A REPORT

ON

BLOOD BANK MANAGEMENT SYSTEM (PROJECT 15)

BY

AADIT NAYYAR	2021A7PS2687P
YASH SEJPAL	2021A7PS2683P

AT

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI, PILANI CAMPUS

APRIL 2023

Anti-Plagiarism Statement

1.) We know that plagiarism means taking and using the ideas, writings, works or inventions of another as if they were one's own. We know that plagiarism not only includes verbatim copying, but also the extensive use of another person's ideas without proper acknowledgement . We know that plagiarism covers this sort of use of material found in textual sources and from the Internet.

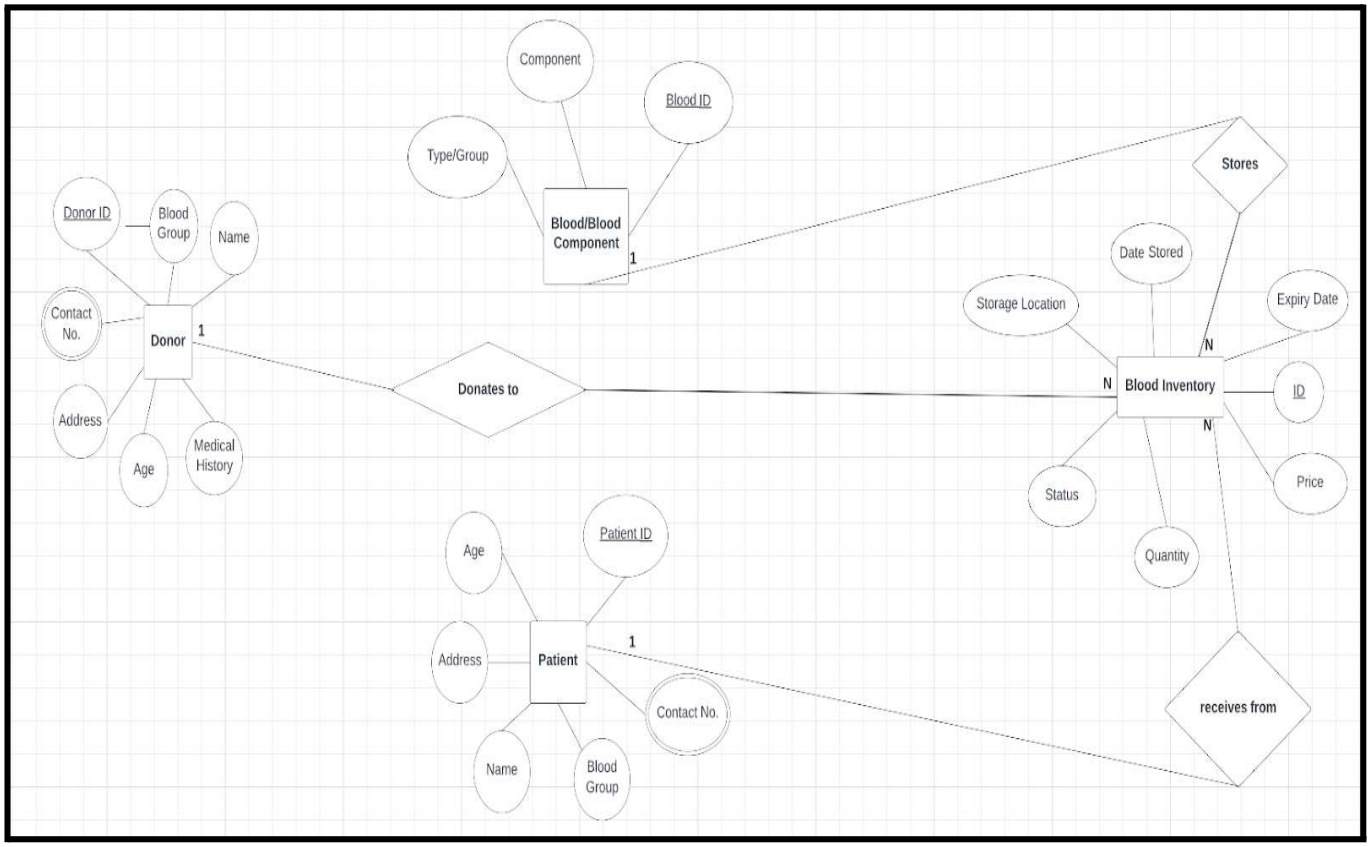
2.) We acknowledge and understand that plagiarism is wrong.

