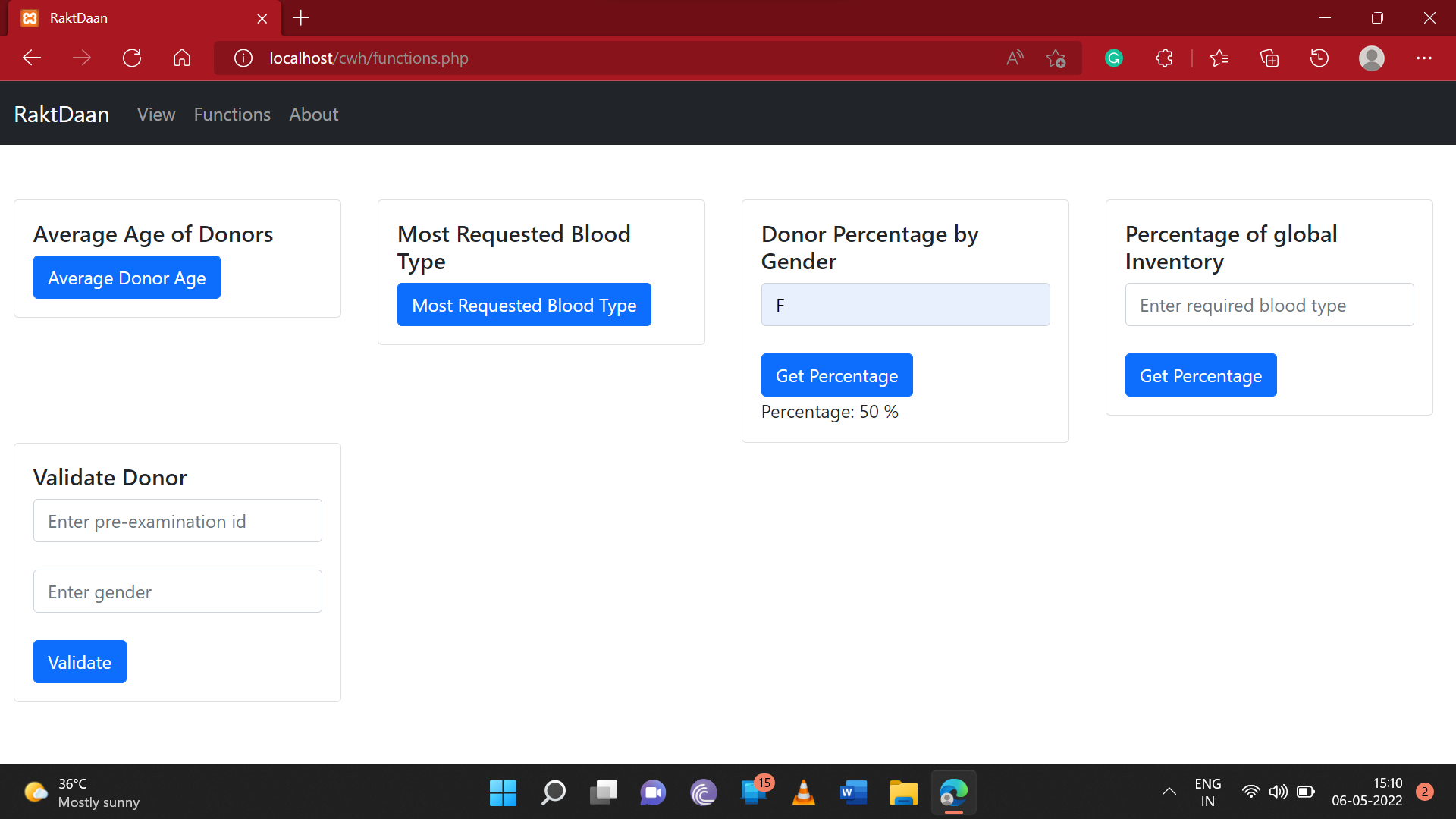
RETURN per;

END$$

DELIMITER ;

**Explanation:** This function provides the most requested blood type for transfusion for patients. Here, we take input as blood type that is requested.

