THE CATHOLIC UNIVERSITY OF EASTERN AFRICA

CONSECRATE THEM IN THE TRUTH

BLOOD BANK MANAGEMENT SYSTEM

GROUP 26

1061783 – MUNGE WANJIKU

1061734 – MARIAL MABOR

1052076 – GICHUKI MWANGI LEE

1037410 – JOSHUA AGEGA MAYIEKA

1048777 – AUTA JANE SHARON

DUE DATE: 30TH NOVEMBER 2024

INTRODUCTION

Overview

The BloodBank Management System is a database-driven application designed to streamline the processes associated with blood donation, inventory management, and recipient allocation. This system integrates key modules, including donors, blood banks, blood inventory, recipients, and staff, enabling efficient monitoring and management of resources critical to healthcare operations.

Rationale

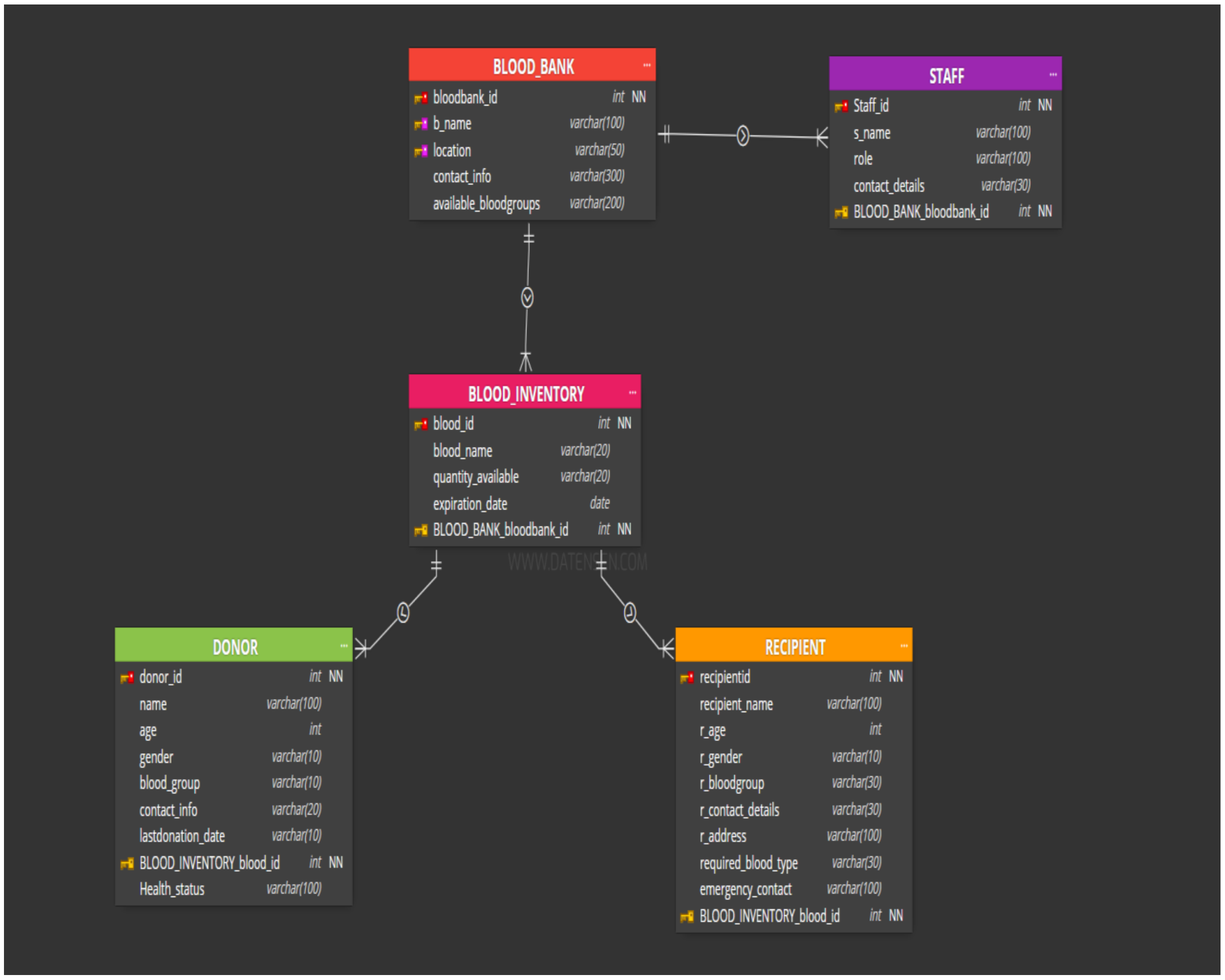
Efficient blood management is a critical requirement in healthcare to address emergencies, surgeries, and routine medical treatments. Traditional paper-based systems often lead to inefficiencies, errors, and delays. This system is developed to provide a centralized platform for managing donor details, blood inventory, and recipient information while ensuring real-time data consistency and accuracy.

