DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

PES2201800511 **Sunad Suhas**

BLOOD BANK MANAGEMENTSYSTEM

Transfusion of blood and blood components is an established standard way of treating patients who are deficient in one or more blood constituents and is therefore an essential part of health care. A blood transfusion service is a complex organization requiring careful design and management.

Essential functions of blood transfusion service are donor requirement, blood collection, testing of donor blood, preparation and supply of these components to the patients.

This Database is designed to record every donor or patient who donates and receives the blood.

This Database keeps track of the amount of blood stored in the stock inventory as and when a donation or transfusion takes place. In other words the stock inventory automatically updates the data when blood is donated by the donor and when blood is received by the patient. This automatic change happens via a trigger which I will be describing below.

The report includes all the required information regarding this database such as the tables used, attributes of the tables, All the keys, Functional dependencies, Normalization, violations of Normalization, triggers, DDL , complex queries, conclusion.

CONTENTS

INTRODUCTION

Mini world description of the blood bank management system

- Every person who comes to the blood bank will have to register by giving personal information such as Name, age, weight, blood group . After the registration, each person will be associated with an ID (PID).
- A person can register himself as donor or patient. The differentiation is done by allotting a D(donor) or P(patient/Recipient) to the person respectively.
- Whenever a donation happens, the blood given by the donor is recorded in the stock inventory which is automatically updated. In other words if a Donor gives O+ blood , then the total amount of O+ blood is automatically added in the stock inventory. The same happens with transfusion.
- The records of each donor and patient is recorded in the Donation Records and Transfusion Records tables.
- The Total amount of blood (of all blood groups) is recorded in the Inventory.

EACH ENTITY AND ITS ATTRIBUTES

- **Persons**

  PID (PK), name, gender, age, Blood_group, weight, DP

- **Donors**

  PID(PK, FK references Persons), Blood_group, constraint FK PID cascades on delete and update

- **Patients**

  PID(PK, FK references Persons), Blood_group, constraint FK PID cascades on delete and update

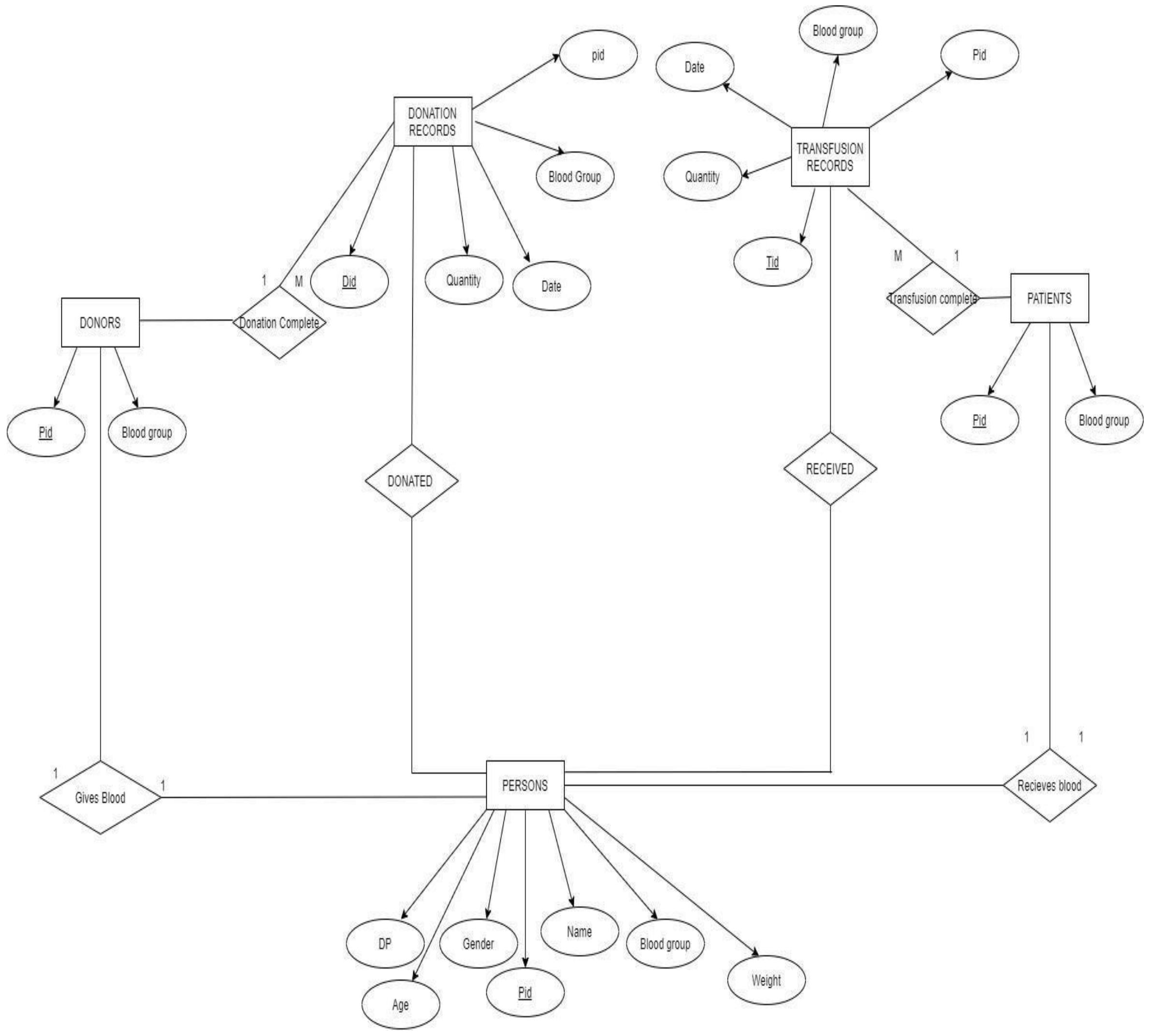
- **Transfusion_Records**

  TID(PK), PID(PK)(FK references Persons), Blood_group, Quantity, Date, constraint FK PID cascades on update and sets default ’deleted’ on delete