1. **gender\_percentage\_of\_donor**



DELIMITER $$

CREATE DEFINER=`root`@`localhost` FUNCTION `gender\_percentage\_of\_donor`(`gen` VARCHAR(1)) RETURNS float

BEGIN

DECLARE cnt float;

DECLARE total\_cnt float;

DECLARE per float;

SET per = 0;

IF gen = "M" THEN

SELECT COUNT(pid) INTO cnt FROM donor WHERE gender = "M";

ELSE

SELECT COUNT(pid) INTO cnt FROM donor WHERE gender = "F";

END IF;

SELECT COUNT(pid) INTO total\_cnt FROM donor;

SET per = (cnt/total\_cnt)\*100;

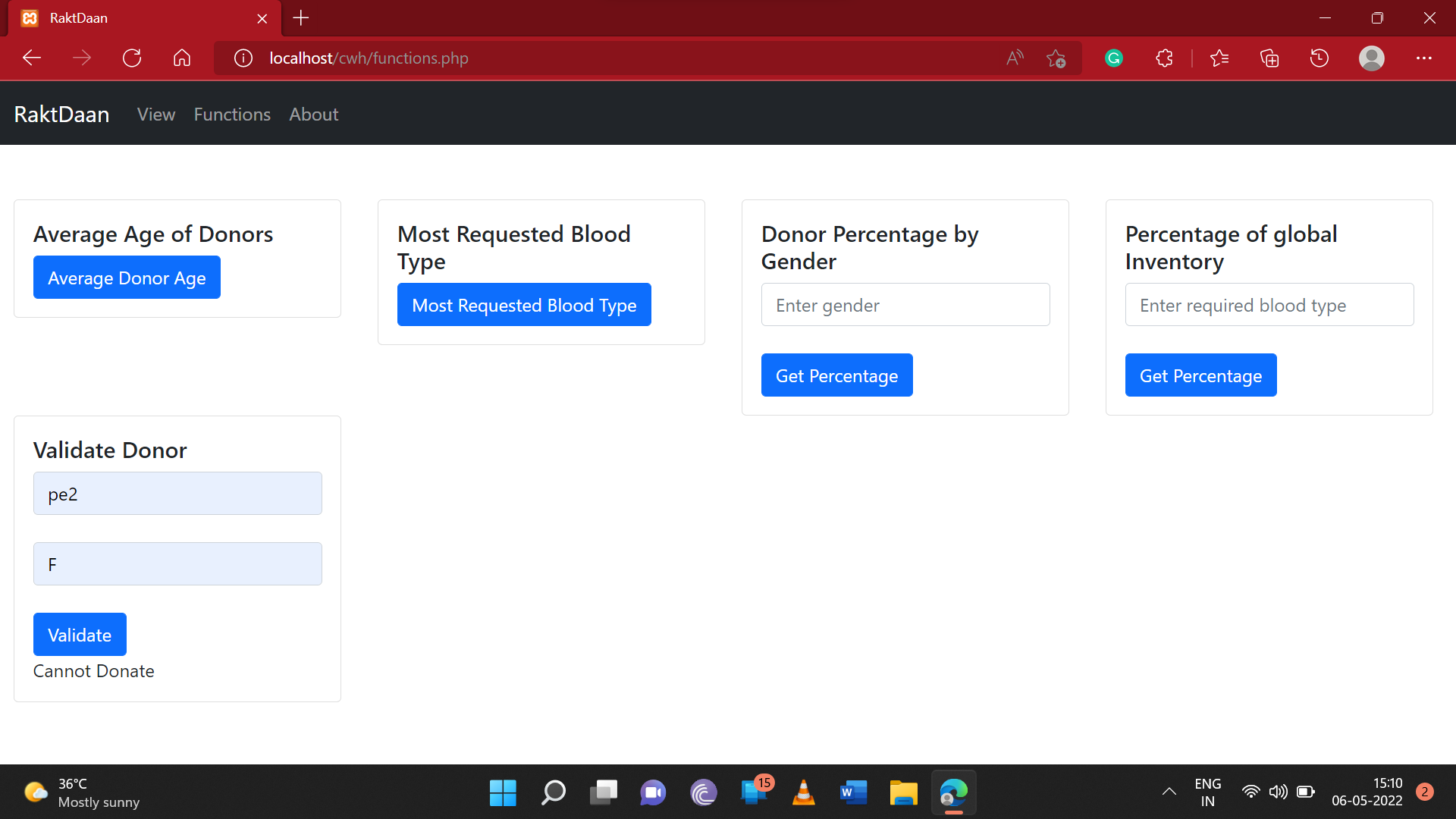
RETURN per;