Objectives

1. To provide a structured database for managing blood banks, donors, and recipients.
2. To streamline blood inventory tracking, ensuring the availability of various blood types.
3. To enable efficient allocation of blood resources to recipients based on compatibility.
4. To improve the overall efficiency of blood bank operations by reducing errors and enhancing data accessibility.

SYSTEM DESIGN

ER DIAGRAM


BLOOD_BANK
bloodbank_id int NN
b_name varchar(100)
location varchar(50)
contact_info varchar(300)
available_bloodgroups varchar(200)
STAFF
Staff_id int NN
s_name varchar(100)
role varchar(100)
contact_details varchar(30)
BLOOD_BANK_bloodbank_id int NN
BLOOD_INVENTORY
blood_id int NN
blood_name varchar(20)
quantity_available varchar(20)
expiration_date date
BLOOD_BANK_bloodbank_id int NN
DONOR
donor_id int NN
name varchar(100)
age int
gender varchar(10)
blood_group varchar(10)
contact_info varchar(20)
lastdonation_date varchar(10)
BLOOD_INVENTORY_blood_id int NN
Health_status varchar(100)
RECIPIENT
recipientid int NN
recipient_name varchar(100)
r_age int
r_gender varchar(10)
r_bloodgroup varchar(30)
r_contact_details varchar(30)
r_address varchar(100)
required_blood_type varchar(30)
emergency_contact varchar(100)
BLOOD_INVENTORY_blood_id int NN

DATABASE SCHEMA

This is the schema for a blood group management system, with the following entities:

1. Donor

   Attributes:

   - DonorID

- Name
- Age
- BloodGroup
- Phone (Contact No)
- LastDonationDate
- Gender

- Description: Represents individuals who donate blood.

Relationship: A blood inventory requests blood from one or more donors

2. Recipient

Attributes:

- RecipientID
- Name
- Age
- BloodGroup
- Phone
- RequiredBloodType
- EmergencyContact
- Gender

- Description: Represents patients who need blood.

- Relationship: A request is made from the blood inventory by a recipient

3. BloodBank

Attributes:

- BloodBankID
- Name
- Location
- AvailableBloodGroups
- Phone/Contact

- Description: Represents the blood bank facility with available blood types.

- Relationship:A blood bank stores multiple blood types in the inventory

4. BloodInventory

Attributes:

- InventoryID
- BloodGroupName
- QuantityAvailable
- ExpirationDate

-Description: Represents the inventory of blood available in the blood bank.

- Relationship : A has one or many recipients/donors

5. Staff

Attributes:

- Staff_id
- S_name
- Role
- Contact_info
- Description :it shows all the staff members working in the blood bank
- Relationship: a blood bank has one or many staff members