3.) All the codes written throughout this project have been written by us. All the work done in this project is our original work.

4.) We have not allowed, nor will we in the future allow, anyone to copy my work with the intention of passing it off as their own work.

5.) We also understand that if we are found to have violated this policy, we will face the consequences accordingly.

Entity - Relationship Diagram



Some statements explaining the ER diagram are:
1)A donor donates to the blood inventory
2)The blood inventory stores blood and blood components
3)A patient receives the requirement from the blood inventory
4)Inventory represents one blood unit
5)1 Donor can Donate N blood units but 1 blood unit can be donated by one donor only
6)1 patient can receive N blood units but 1 blood unit can be utilized by 1 patient only
7)1 blood unit can store only 1 blood component but 1 blood component can be stored in N blood units

Relational Schema

Conversion

To create a relational schema from an ER diagram, the first step is to identify all entities in the ER diagram and create a table for each entity. We created a table for each entity in the relational schema. We made sure that the name of each table matched the name of the corresponding entity, and that each table had a primary key column that uniquely identifies each row in the table.

Next, we identified the primary key and any foreign keys that reference other tables for each table. We made sure that the primary key column uniquely identified each row in the table, and if an entity had a relationship with another entity, we created a foreign key column in its table that referenced the primary key of the related table. For example, for the table “Inventory”, we implement the foreign key “Donor_ID” which is a primary key of the entity table “Donor”.

For each attribute of an entity in the ER diagram, we created a corresponding column in the table for the entity in the relational schema. We made sure that the column name matched the attribute name.

Overall we have one type of relationship in our ER diagram and that is One-to-Many relationship.

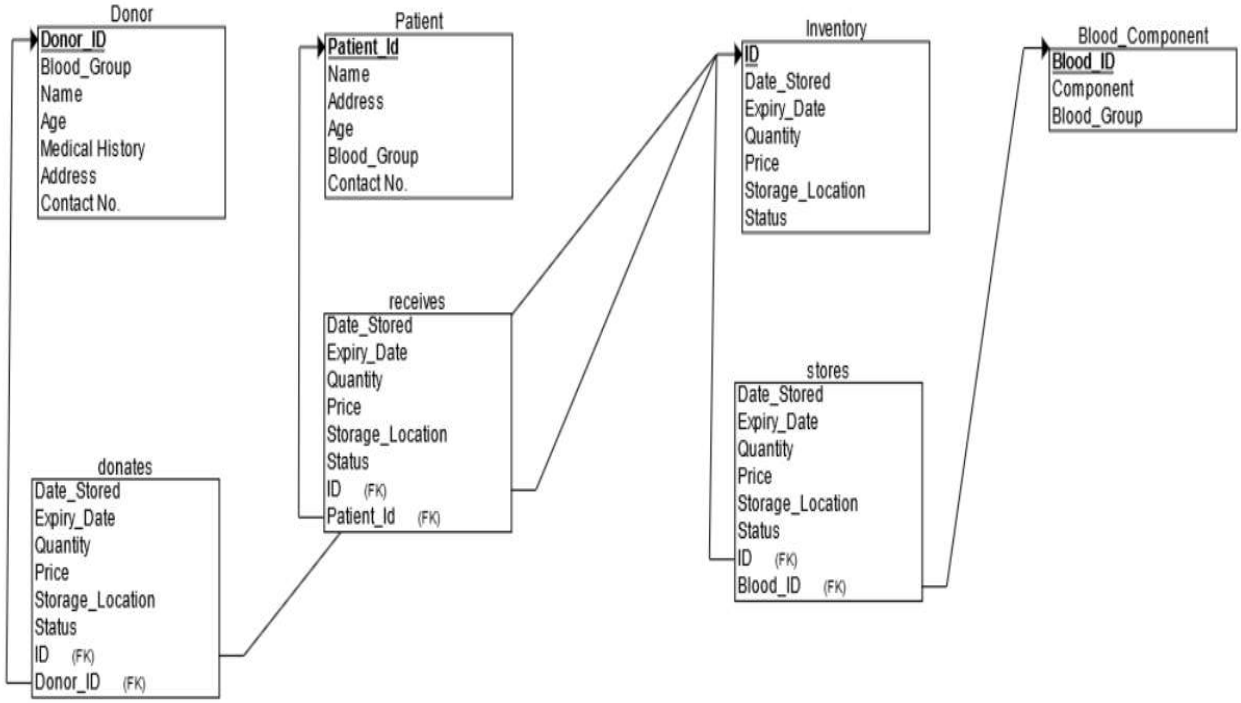
Donates_to is a one to many relationship with total participation by Inventory.