END$$

DELIMITER ;

**Explanation:** This function provides the percentage of male and female donors available for donation. Here, we take gender as input.

1. **validate\_donation**



DELIMITER $$

CREATE DEFINER=`root`@`localhost` FUNCTION `validate\_donation`(`pe\_id` VARCHAR(8), `gen` VARCHAR(1)) RETURNS tinyint(1)

BEGIN

DECLARE hemgdl DECIMAL(5,2);

DECLARE temp\_F DECIMAL(5,2);

DECLARE p\_rate INT;

DECLARE canDonate BOOLEAN;

SET canDonate=FALSE;

SELECT hemoglobin\_gDL, temperature\_F, pulse\_rate\_BPM INTO hemgdl, temp\_F, p\_rate FROM pre\_exam WHERE peid=pe\_id;

IF(gen="F" AND hemgdl>=12.5 AND temp\_F>=96.4 AND temp\_F<=100 AND p\_rate>=50 AND p\_rate<=100) THEN

SET canDonate=TRUE;

ELSEIF(gen="M" AND hemgdl>=13.0 AND temp\_F>=96.4 AND temp\_F<=100 AND p\_rate>=50 AND p\_rate<=100) THEN

SET canDonate=TRUE;

END IF;

RETURN canDonate;

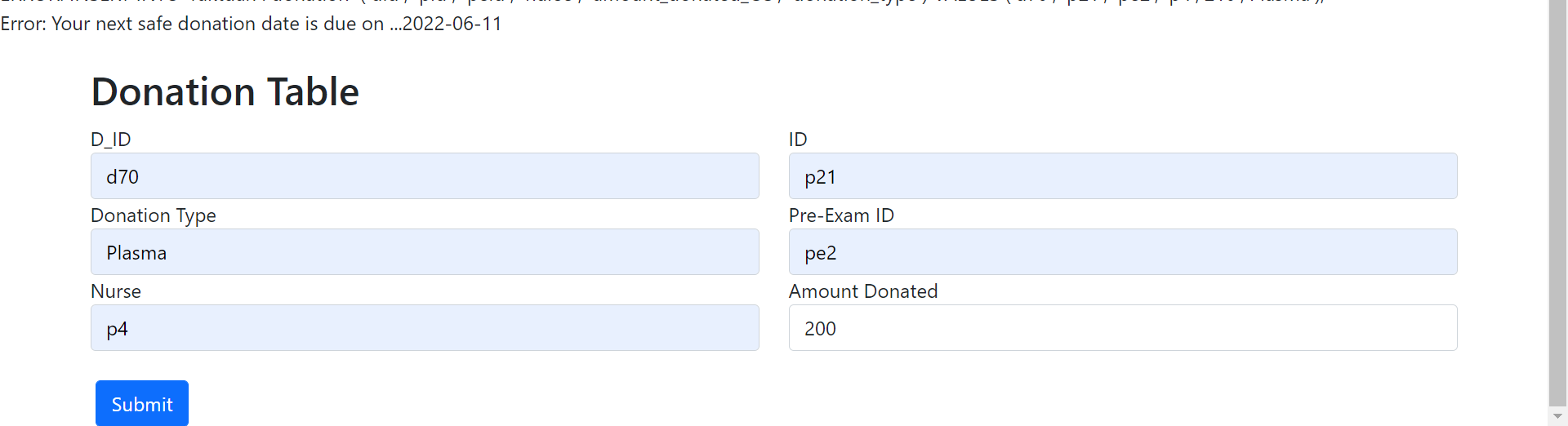
END$$

DELIMITER ;