DATABASE SCRIPT

```
show databases;
create database BloodBank_management;
use BloodBank_management;
show tables ;

CREATE TABLE DONOR
(
Donor_id INT PRIMARY KEY NOT NULL,
Name VARCHAR(200) NOT NULL,
Age INT,
Gender VARCHAR(6),
Blood_group VARCHAR(30) NOT NULL,
Contact_Info VARCHAR(15) UNIQUE,
Address VARCHAR(50),
Last_donationDate Date,
Health_status VARCHAR(100) NOT NULL,
BloodInventory_id INT,
FOREIGN KEY (BloodIventory_id) REFERENCES BLOOD_INVENTORY (BloodInventory_id)
);

CREATE TABLE BLOODBANK
(
Bloodbank_id INT PRIMARY KEY NOT NULL,
B_name VARCHAR(200) NOT NULL,
Location  VARCHAR(50) NOT NULL,
Contact_Info VARCHAR(15) UNIQUE,
Avaiable_bloodgroups VARCHAR(50)
);

CREATE TABLE RECIPIENT
(
Recipient_id INT PRIMARY KEY NOT NULL,
Recipient_name VARCHAR(50) NOT NULL,
R_age INT,
R_gender VARCHAR(6),
R_bloodgroup VARCHAR(30),
R_contactdetails VARCHAR(30),
R_address VARCHAR(50),
Emergency_contact VARCHAR(30),
Bloodbank_id INT,
FOREIGN KEY (Bloodbank_id) REFERENCES BLOODBANK (Bloodbank_id)
);

CREATE TABLE BLOOD_INVENTORY
(
BloodInventory_id INT PRIMARY KEY NOT NULL,
Blood_name VARCHAR(50) NOT NULL,
Quantity_available VARCHAR(50),
Expiration_date DATE,
Bloodbank_id INT,
FOREIGN KEY (Bloodbank_id) REFERENCES BLOODBANK (Bloodbank_id)
);

CREATE TABLE STAFF
(
Staff_id INT PRIMARY KEY NOT NULL,
S_name VARCHAR(50) NOT NULL,
Role VARCHAR(50) NOT NULL,
Contact_Info VARCHAR(15) UNIQUE,
Bloodbank_id INT,
FOREIGN KEY (Bloodbank_id) REFERENCES BLOODBANK (Bloodbank_id)
);
```

CRUD OPERATIONS

The system implemented the following crud operations :

1. Create Operation:

We created five tables and inserted data into each table, with approximately 10 entries per table. Below is an example of the staff table creation:

```
CREATE TABLE STAFF

(

Staff_id INT PRIMARY KEY NOT NULL,

S_name VARCHAR(50) NOT NULL,

Role VARCHAR(50) NOT NULL,

Contact_Info VARCHAR(15) UNIQUE,

Bloodbank_id INT,

FOREIGN KEY (Bloodbank_id) REFERENCES BLOODBANK (Bloodbank_id)

);
```

2. Update Operation:

We updated various tables such as BLOOD_INVENTORY, DONOR, and BLOODBANK. Below are a few examples of the update operations:

```
Update BLOOD_INVENTORY SET Quantity_Available='20 units' where BloodInventory_id=4;

update Donor set contact_info ='25411111111' where Donor_id=5;

update BLOODBANK set B_name='Breakdown Hospital' where Bloodbank_id=7;
```

3. Delete Operation:

We performed delete operations across various tables. Below are examples of the delete queries:

Examples:

Delete from Recipient where Recipient_id=7;

Delete from STAFF where Staff_id=4;

delete from DONOR where BloodInventory_id=1;

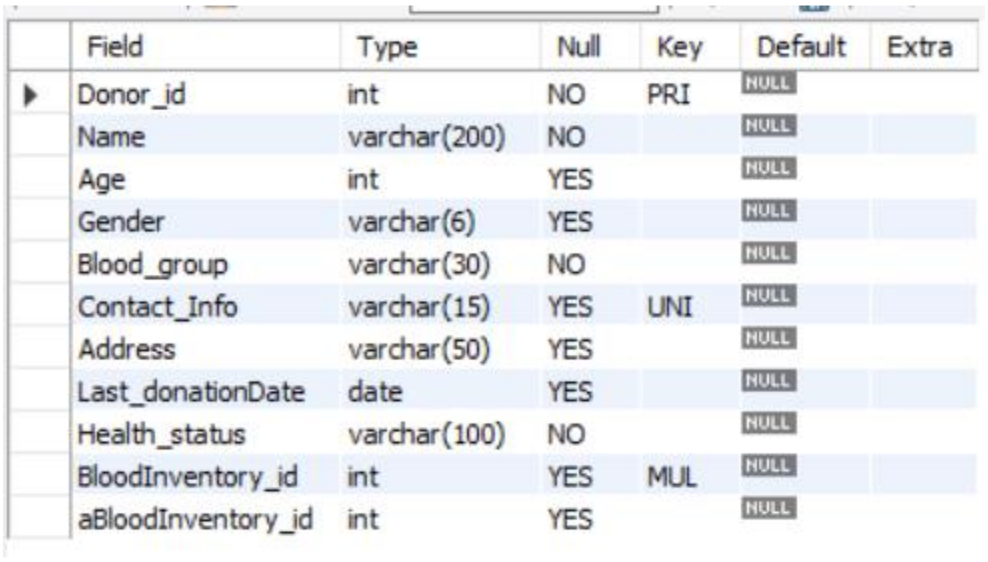
TESTING AND VALIDATION

During the database creation and development phase, rigorous testing was conducted to ensure that all operations performed as expected. Below are examples of tables and operations that were validated, along with sample outputs.