Receives_from is a one to many relationship between patient and inventory.

Stores is a one to many relationship between Blood/Blood Components and Inventory with total participation by Inventory.

For Many-to-One relationships, I had to add separate tables in the Relational Schema which contains all the attributes of the entity which is in the Many side and the primary key of the entity which is in the One side. For example, the relationship Donates_to is represented in the relational schema with a separate table which contains all the attributes of Inventory and the primary key of Donor which is referenced as a foreign key.

The relational schema after conversion from the ER diagram is shown below:



Functional Dependencies and Normalisation

The functional dependencies that exist in our relational schema are as follows:

- The **Date_Stored,Expiry_Date,Quantity,Price,Storage_Location,Status** all depend on **ID** in the inventory table
- The **Component and Blood_Group** depend on **Blood_ID** in the Blood_Component table.
- The **Blood_Group,Name,Age,Medical_History,Address,Contact_No.,** all depend on Donor_ID in the Donor table
- The **Name,Address,Age,Blood_group,Contact_No,** all depend on the **Patient_ID** in the Patient table.

The relational schema is currently not normalized as multivalued attributes like Patient Contact No. and Donor Contact No. are present in the same table as the other attributes. A relation is in 1 NF if its attributes all contain atomic values. After converting to 1 NF, the multivalued attributes are contained in separate relations.