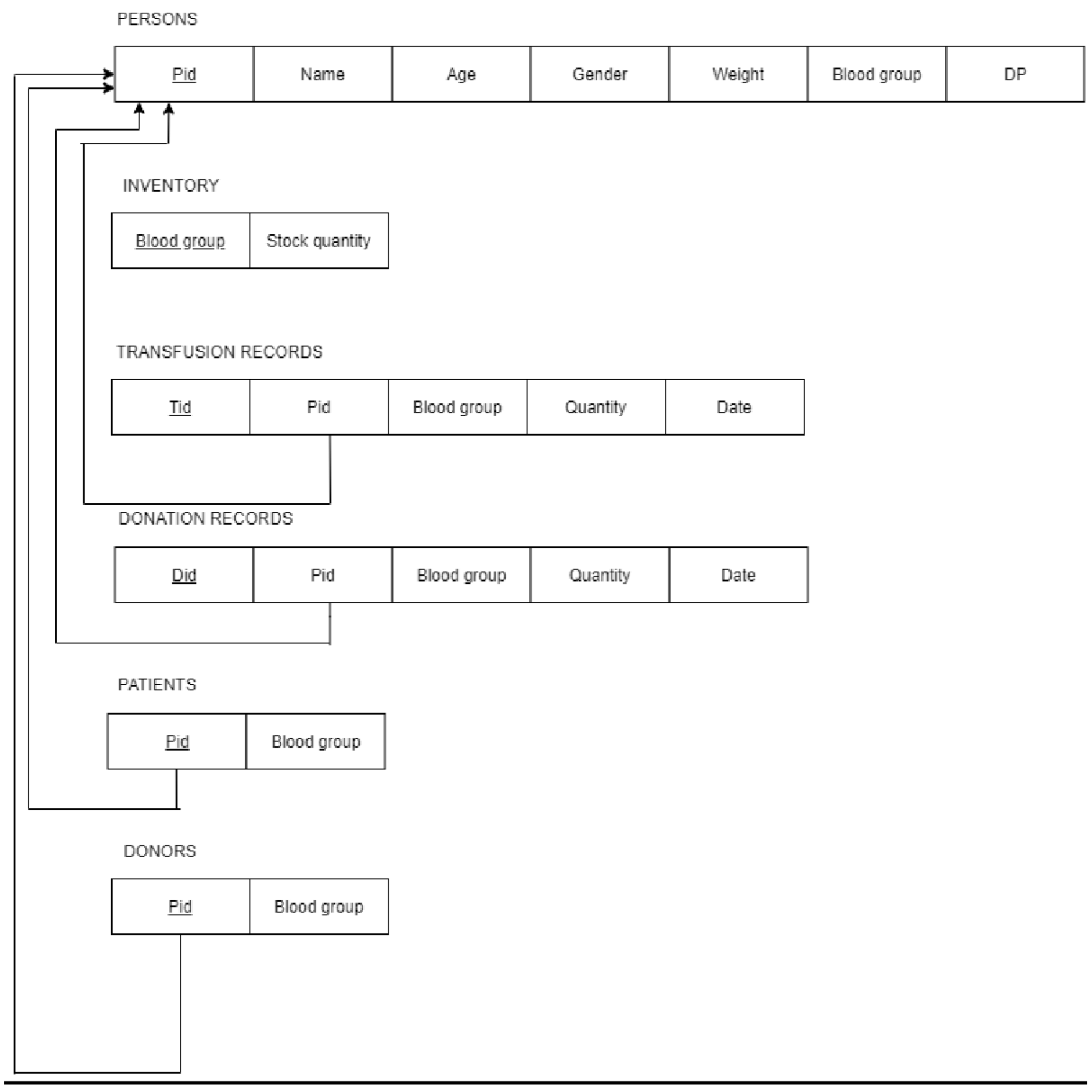
- **Donation_Records**

  DID(PK), PID(PK)(FK references Persons), Blood_group, Quantity, Date, constraint FK PID cascades on update and sets default ’deleted’ on delete

DATA MODEL


DONATION RECORDS
pid
Blood Group
Did
Quantity
Date
TRANSFUSION RECORDS
Date
Blood group
Pid
Quantity
Tid
DONORS
Pid
Blood group
1
M
Donation Complete
M
1
Transfusion complete
PATIENTS
Pid
Blood group
DONATED
RECEIVED
1
1
Gives Blood
PERSONS
1
1
Recieves blood
DP
Age
Gender
Pid
Name
Blood group
Weight

RELATIONAL SCHEMA DIAGRAM



TABLES USED

- Persons (Identified by PID, with name, gender, age, Blood_group, weight, Donor/Patient indicator as attributes)
  PK - PID

- Donors (Identified by PID, with Blood_group as an attribute)
  PK - PID
  FK - PID

- Patients (Identified by PID, with Blood_group as an attribute)
  PK - PID
  FK - PID

- Transfusion_Records (Identified by TID and PID with Blood_group Quantity, Date as attributes)
  PK - TID, PID
  FK - PID

- Donation_Records (Identified by DID and PID with Quantity, Blood_group, Date as attributes)
  PK – DID, PID
  FK - PID

- Inventory (Identified by Blood_group, with Stock_quantity as an attribute).
  PK – Blood group

NOTE :

- **Donor List should contain requests that are insertedinto Donation Records only if they satisfy the following constraints:**
  Donor age is above 18 and weight is above 45kg.