Donor table:

```
CREATE TABLE DONOR

(

Donor_id INT PRIMARY KEY NOT NULL,

Name VARCHAR(200) NOT NULL,

Age INT,

Gender VARCHAR(6),

Blood_group VARCHAR(30) NOT NULL,

Contact_Info VARCHAR(15) UNIQUE,

Address VARCHAR(50),

Last_donationDate Date,

Health_status VARCHAR(100) NOT NULL,

BloodInventory_id INT,

FOREIGN KEY (BloodInventory_id) REFERENCES BLOOD_INVENTORY (BloodInventory_id)

);
```

Output



Field	Type	Null	Key	Default	Extra
Donor_id	int	NO	PRI	NULL	
Name	varchar(200)	NO		NULL	
Age	int	YES		NULL	
Gender	varchar(6)	YES		NULL	
Blood_group	varchar(30)	NO		NULL	
Contact_Info	varchar(15)	YES	UNI	NULL	
Address	varchar(50)	YES		NULL	
Last_donationDate	date	YES		NULL	
Health_status	varchar(100)	NO		NULL	
BloodInventory_id	int	YES	MUL	NULL	
aBloodInventory_id	int	YES		NULL	

2.blood inventory

INSERT INTO BLOOD_INVENTORY (BloodInventory_id, Blood_name, Quantity_available, Expiration_date, Bloodbank_id) VALUES

(1, 'A+', '50 Units', '2024-12-01', 1),

(2, 'O+', '30 Units', '2024-11-15', 2),

(3, 'B+', '40 Units', '2024-12-10', 3),

(4, 'AB+', '20 Units', '2024-11-20', 4),

(5, 'A-', '60 Units', '2024-12-05', 5),

(6, 'B-', '35 Units', '2024-11-25', 6),

(7, 'AB-', '15 Units', '2024-12-15', 7),

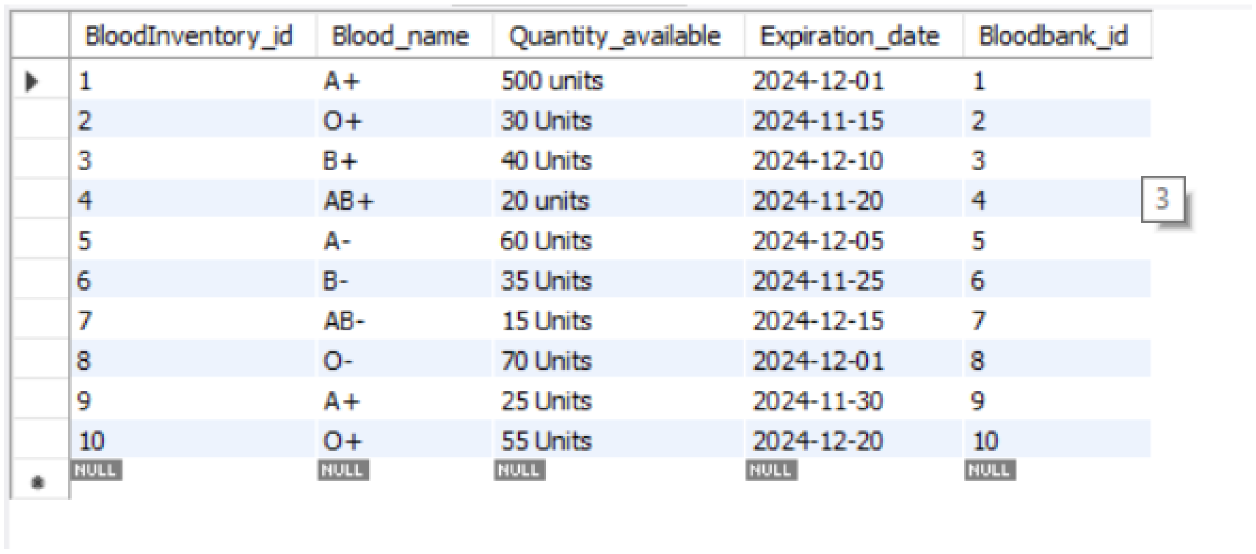
(8, 'O-', '70 Units', '2024-12-01', 8),

(9, 'A+', '25 Units', '2024-11-30', 9),

(10, 'O+', '55 Units', '2024-12-20', 10);