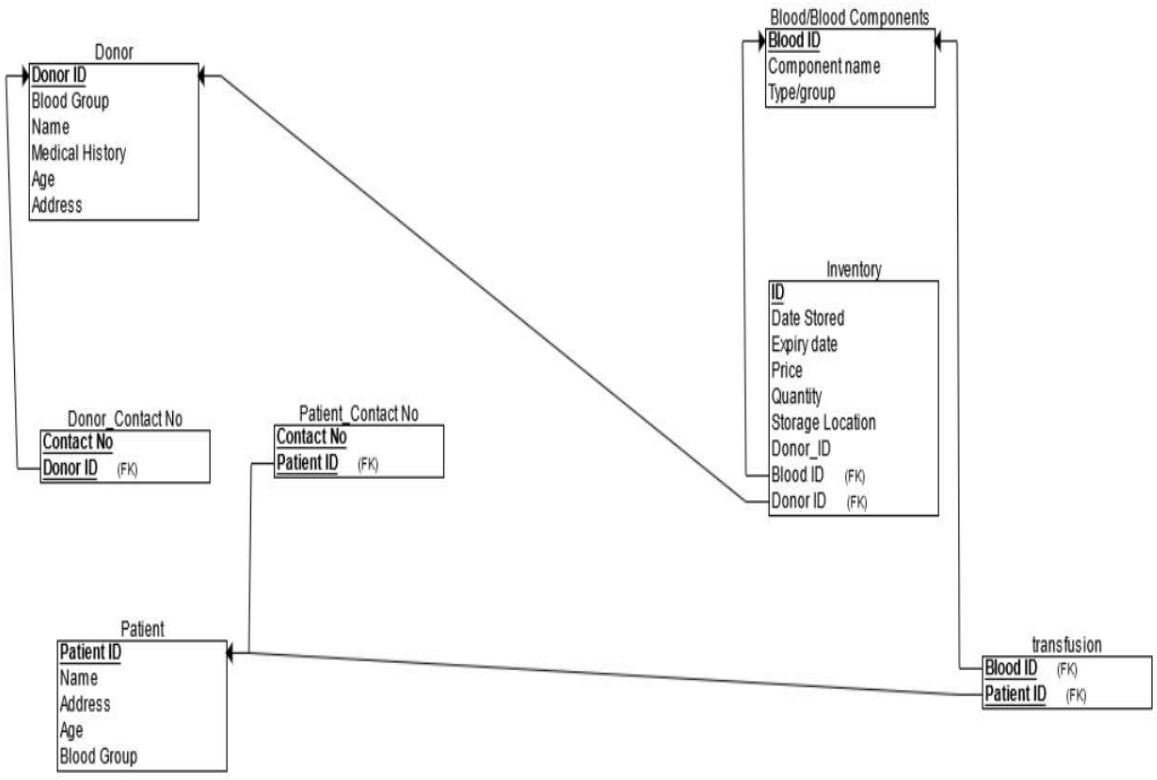
Then we check if the relational schema is in 2 NF. A relation is in 2 NF if there is complete functional dependency. The schema is in 2 NF.

For 3 NF we check if there is any transitive functional dependency.

A relation is in 3NF if at least one of the following condition holds in every non-trivial functional dependency X –> Y:

1. X is a super key.
2. Y is a prime attribute (each element of Y is part of some candidate key).

The relational schema after converting into 3 NF is shown below:



As it can be seen, there is no transitive functional dependency and each relation is fully functionally dependent.

With this Relational Schema created from the ER diagram, we were able to create a database by executing the appropriate SQL commands to create the tables and constraints in the database.

SQL QUERIES

- Query to retrieve the list of all blood units with their current availability status.

```
SELECT ID, Component, Blood_Group, Status, Price, Quantity
FROM Inventory
JOIN Blood_Component ON Inventory.Blood_ID = Blood_Component.Blood_ID;
```

ID	Component	Blood_Group	Status	Price	Quantity
B001	Red Blood Cells	A+	transfused	200	10
B002	Plasma	B+	Available	300	10
I003	Red Blood Cells	A+	Available	500	30

- Query to retrieve the list of all blood donors along with their personal details and donation history.
  The LEFT JOIN is used to include all donors from the "Donor" table, even if they have not made any donations yet.

```
SELECT D.Donor_ID, D.Name, D.Blood_Group, D.Medical_History, D.Age,
D.Address, DH.Date_Stored
FROM Donor D
LEFT JOIN Inventory DH ON D.Donor_ID = DH.Donor_ID
ORDER BY D.Donor_ID;
```

Donor_ID	Name	Blood_Group	Medical_History	Age	Address	Date_Stored
D001	John Smith	A+	None	27	123 Main St.	2023-04-11
D001	John Smith	A+	None	27	123 Main St.	2023-04-11
D002	Emily Johnson	B+	High blood pressure	35	456 Elm St.	2023-04-11
D003	Adam Lee	O+	Allergies	42	789 Maple St.	NULL
D004	Lila Patel	AB+	None	21	321 Oak St.	NULL
D005	Tyler Davis	A-	Diabetes	29	654 Pine St.	NULL
D006	Sophia Wilson	B-	None	30	987 Cedar St.	NULL
D007	Michael Brown	O-	Asthma	46	246 Birch St.	NULL
D008	Avery Taylor	AB-	None	25	369 Spruce St.	NULL
D009	Noah Thomas	A+	None	32	753 Hickory St.	NULL
D010	Olivia Hernan...	O+	None	39	159 Willow St.	NULL

- Query to add a new blood donor to the database.