- **Records should be retained even if persons are deleted from the persons list.**

- **The stock quantity should update its value after every transfusion or donation to reflect an active database with up-to-date information.**

FUNCTIONAL DEPENDECIES AND NORMALIZATION

Functional Dependencies -

- Persons.PID -> name, gender, age, weight, Blood_group, dp
- Donors.PID -> Blood_group
- Patients.PID -> Blood_group
- Transfusions_Records.(TID,PID) -> Blood_group, Quantity, Date
- Transfusions_Records.TID ->PID, Blood_group, Quantity, Date
- Donations_Records.DID ->PID, Blood_group, Quantity, Date
- Donations_Records.(DID,PID) -> Blood_group, Quantity, Date
- Inventory.Blood_group -> Stock_quantity

These are the Candidate keys-

- Persons.PID - PK
- Donors.PID – PK
- Patients.PID – PK
- Transfusion_Records.(TID,PID) (both are candidate keys) - PK
- Donation_Records.(DID,PID) (both are candidate keys) - PK
- Inventory.Blood_group – PK

NORMALIZATION

- As we can see, all tables are normalized till 3NF.
- Each table has one or two candidate keys.
- None of the tables hold multiple values, and holds only atomic values. This proves that all the tables are 1F normalized
- None of the non- prime attributes of the table is dependant on the proper subset of any candidate key of the table. This proves that all the tables are 2NF normalized.
- There are no Transitive functional dependency of non – prime attributes on any super key and hence this proves that all the table are 3NF normalized.

DISCUSSION ON WHICH NORMAL FORM WILL BE VIOLATED

- If any table above has more than one value in a cell, then 1 NF is violated. Let's say in Donation_record table , a donor donated blood twice, and hence we write both the quantities of blood donated in the same cell. This would violate 1NF.

- Transfusion_Records.(TID,PID) – If TID alone was candidate key, multiple transfusions by a single patient would cause transitive dependency and violate 3NF.

- Donation_Records.(DID,PID) – If DID alone was candidate key, multiple transfusions by a single patient would cause transitive dependency and violate 3NF.

- Table is in 3NF, 2NF may be violated in Transfusion_Records or Donation_Records, as although the primary keys is (DID, PID), DID can uniquely identify the tuples.

DDL

These are the create table commands

- CREATE TABLE bloodbank.persons(pid char(8) not null unique, name text not null, age integer not null, gender char(1) not null, weight integer not null, Blood_group char(3) not null, DP char(1) not null, CONSTRAINT check_gender CHECK(gender='M' OR gender='F'),CONSTRAINT check_dp CHECK(DP='D' OR DP='P'), primary key (pid));

- CREATE TABLE bloodbank.donors(pid char(8) not null, Blood_group char(3) not null, primary key(pid), CONSTRAINT dpidfk FOREIGN KEY(pid) REFERENCES persons(pid) ON DELETE CASCADE ON UPDATE CASCADE);

- CREATE TABLE bloodbank.patients(pid char(8) not null, Blood_group char(3) not null, primary key(pid), CONSTRAINT ppidfk FOREIGN KEY(pid) REFERENCES persons(pid) ON DELETE CASCADE ON UPDATE CASCADE);

- CREATE TABLE bloodbank.Donation_Records(did char(8) not null unique, pid char(8) not null unique DEFAULT('deleted_person'), Blood_group char(3) not null, quantity integer not null, date DATE not null, primary key(did,pid),CONSTRAINT drpidfk FOREIGN KEY(pid) REFERENCES persons(pid) ON DELETE SET DEFAULT ON UPDATE CASCADE);

- CREATE TABLE bloodbank.Transfusion_Records(tid char(8) not null unique, pid char(8) not null unique DEFAULT('deleted_person'), Blood_group char(3) not null, quantity integer not null, date DATE not null, primary key(tid,pid), CONSTRAINT trpidfk FOREIGN KEY(pid) REFERENCES persons(pid) ON DELETE SET DEFAULT ON UPDATE CASCADE);

- CREATE TABLE bloodbank.Inventory(Blood_group char(3) not null unique, Stock_quantity integer not null, primary key(Blood_group));

TESTING FOR LOSSLESS JOIN PROPERTY

To check for lossless join decomposition using FD set, following conditions must hold:

1. Union of Attributes of R1 and R2 must be equal to attribute of R. Each attribute of R must be either in R1 or in R2.

   Att(R1) U Att(R2) = Att(R)

2. Intersection of Attributes of R1 and R2 must not be NULL.

   Att(R1) ∩ Att(R2) ≠ Φ

3. Common attribute must be a key for at least one relation (R1 or R2)

   Att(R1) ∩ Att(R2) -> Att(R1) or Att(R1) ∩ Att(R2) -> Att(R2)

We tried to decompose the Persons table into two parts, one Table with Pid and Blood group, and the other table with the rest of the attributes of the same table. So, let's check the lossless join property for the Persons table.
R1(pid, Name, Age, Gender, Weight, Blood Group, DP)
R2(Pid, Blood Group).

1) Since, the union of both the tables has the same relations as there are in the Persons Table itself, hence this condition gets satisfied.

2) Since, both the table attributes intersection is not Null, hence this condition gets satisfied too.

3) The common attributes must be a key for at least one relation, so as we can see the Pid functionally determines Blood group, and since it has been given already, this condition holds true too.

Hence, we can say that the Lossless Join Property has been successfully verified.