**Explanation:** This function provides whether a donor can donate blood or not based on Hemoglobin, Temperature and Pulse Rate. Here, we take input as peid and gender as hemoglobin range is different for male and female.

* **Triggers**

1. **check\_date\_before\_donation**



CREATE TRIGGER `check\_date\_before\_donation` BEFORE INSERT ON `donation`

FOR EACH ROW begin

DECLARE msg VARCHAR(128);

DECLARE cdate date;

SELECT nextSafeDonation into cdate from donor where donor.pid = new.pid;

if (sysdate() < cdate) then

set msg = concat('Error: Your next safe donation date is due on ...',cdate);

signal sqlstate '45001' set message\_text = msg;

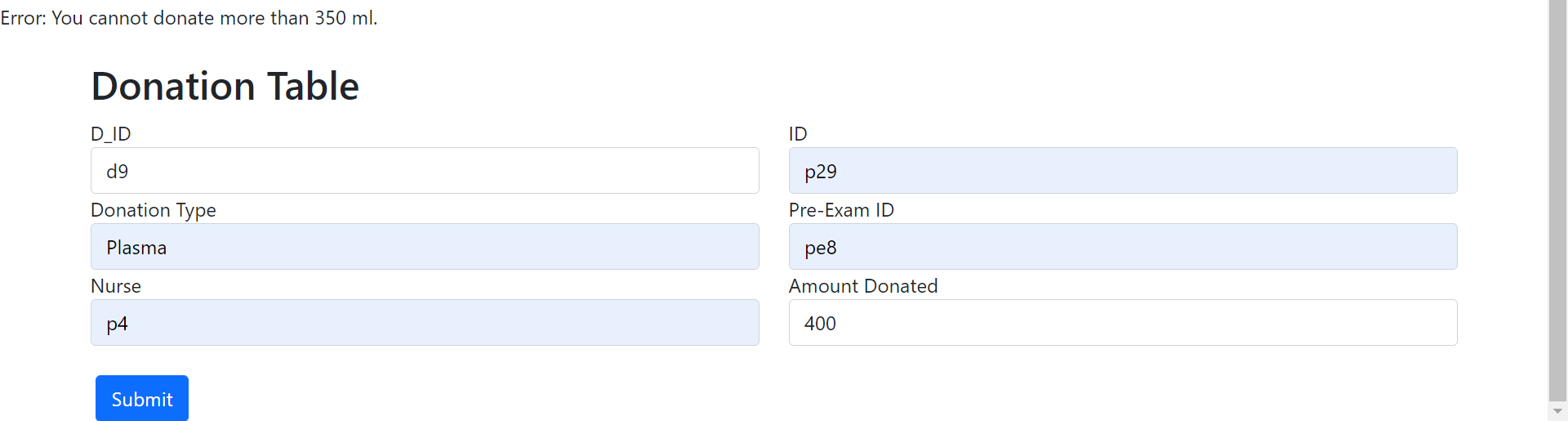
end if;

end

**Explanation:**

The Trigger that validates whether the donation date is after the nextSafeDonation date before inserting a record on the donation table.