INSERT INTO Donor (Donor_ID, Blood_Group, Name, Medical_History, Age, Address)
VALUES ('D001', 'A+', 'John Doe', 'No medical history', 30, '1234 Elm Street, Springfield, IL');

- Query to update the information of a blood donor.

UPDATE Donor
SET Blood_Group = 'B+', Name = 'Jane Smith', Medical_History = 'No significant medical history', Age = 35, Address = '5678 Oak Street, Chicago, IL'
WHERE Donor_ID = 'D001';

- Query to add a new blood unit to the inventory.

INSERT INTO Inventory (Date_Stored, Expiry_Date, ID, Price, Quantity, Storage_Location, Status, Donor_ID, Blood_ID)
VALUES ('2023-04-11', '2023-06-11', 'I001', 50, 1, 'Refrigerator 1, Shelf 3', 'Available', 'D001', 'B001');

- Query to retrieve the information of all the donors whose blood group is AB+ .

```
SELECT *
FROM Donor
WHERE Blood_Group = 'AB+';
```

Donor_ID	Blood_Group	Name	Medical_History	Age	Address
D004	AB+	Lila Patel	None	21	321 Oak St.
NULL	NULL	NULL	NULL	NULL	NULL

- Query to delete the information of a specific donor.

```
DELETE FROM Donor
WHERE Donor_ID = 'D001';
```

- Query to retrieve the list of blood units that are about to expire in the next 30 days.

```
SELECT *
FROM Inventory
WHERE Expiry_date <= DATE_ADD(CURDATE(), INTERVAL 30 DAY);
```

Date_Stored	Expiry_date	ID	Price	Quantity	Storage_Location	Status	Donor_ID	Blood_ID
2023-02-15	2023-04-15	I001	100	10	Shelf A	Available	D001	B001
2023-02-01	2023-04-01	I002	150	5	Shelf B	Expired	D002	B001
2023-03-01	2023-05-01	I004	120	8	Shelf D	Available	D004	B002
2023-02-25	2023-04-25	I005	90	12	Shelf E	Available	D005	B003
2023-03-05	2023-05-05	I006	80	6	Shelf F	Available	D006	B003
2023-02-10	2023-04-10	I007	70	3	Shelf G	Expired	D007	B004
2023-02-20	2023-04-20	I008	200	20	Shelf H	Available	D008	B004
2023-01-05	2023-03-05	I009	150	7	Shelf I	Expired	D009	B005
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

- Query to get the Blood ID of a particular combination of Blood Group and Component

```
SELECT Blood_ID
FROM Blood_Component
WHERE blood_group = "A+" AND component = "Red Blood Cells";
```

	Blood_ID
▶	B001
*	NULL

- Query to get the Available blood units in ascending order of Expiry Date

```
SELECT ID, Component, Expiry_date FROM Inventory_View WHERE Blood_Group =
"A+" AND Status = 'Available' ORDER BY Expiry_date;
```

	ID	Component	Expiry_Date
▶	I003	Red Blood Cells	2023-05-23

- Query to get the Transfusion History of a particular Patient

```
SELECT t.Patient_ID, i.ID, t.Date_Transfused, i.Donor_ID
FROM transfusion t
JOIN inventory i ON t.Unit_ID = i.ID
WHERE t.Patient_ID = "P001";
```

	Patient_ID	ID	Date_Transfused	Donor_ID
▶	P001	B001	2023-04-11	D001

STEPS TO RUN PYTHON FILE

1. Run the SQL script(.sql) files in MySQL Workbench, in the order tables.sql -> triggers.sql -> data.sql

2. Open the file home.py in a text editor and add the database host, username and password to line 2 as shown in the figure below.

3. Run the python file.

Drive link for the recorded videos:
https://drive.google.com/drive/folders/1TaJ84NXeoiQ0tMaL5DPF9tw7J3Ky6bPr