CONSTRAINTS

- The user should enter ‘M’ or ‘F’ in the gender columns as male or female. Entering any other entry will result in an error.
- The user should enter ‘D’ or ‘P’ in the DP column as donor or patient. Entering any other entry will result in an error.
- I have also used constraints in the other tables namely Donors, patients , Donation_records and Transfusion_records where pid is foreign key . Whenever pid (which is primary key in the persons table) is updated in the persons table, the same should happen in the other tables which has pid as the foreign key and whenever it is deleted, the default value should be set in the other tables.

TRIGGERS

There are 4 Triggers used in this database-

- The First trigger is used to update the Blood bank Inventory as and when a donor donates blood. In other words, once a donor is entered into the donation record, then automatically the total amount of blood donated by the donor is updated in the inventory. For example – Let’s say there is 500CC of O+ blood group blood in the stock inventory. As soon as the donor gives 200 cc of O+ blood group, then the stock inventory automatically updates to 700CC of O+. The same happens with all the other blood groups.

  This command is for After Update-

```
DELIMITER $$
CREATE TRIGGER bloodbank.Inventory_Increase
AFTER UPDATE ON Donation_Records
FOR EACH ROW
```

```
BEGIN
IF NEW.Quantity is NOT NULL THEN
UPDATE Inventory
SET Stock_quantity = Stock_quantity + NEW.Quantity
WHERE NEW.Blood_Group = Inventory.Blood_group;
END IF;
END$$
DELIMITER;
```

This command is for after insert-

```
DELIMITER$$
CREATE TRIGGER bloodbank.Inventory_Increase1
AFTER INSERT ON Donation_Records
FOR EACH ROW
BEGIN
IF NEW.Quantity is NOT NULL THEN
UPDATE Inventory
SET Stock_quantity = Stock_quantity + NEW.Quantity
WHERE NEW.Blood_Group = Inventory.Blood_group;
END IF;
END$$
DELIMITER;
```

- The second trigger is used to update the Blood bank Inventory as and when a patient (recipient) receives blood. In other words, once a patient is entered into the Transfusion record, then automatically the total amount of blood received by the patient is updated in the inventory.
  For example- Let’s say there is 500CC of O+ blood group blood in the stock inventory. As soon as the patient receives 200 cc of O+ blood group, then the stock inventory automatically updates to 300CC of O+ as the blood is used from the inventory . The same happens with all the other blood groups.

This command is for before update-

```
DELIMITER $$
CREATE TRIGGER bloodbank.Inventory_Decrease
BEFORE UPDATE ON Transfusion_Records
FOR EACH ROW
BEGIN
IF NEW.Quantity is NOT NULL THEN
UPDATE Inventory
SET Stock_quantity = Stock_quantity - NEW.Quantity
WHERE NEW.Blood_Group = Inventory.Blood_group;
END IF;
END$$
DELIMITER;
```

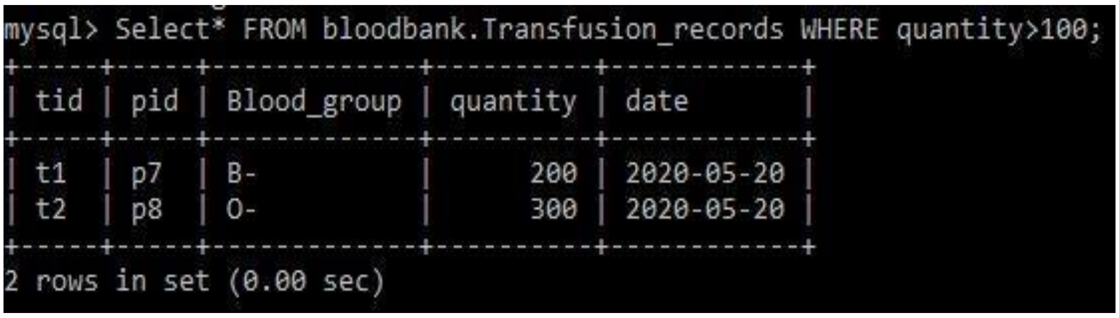
This command is for before insert-

```
DELIMITER $$
CREATE TRIGGER bloodbank.Inventory_Decrease1
BEFORE INSERT ON Transfusion_Records
FOR EACH ROW
BEGIN
IF NEW.Quantity is NOT NULL THEN
UPDATE Inventory
SET Stock_quantity = Stock_quantity - NEW.Quantity
WHERE NEW.Blood_Group = Inventory.Blood_group;
END IF;
END$$
DELIMITER;
```

SQL Queries (SIMPLE AND COMPLEX QUERIES)

- **This query choses the patients from the transfusion records who have received more than 100CC of blood -**

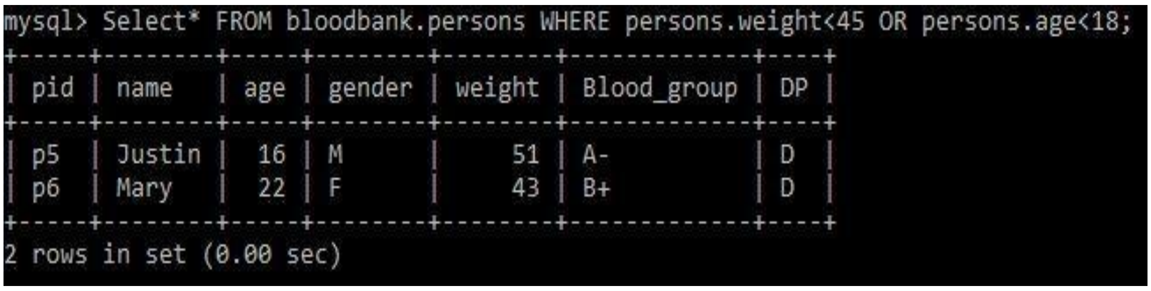
  Select* FROM bloodbank.Transfusion_records WHERE quantity>100;



```
mysql> Select* FROM bloodbank.Transfusion_records WHERE quantity>100;
+-----+-----+-------------+----------+------------+
| tid | pid | Blood_group | quantity | date       |
+-----+-----+-------------+----------+------------+
| t1  | p7  | B-          |      200 | 2020-05-20 |
| t2  | p8  | O-          |      300 | 2020-05-20 |
+-----+-----+-------------+----------+------------+
2 rows in set (0.00 sec)
```