1. **check\_donation\_amount**

****

CREATE TRIGGER `check\_donation\_amount` BEFORE INSERT ON `donation`

FOR EACH ROW begin

declare msg varchar(128);

if (new.amount\_donated\_CC > 350) then

set msg = concat('Error: You cannot donate more than 350 ml.');

signal sqlstate '45001' set message\_text = msg;

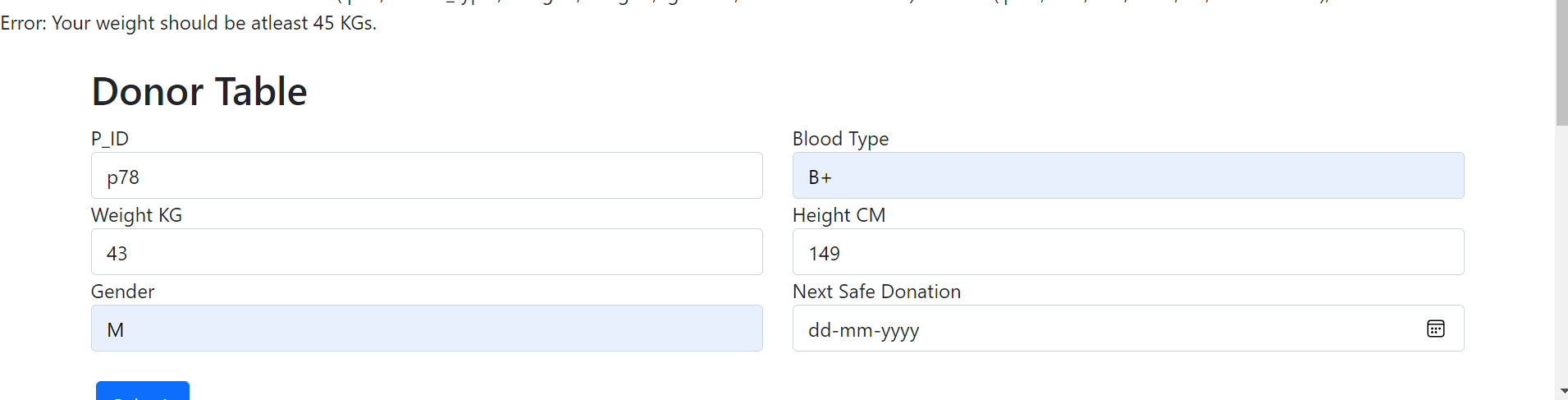
end if;

end

**Explanation:**

The Trigger that validates whether the donation amount is less than or equal to 350 ml before inserting a record on the donation table.

1. **check\_weight\_donor**

****

CREATE TRIGGER `check\_weight\_donor` BEFORE INSERT ON `donor`

FOR EACH ROW begin

declare msg varchar(128);

if (new.weight < 45) then

set msg = concat('Error: Your weight should be atleast 45 KGs.');

signal sqlstate '45001' set message\_text = msg;

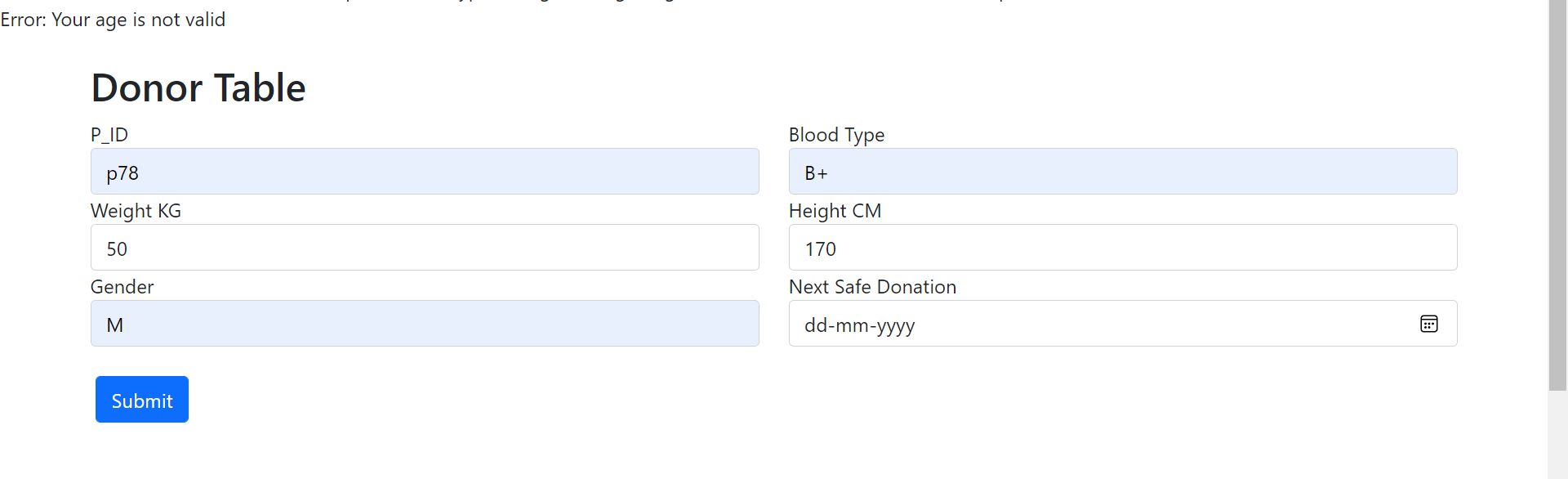
end if;