*select * from Blood_inventory;*

output



	BloodInventory_id	Blood_name	Quantity_available	Expiration_date	Bloodbank_id
▶	1	A+	500 units	2024-12-01	1
	2	O+	30 Units	2024-11-15	2
	3	B+	40 Units	2024-12-10	3
	4	AB+	20 units	2024-11-20	4
	5	A-	60 Units	2024-12-05	5
	6	B-	35 Units	2024-11-25	6
	7	AB-	15 Units	2024-12-15	7
	8	O-	70 Units	2024-12-01	8
	9	A+	25 Units	2024-11-30	9
	10	O+	55 Units	2024-12-20	10
*	NULL	NULL	NULL	NULL	NULL

3

CONCLUSION AND RECOMMENDATIONS

Conclusions

1. **Data Integrity:** The system ensures data consistency and accuracy by enforcing foreign key and unique constraints. This ensures that donor and recipient details are linked appropriately to the relevant blood inventory and blood bank**.**
2. **Efficiency and Performance**: Query performance is optimized with effective use of joins and aggregate functions. Complex queries, such as matching blood donations to recipients, are handled efficiently by the database, and updates or deletions are seamlessly reflected across tables.
3. **Error Handling and Security**: Proper error handling mechanisms are in place to block invalid data entries, such as duplicate contact information or missing foreign keys, ensuring that data integrity is maintained throughout the system. Deletions of records are managed in a way that prevents orphaned data, ensuring smooth database operations.
4. **Data Accuracy**: Testing results confirm that the system accurately handles aggregate queries and displays correct counts and totals, including blood inventory and donor information. This ensures that blood banks can maintain accurate records of blood donations, stock levels, and recipient needs.

Recommendations:

1. **User Interface (UI)**: The system's back-end design is robust, but the front-end interface for staff and recipients could be developed to be more user-friendly. It should be designed with intuitive navigation to make it easier for blood bank staff to manage the data.
2. **Scalability**: As the database is expected to grow with the number of donors, recipients, and blood banks, it is recommended to periodically review the performance of complex queries and optimize indexing to maintain system speed and responsiveness.
3. **Security Enhancements**: While the system includes data validation at the database level, it is essential to incorporate more security measures, such as user role-based access

control (RBAC) to ensure that only authorized staff can access sensitive data like donor and recipient details.

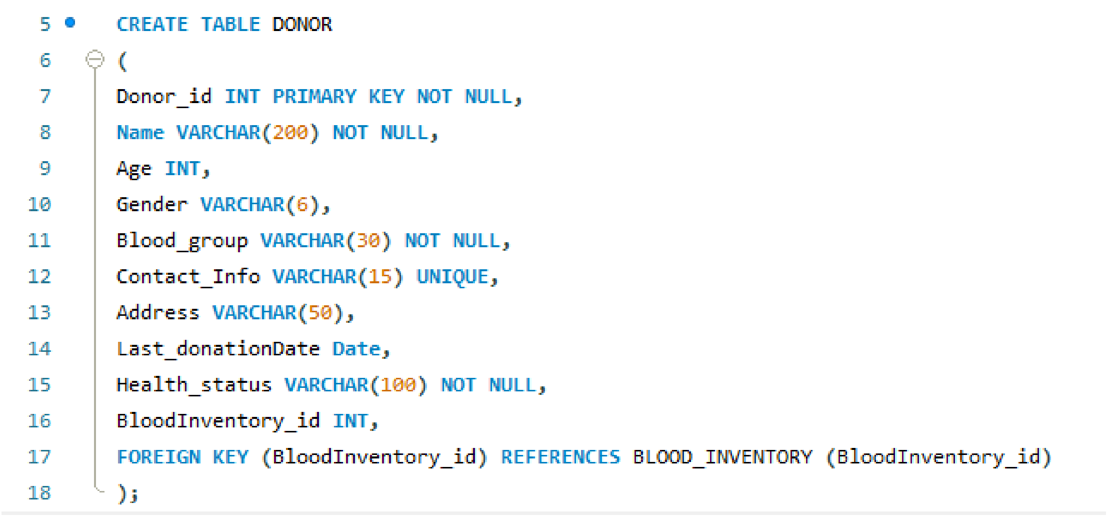
4. **Regular Backups**: To prevent data loss, especially for critical information such as blood inventory and recipient data, regular backups of the database should be scheduled. Additionally, implementing a disaster recovery plan would safeguard the system from unexpected data loss.
5. **Integration with Other Healthcare Systems**: For a more holistic healthcare solution, the BloodBank Management System could be integrated with other hospital management systems or emergency services. This would help streamline blood requests during emergencies or surgeries.
6. **Real-time Inventory Updates**: While the current system supports inventory tracking, it is recommended to implement real-time updates for inventory levels, especially when blood is donated or used. This would allow staff to better predict shortages and plan for future needs.
7. **Feedback System**: Implementing a feedback system for both donors and recipients can help improve the quality of service and track potential issues, enhancing the overall donor experience and making the system more adaptable.