- **This query chooses the people from the persons table whose weight is below 45kg and age is below 18 as these people are not eligible to donate blood -**

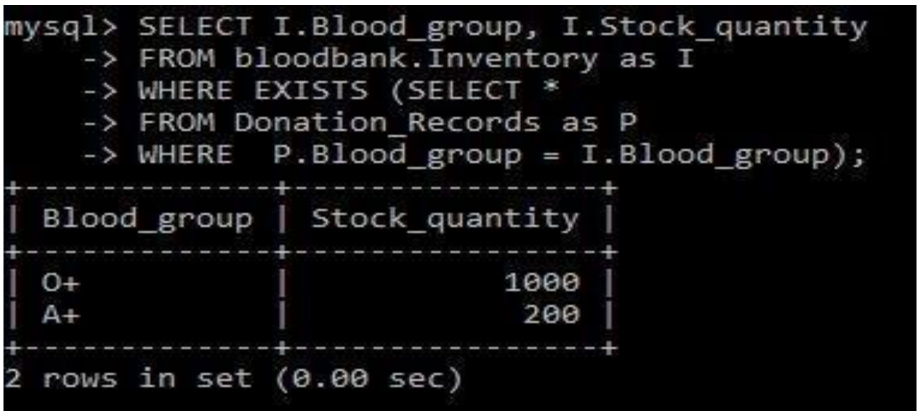
  Select* FROM bloodbank.persons WHERE persons.weight<45 OR persons.age<18;



```
mysql> Select* FROM bloodbank.persons WHERE persons.weight<45 OR persons.age<18;
+-----+--------+-----+--------+--------+-------------+----+
| pid | name   | age | gender | weight | Blood_group | DP |
+-----+--------+-----+--------+--------+-------------+----+
| p5  | Justin |  16 | M      |     51 | A-          | D  |
| p6  | Mary   |  22 | F      |     43 | B+          | D  |
+-----+--------+-----+--------+--------+-------------+----+
2 rows in set (0.00 sec)
```

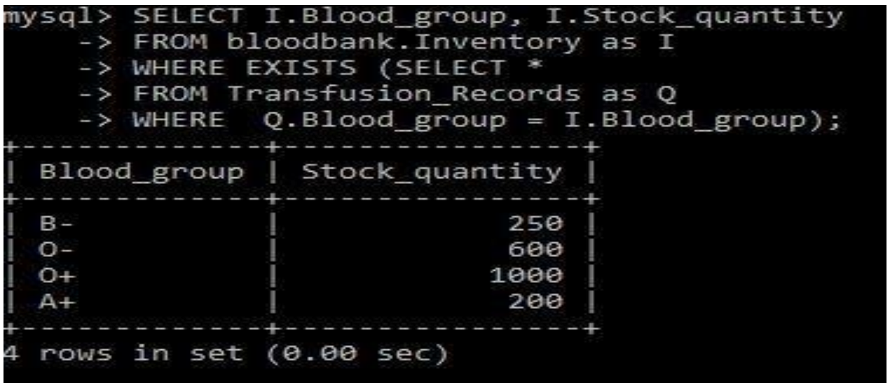
- **This query helps in showing which all donors donated the blood of which blood group along with the stock quantity available in the inventory -**

```
SELECT I.Blood_group, I.Stock_quantity
FROM bloodbank.Inventory as I
WHERE EXISTS (SELECT *
                    FROM Donation_Records as P
                    WHERE P.Blood_group = I.Blood_group);
```



```
mysql> SELECT I.Blood_group, I.Stock_quantity
    -> FROM bloodbank.Inventory as I
    -> WHERE EXISTS (SELECT *
    -> FROM Donation_Records as P
    -> WHERE  P.Blood_group = I.Blood_group);
+-------------+----------------+
| Blood_group | Stock_quantity |
+-------------+----------------+
| O+          |           1000 |
| A+          |            200 |
+-------------+----------------+
2 rows in set (0.00 sec)
```

- **This query helps in showing which all patients (recipients) received the blood of which blood group along with the stock quantity available in the inventory –**

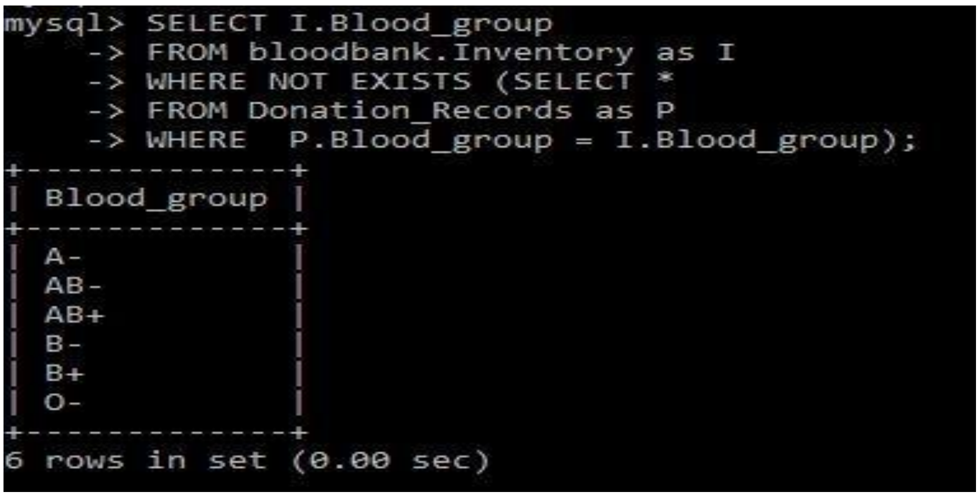


```
mysql> SELECT I.Blood_group, I.Stock_quantity
    -> FROM bloodbank.Inventory as I
    -> WHERE EXISTS (SELECT *
    -> FROM Transfusion_Records as Q
    -> WHERE  Q.Blood_group = I.Blood_group);
+-------------+----------------+
| Blood_group | Stock_quantity |
+-------------+----------------+
| B-          |            250 |
| O-          |            600 |
| O+          |           1000 |
| A+          |            200 |
+-------------+----------------+
4 rows in set (0.00 sec)
```

```
SELECT I.Blood_group, I.Stock_quantity
FROM bloodbank.Inventory as I
WHERE EXISTS (SELECT *
                        FROM Transfusion_Records as Q
                        WHERE Q.Blood_group = I.Blood_group);
```