end

**Explanation:**

The Trigger that validates whether the donor weight is greater than or equal to 45 KGs. before inserting a record on the donor table.

1. **check\_donor\_age**

****

CREATE TRIGGER `check\_donor\_age` BEFORE INSERT ON `donor`

FOR EACH ROW begin

declare msg varchar(128);

DECLARE myage integer;

select age into myage from persons where pid=new.pid;

if (myage < 18 OR myage > 65)

then set msg = concat('Error: Your age is not valid');

signal sqlstate '45001' set message\_text = msg;

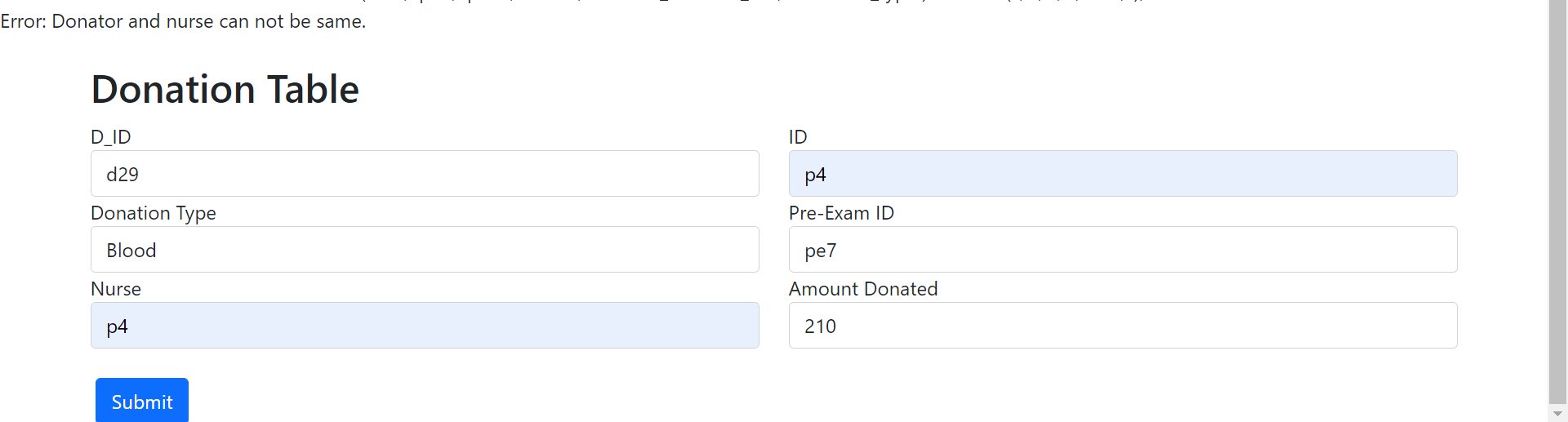
end if;

end

**Explanation:**

The Trigger that validates whether the donor age is in the range of 18 to 65 years with the help of fetching age data from the persons table before inserting a record on the donor table.