By incorporating these recommendations, the BloodBank Management System can be further optimized to meet the needs of the healthcare industry, ensuring timely and effective blood supply management.

APPPENDICES

1. Code snippets

   Sample create table commands for donor, bloodbank, blood_inventory, and etc.

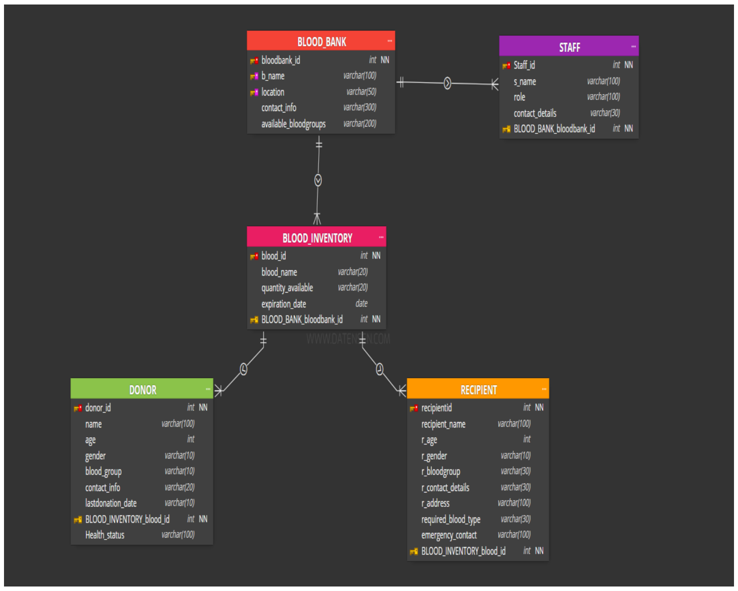


```sql
CREATE TABLE DONOR
(
Donor_id INT PRIMARY KEY NOT NULL,
Name VARCHAR(200) NOT NULL,
Age INT,
Gender VARCHAR(6),
Blood_group VARCHAR(30) NOT NULL,
Contact_Info VARCHAR(15) UNIQUE,
Address VARCHAR(50),
Last_donationDate Date,
Health_status VARCHAR(100) NOT NULL,
BloodInventory_id INT,
FOREIGN KEY (BloodInventory_id) REFERENCES BLOOD_INVENTORY (BloodInventory_id)
);
```

   Insert, update, and delete commands for managing data

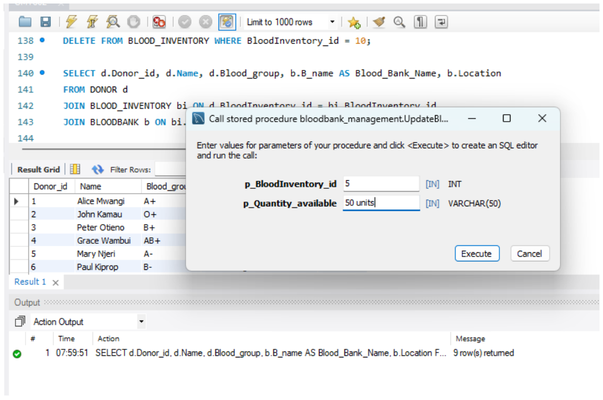
2. Diagram

   Entity-Relationship Diagram (ERD) illustrating relationships between tables.