- **This query helps in showing which blood group has not been donated by the donors. -**

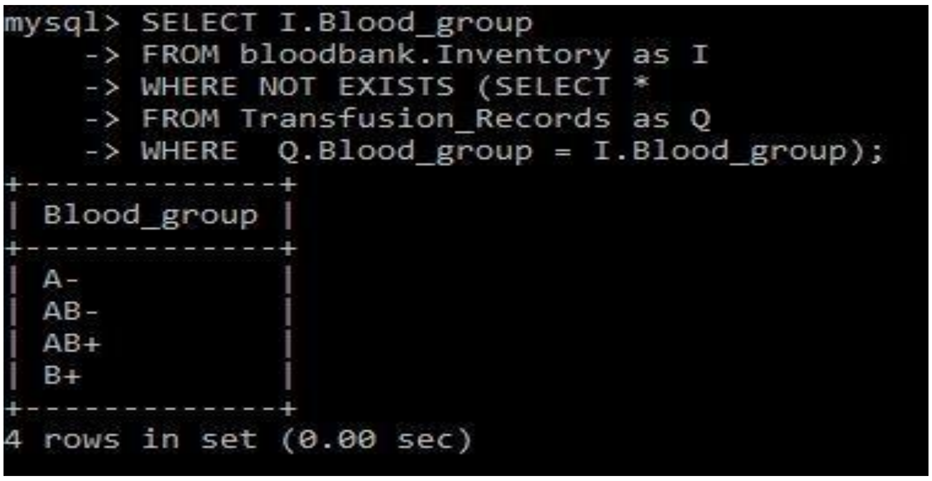
```
SELECT I.Blood_group
FROM bloodbank.Inventory as I
WHERE NOT EXISTS (SELECT *
                        FROM Donation_Records as P
                        WHERE P.Blood_group = I.Blood_group);
```



```
mysql> SELECT I.Blood_group
    -> FROM bloodbank.Inventory as I
    -> WHERE NOT EXISTS (SELECT *
    -> FROM Donation_Records as P
    -> WHERE  P.Blood_group = I.Blood_group);
+-------------+
| Blood_group |
+-------------+
| A-          |
| AB-         |
| AB+         |
| B-          |
| B+          |
| O-          |
+-------------+
6 rows in set (0.00 sec)
```

- **This query helps in showing which blood group has notbeen used by the patients (recipients) -**

```
SELECT I.Blood_group
FROM bloodbank.Inventory as I
WHERE NOT EXISTS (SELECT *
                        FROM Transfusion_Records as Q
                        WHERE Q.Blood_group = I.Blood_group);
```



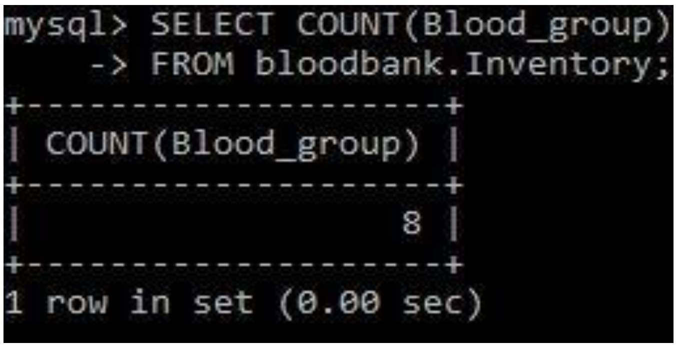
```
mysql> SELECT I.Blood_group
    -> FROM bloodbank.Inventory as I
    -> WHERE NOT EXISTS (SELECT *
    -> FROM Transfusion_Records as Q
    -> WHERE  Q.Blood_group = I.Blood_group);
+-------------+
| Blood_group |
+-------------+
| A-          |
| AB-         |
| AB+         |
| B+          |
+-------------+
4 rows in set (0.00 sec)
```

- **This query helps in calculating the total sum, maximum , minimum and average of the stock quantity of blood present in the inventory**

SELECT SUM(Stock_quantity), MAX(Stock_quantity),
MIN(Stock_quantity), AVG(Stock_quantity)
FROM bloodbank.Inventory;

- **This query helps in counting the total blood groups present in the blood bank inventory -**