1. **check\_nurse\_before\_Donation**

****

CREATE TRIGGER `check\_nurse\_before\_Donation` BEFORE INSERT ON `donation`

FOR EACH ROW begin

declare c\_pid int;

declare msg varchar(128);

if new.pid = new.nurse THEN

set msg = concat('Error: Donator and nurse can not be same.');

signal sqlstate '45001' set message\_text = msg;

end if;

End

**Explanation:**

The Trigger that validates whether the donor and the nurse is the same person or not before inserting a record on the donation table.

1. **before\_delete\_on\_persons**

CREATE TRIGGER `before\_delete\_on\_persons` BEFORE DELETE ON `persons`

FOR EACH ROW BEGIN

delete from nurse where pid = old.pid;

delete from patient where pid = old.pid;

delete from donor where pid = old.pid;

end

**Explanation:**

The Trigger that deletes the records of nurse, patient and donor whenever persons record is deleted in the persons table corresponding to the given pid.

1. **check\_nurse\_before\_Transfusion**

CREATE TRIGGER `check\_nurse\_before\_Transfusion` BEFORE INSERT ON `transfusion`

FOR EACH ROW begin

declare c\_pid int;

declare msg varchar(128);

if new.pid = new.nurse THEN

set msg = concat('Error: Patient and nurse can not be same.');

signal sqlstate '45002' set message\_text = msg;

end if;

end

**Explanation:**

The Trigger that validates whether the patient and the nurse is the same person or not before inserting a record on the transfusion table works same as previous trigger.

1. **update\_next\_donation\_date**

CREATE TRIGGER `update\_next\_donation\_date` BEFORE INSERT ON `donation\_records`

FOR EACH ROW BEGIN

DECLARE wait\_d int;

DECLARE d\_date DATE;

DECLARE u\_did char(8);

SET d\_date = NEW.donation\_date;

SET u\_did = NEW.did;

SELECT donation\_types.frequency\_days INTO wait\_d FROM donation INNER JOIN donation\_types ON donation.donation\_type = donation\_types.type WHERE donation.did = u\_did;

UPDATE donor SET nextSafeDonation = DATE\_ADD(NEW.donation\_date , INTERVAL 56 DAY) WHERE donor.pid in (SELECT donor.pid

FROM donation INNER JOIN donation\_records ON donation.did = donation\_records.did

INNER JOIN donor ON donor.pid = donation.pid

WHERE donation.did = u\_did);

END

**Explanation:**

When a record is being inserted in donation\_records it takes the donation date and donation type into account and updates the next safe donation date in the donor table.

1. **update\_status**

CREATE TRIGGER `update\_status` BEFORE INSERT ON `transfusion\_records`

FOR EACH ROW BEGIN

UPDATE global\_inventory SET available = FALSE WHERE bbid = new.bbid;

END

**Explanation:**

When a blood bag is used in transfusion this trigger makes its availability false in global\_inventory table.

1. **before\_delete\_t\_records**

CREATE TRIGGER `before\_delete\_t\_records` BEFORE DELETE ON `transfusion\_records`

FOR EACH ROW BEGIN

delete from transfusion where tid = old.tid;

end

**Explanation:**

The Trigger that deletes the records of the transfusion table whenever a transfusion record is deleted in the transfusion\_records table corresponding to the given tid.

1. **before\_delete\_d\_records**

CREATE TRIGGER `before\_delete\_d\_records` BEFORE DELETE ON `donation\_records`

FOR EACH ROW BEGIN

delete from donation where did = old.did;

end

**Explanation:**

The Trigger that deletes the records of the donation table whenever a donation record is deleted in the donation\_records table corresponding to the given did.

THANK YOU