3. Testing queries

   Test case for data insertion, updates, and selections.



```
DELETE FROM BLOOD_INVENTORY WHERE BloodInventory_id = 10;

SELECT d.Donor_id, d.Name, d.Blood_group, b.B_name AS Blood_Bank_Name, b.Location
FROM DONOR d
JOIN BLOOD_INVENTORY bi ON d.BloodInventory_id = bi.BloodInventory_id
JOIN BLOODBANK b ON bi.
```

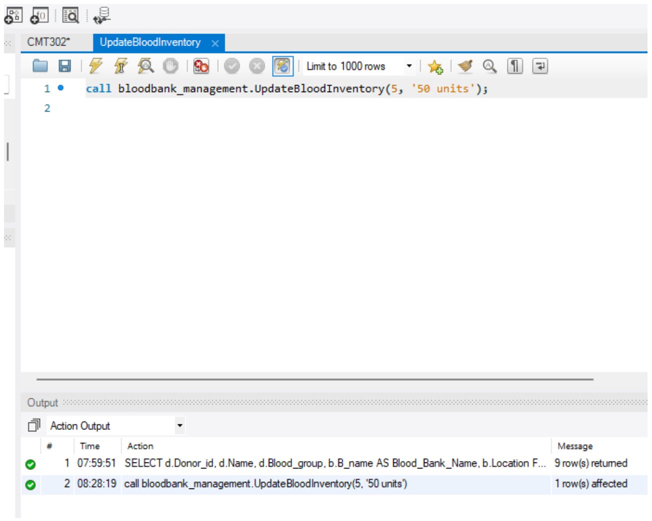
Call stored procedure bloodbank_management.UpdateBl...

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

p_BloodInventory_id	5	[IN]	INT
p_Quantity_available	50 units	[IN]	VARCHAR(50)

	Donor_id	Name	Blood_grou
▶	1	Alice Mwangi	A+
	2	John Kamau	O+
	3	Peter Otieno	B+
	4	Grace Wambui	AB+
	5	Mary Njeri	A-
	6	Paul Kiprop	B-

```
#  Time      Action                                                                          Message
1  07:59:51  SELECT d.Donor_id, d.Name, d.Blood_group, b.B_name AS Blood_Bank_Name, b.Location F...  9 row(s) returned
```



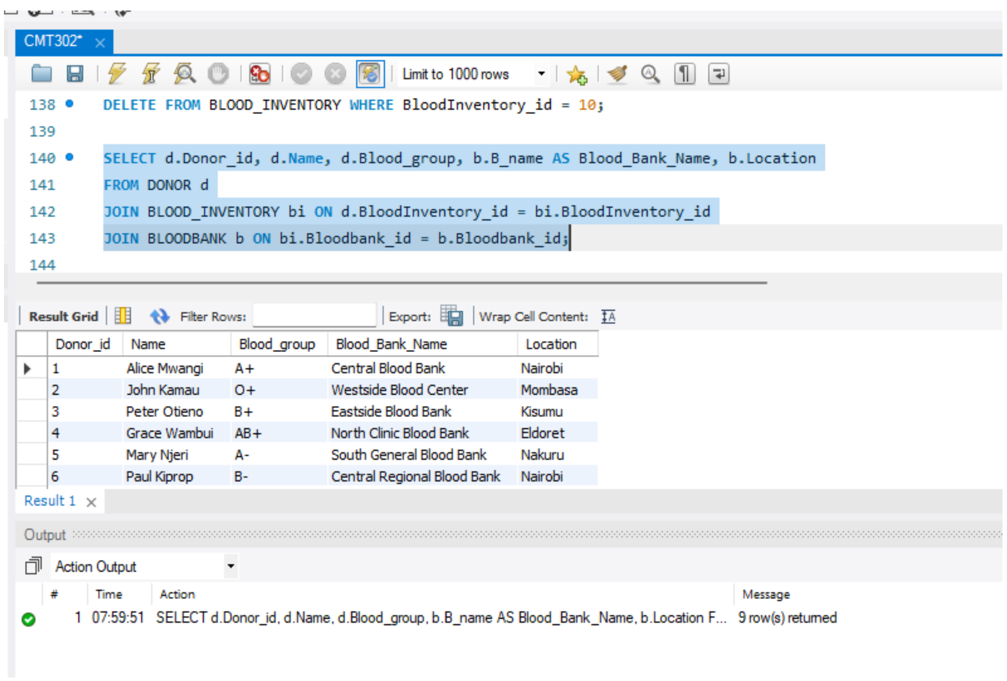
```
call bloodbank_management.UpdateBloodInventory(5, '50 units');
```

```
#  Time      Action                                                                          Message
1  07:59:51  SELECT d.Donor_id, d.Name, d.Blood_group, b.B_name AS Blood_Bank_Name, b.Location F...  9 row(s) returned
2  08:28:19  call bloodbank_management.UpdateBloodInventory(5, '50 units')                      1 row(s) affected
```

Complex queries demonstrating the relationship between tables and aggregated data.