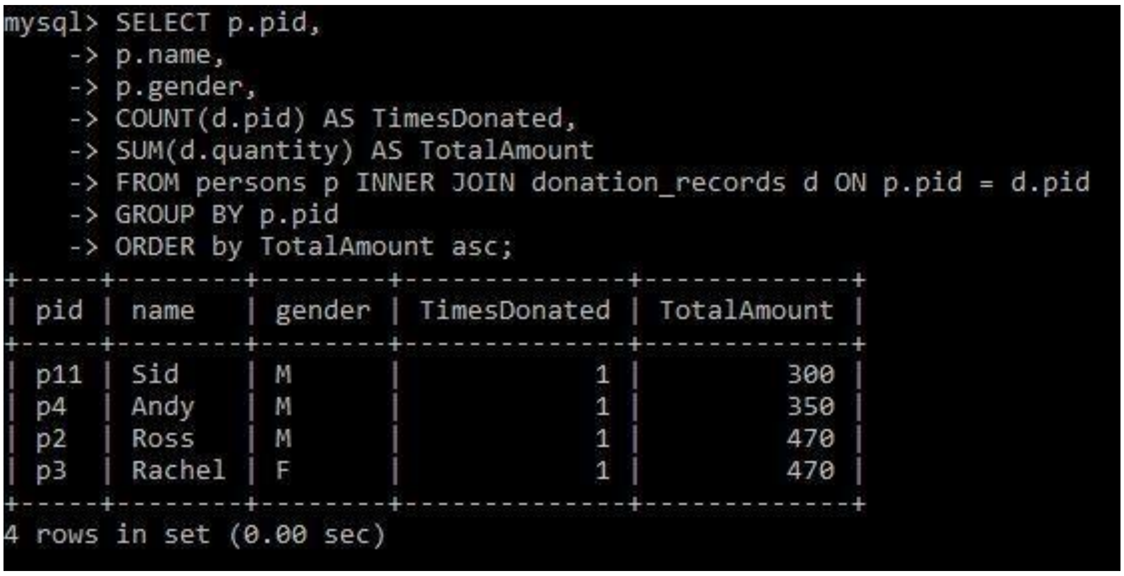
SELECT COUNT(Blood_group)
FROM bloodbank.Inventory;



```
mysql> SELECT COUNT(Blood_group)
    -> FROM bloodbank.Inventory;
+--------------------+
| COUNT(Blood_group) |
+--------------------+
|                  8 |
+--------------------+
1 row in set (0.00 sec)
```

- **This query used the INNER JOIN commands and joins the table persons and donation_records accordingly** -

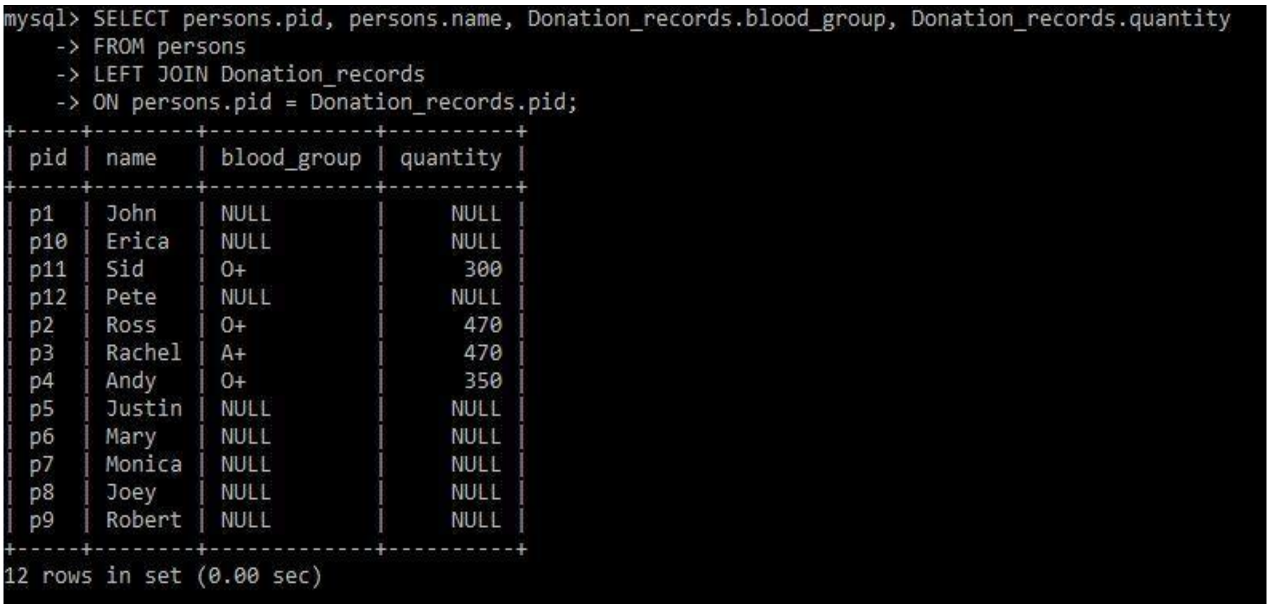
  SELECT p.pid,
  p.name,
  p.gender,
  COUNT(d.pid) AS TimesDonated,
  SUM(d.quantity) AS TotalAmount
  FROM persons p INNER JOIN donation_records d ON p.pid = d.pid
  GROUP BY p.pid
  ORDER by TotalAmount asc;



```
mysql> SELECT p.pid,
    -> p.name,
    -> p.gender,
    -> COUNT(d.pid) AS TimesDonated,
    -> SUM(d.quantity) AS TotalAmount
    -> FROM persons p INNER JOIN donation_records d ON p.pid = d.pid
    -> GROUP BY p.pid
    -> ORDER by TotalAmount asc;
+-----+--------+--------+--------------+-------------+
| pid | name   | gender | TimesDonated | TotalAmount |
+-----+--------+--------+--------------+-------------+
| p11 | Sid    | M      |            1 |         300 |
| p4  | Andy   | M      |            1 |         350 |
| p2  | Ross   | M      |            1 |         470 |
| p3  | Rachel | F      |            1 |         470 |
+-----+--------+--------+--------------+-------------+
4 rows in set (0.00 sec)
```

- **This query used the LEFT JOIN command and joins the table persons and Donation_records accordingly -**

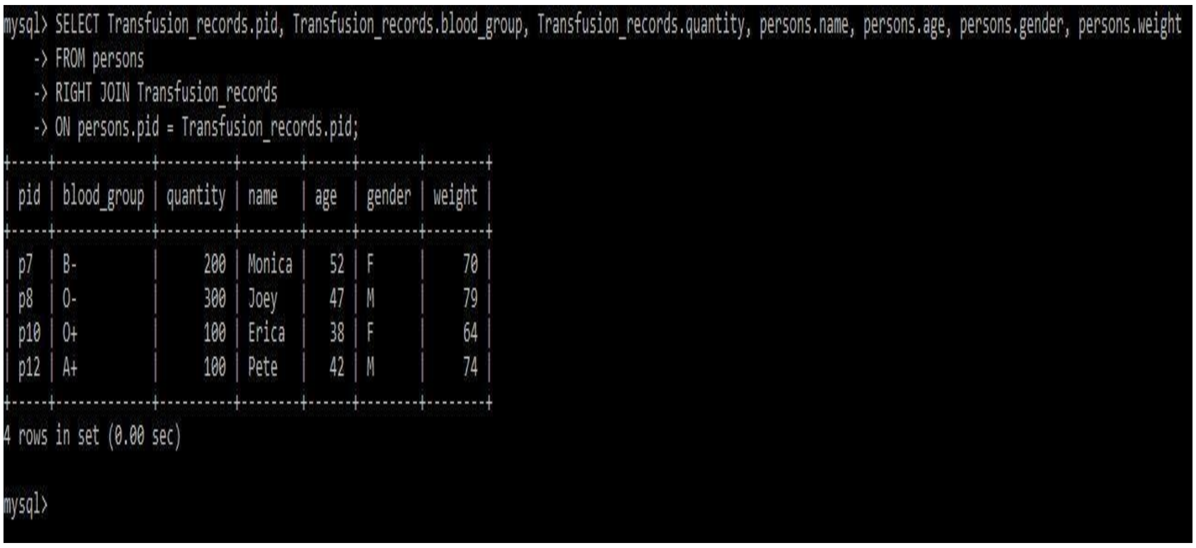
  SELECT persons.pid, persons.name, Donation_records.blood_group, Donation_records.quantity
  FROM persons
  LEFT JOIN Donation_records
  ON persons.pid = Donation_records.pid;



```
mysql> SELECT persons.pid, persons.name, Donation_records.blood_group, Donation_records.quantity
    -> FROM persons
    -> LEFT JOIN Donation_records
    -> ON persons.pid = Donation_records.pid;
+-----+--------+-------------+----------+
| pid | name   | blood_group | quantity |
+-----+--------+-------------+----------+
| p1  | John   | NULL        |     NULL |
| p10 | Erica  | NULL        |     NULL |
| p11 | Sid    | O+          |      300 |
| p12 | Pete   | NULL        |     NULL |
| p2  | Ross   | O+          |      470 |
| p3  | Rachel | A+          |      470 |
| p4  | Andy   | O+          |      350 |
| p5  | Justin | NULL        |     NULL |
| p6  | Mary   | NULL        |     NULL |
| p7  | Monica | NULL        |     NULL |
| p8  | Joey   | NULL        |     NULL |
| p9  | Robert | NULL        |     NULL |
+-----+--------+-------------+----------+
12 rows in set (0.00 sec)
```

- **This query used the RIGHT JOIN command and joins the table persons and Transfusion_records accordingly -**

  SELECT Transfusion_records.pid, Transfusion_records.blood_group, Transfusion_records.quantity, persons.name, persons.age, persons.gender, persons.weight
  FROM persons
  RIGHT JOIN Transfusion_records
  ON persons.pid = Transfusion_records.pid;



```
mysql> SELECT Transfusion_records.pid, Transfusion_records.blood_group, Transfusion_records.quantity, persons.name, persons.age, persons.gender, persons.weight
    -> FROM persons
    -> RIGHT JOIN Transfusion_records
    -> ON persons.pid = Transfusion_records.pid;
+-----+-------------+----------+--------+-----+--------+--------+
| pid | blood_group | quantity | name   | age | gender | weight |
+-----+-------------+----------+--------+-----+--------+--------+
| p7  | B-          |      200 | Monica |  52 | F      |     70 |
| p8  | O-          |      300 | Joey   |  47 | M      |     79 |
| p10 | O+          |      100 | Erica  |  38 | F      |     64 |
| p12 | A+          |      100 | Pete   |  42 | M      |     74 |
+-----+-------------+----------+--------+-----+--------+--------+
4 rows in set (0.00 sec)

mysql>
```

CONCLUSION

This project gave me an idea of how a bloodbank works and how it stores all the data. This not only helped us to understand bloodbank management in particular but also helped us understand in general how data is collected and stored in the database and how various operations can be performed on that.
The project gave me clarity of all the concepts used in Data base management system.
The project also helps in keeping record of every donor and patient without any ambiguity.
There are also triggers installed which does not need the user(administrator) to do any manual changes , which in turn reduces mistakes and ambiguity and stores data correctly.
The main benefit of any Database Management is that it reduces ambiguity (confusion) and stores any record accurately.

FUTURE ENHANCEMENTS

- I would like to add a requests table which would help patients put a request as it would help people who are actually in need of blood get a faster access.
- Also add a locations attribute which would help in storing records across different locations. This would expand our database across states and then across country.
- Implement checks on blood type input to make sure it is avalid blood type input.
- Implement a way to check the global inventory.