4. Screenshots

Output of select queries for various tables.

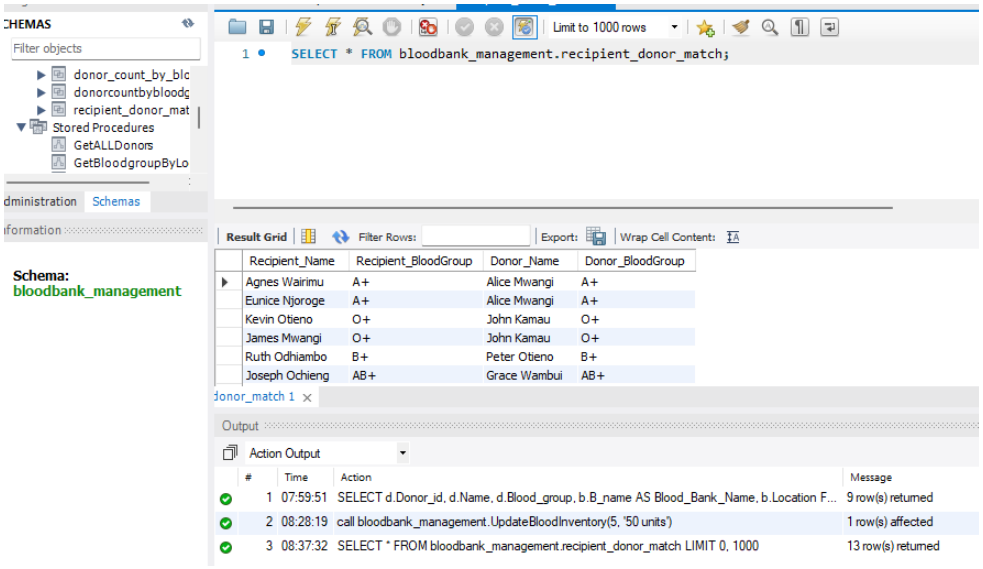


```sql
DELETE FROM BLOOD_INVENTORY WHERE BloodInventory_id = 10;

SELECT d.Donor_id, d.Name, d.Blood_group, b.B_name AS Blood_Bank_Name, b.Location
FROM DONOR d
JOIN BLOOD_INVENTORY bi ON d.BloodInventory_id = bi.BloodInventory_id
JOIN BLOODBANK b ON bi.Bloodbank_id = b.Bloodbank_id;
```

Donor_id	Name	Blood_group	Blood_Bank_Name	Location
1	Alice Mwangi	A+	Central Blood Bank	Nairobi
2	John Kamau	O+	Westside Blood Center	Mombasa
3	Peter Otieno	B+	Eastside Blood Bank	Kisumu
4	Grace Wambui	AB+	North Clinic Blood Bank	Eldoret
5	Mary Njeri	A-	South General Blood Bank	Nakuru
6	Paul Kiprop	B-	Central Regional Blood Bank	Nairobi

```
#  Time      Action                                                                      Message
1  07:59:51  SELECT d.Donor_id, d.Name, d.Blood_group, b.B_name AS Blood_Bank_Name, b.Location F...  9 row(s) returned
```

Results of joined tables showing donors linked to and recipients.



Schema:
bloodbank_management

```sql
SELECT * FROM bloodbank_management.recipient_donor_match;
```

Recipient_Name	Recipient_BloodGroup	Donor_Name	Donor_BloodGroup
Agnes Wairimu	A+	Alice Mwangi	A+
Eunice Njoroge	A+	Alice Mwangi	A+
Kevin Otieno	O+	John Kamau	O+
James Mwangi	O+	John Kamau	O+
Ruth Odhiambo	B+	Peter Otieno	B+
Joseph Ochieng	AB+	Grace Wambui	AB+

```
#  Time      Action                                                                      Message
1  07:59:51  SELECT d.Donor_id, d.Name, d.Blood_group, b.B_name AS Blood_Bank_Name, b.Location F...  9 row(s) returned
2  08:28:19  call bloodbank_management.UpdateBloodInventory(5, '50 units')                   1 row(s) affected
3  08:37:32  SELECT * FROM bloodbank_management.recipient_donor_match LIMIT 0, 1000          13 row(s) returned
```

IMPORTANT LINK

https://github.com/bakhita5/BLOOD-MANAGMENT-SYSTEM-DATABASE