BLOOD BANK MANAGEMENT SYSTEM

Mini Project Report
Database Lab (DSE 2241)
Department of Data Science &
Computer Applications

B. Tech Data Science

4th Semester Batch: B4 Group:B7

Submitted By:

MANDAR CHAUDHARI	**220968222**
NAMAN MAHESHWARI	**220968252**
SHIVLI MATHUR	**220968298**
BANDI RISHIK REDDY	**220968300**
DEVESH AHUJA	**220968308**
VIGNESH SURESH MENON	**220968312**

Mentored By

Vinayak M
Assistant Professor-Senior
DSCA, MIT

Archana H
Assistant Professor-Senior
DSCA, MIT

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Date: **06th March 2024**

CERTIFICATE

This is to certify that **Mandar Chaudhari (220968222), Naman Maheshwari (220968252), Shivli Mathur (220968298), Bandi Rishik Reddy (220968300), Devesh Ahuja (220968308), Vignesh Suresh Menon (220968312)**, have successfully executed a mini project titled “**BLOOD BANK MANAGEMENT SYSTEM**” rightly bringing fore the competencies and skill sets they have gained during the course- Database Lab (DSE 2241), thereby resulting in the culmination of this project.

Vinayak M	**Archana H**
Assistant Professor-Senior	**Assistant Professor-Senior**
DSCA, MIT	**DSCA, MIT**

ABSTRACT

The area of blood donation database management systems is of utmost importance in the present-day scenario. These systems play a crucial role in maintaining the inventory of blood donations and blood samples, ensuring transparency, and streamlining the process of obtaining blood from blood banks. With the constant and challenging task of managing an adequate donor pool to meet the increasing demand for blood products, digital transformation and efficient management systems have become essential. The objective of a project in this area would be to develop a robust and user-friendly system that facilitates donor registration, inventory management, and seamless coordination between blood donors and medical staff, ultimately contributing to the efficient and effective management of blood donations and supply.

The methodology adopted for the blood donation database management system involved the use of Oracle SQL, PL/SQL, and database management tools. The project focused on developing a robust and efficient database system to keep track of donors and manage blood inventory. The use of PL/SQL alongside basic SQL was aimed at enhancing the system's power and efficiency. The system was designed to provide quick access to donor records and monitor blood screening to enhance medical service delivery.

The development of blood donation database management systems has yielded significant results in efficiently maintaining the inventory of blood donations, blood samples, donor information, and blood products. These systems, such as the Blood Bank Management System (BBMS) have facilitated the tracking of information about donors, blood banks, and blood groups, streamlining the donation process, and attracting first-time and young donors. The impact of digital transformation on blood donation and donor characteristics has been notable, with the digitalization process changing the donor pool and benefiting medical staff through automated processes.

The development of a Blood Donation Database Management System (DMS) has led to several important conclusions. Firstly, the DMS has successfully addressed real-time issues related to blood donation and management, providing a robust platform for connecting blood donors and doctors, thereby ensuring that every patient can receive the required blood in a timely manner. Additionally, the DMS is a crucial tool for maintaining the inventory of blood donations and blood samples, which is essential for effective blood bank management. Furthermore, the DMS has simplified the process of blood donation, allowing donors to easily register and donate blood, while also enabling doctors to efficiently check the availability of specific blood groups and provide them to patients in need. Overall, the project has demonstrated the use of DMS in easing the blood donation process.

Contents

Chapter 1
Introduction

The area of blood bank management system is crucial for ensuring the efficient handling of blood donations and maintaining blood inventory. It involves the management of donor information, blood group details, and the coordination between donors, blood banks, and patients in need of blood. The blood bank management system aims to bridge the gap between blood donors and recipients, providing a platform for easy registration, donation, and retrieval of blood when needed.

The need for a change in the existing system is evident due to the increasing percentage of people donating blood and the importance of managing the received blood thoroughly to ensure the safety of the recipients. The current manual systems may struggle to keep up with the growing demand and the need for efficient management of blood donations. Therefore, the introduction of a well-designed and comprehensive blood bank management system is essential to address these challenges and ensure the smooth functioning of blood donation processes. In today's world, the need for a proper and robust platform where donors and doctors can get connected for blood donation is paramount. The blood bank management system aims to fulfill the gap between blood donors and doctors, providing an easy and suitable platform for donors to register and donate blood. This not only benefits the donors who wish to contribute but also makes it easier for many people who are willing to donate blood but cannot find a proper platform due to their busy schedules. The system allows unique identification assignment, and efficient storage of donor details, thereby simplifying the process of blood donation and ensuring that every patient can receive the required blood within a timely manner.

The proposed system aims to overcome the limitations of the existing manual systems by providing an error-free, secure, reliable, and fast management system. It is designed to store, process, retrieve, and analyze information related to blood donors, blood groups, and blood inventory within a blood bank. The system also aims to provide services to hospitals and other users 24/7, making it easier to manage donor information and facilitate blood donation and retrieval processes.

In summary, the blood bank management system plays a vital role in the healthcare industry by streamlining the blood donation processes. The need for a change from the existing manual systems is driven by the increasing demand for blood donations and the necessity for efficient and error-free management of blood inventory. The proposed system aims to address these challenges by providing a user-friendly, comprehensive, and secure platform for managing all aspects of blood donation.

Chapter 2
Synopsis

2.1 Proposed System

Blood Bank Management System is a software application that aims to interconnect all the donors and receivers into a single network, validation, store records securely, easy to search the required blood, and easy to manage the blood stock details. The proposed system is scalable, efficient, and adaptable to meet the complex needs of the blood bank, which is a key facilitator for the healthcare sector. The main aim of this system is to launch an interaction medium for the blood donation management, simplify and automate the process of searching for blood in case of emergency, and maintain the records of blood donors, recipients, and blood stocks in the bank. The purpose of the blood bank management system is to simplify and automate the process of searching for blood in case of emergency and maintain the records of blood donors, recipients, and blood stocks in the bank.

2.2 Objectives

Efficient Blood Inventory Management: To maintain accurate and up-to-date records of blood inventory, ensuring that blood is available for patients when needed.

Chapter 3

Functional Requirements

3.1 Donor Management Module

Donor Registration: Allows new donors to register by providing their personal information such as name, contact details, blood type, gender, and date of birth.

Input

1. Donor ID

Unique identifier for each donor

2. Name

Name of the donor

3. Contact Details

Contact details of the donor (phone number, email, etc.)

4. Blood Type

Blood type of the donor

5. Gender

Gender of the donor

6. Date of Birth

Date of birth of the donor

Processing

1. Verifies data/ checks availability.
2. Checks if the provided Donor ID is unique.
3. Ensure that all required fields (Name, Contact Details, Blood Type, Gender, Date of Birth) are filled out.

Output

1. If the donor registration is successful, display a message indicating successful registration.
2. If there are any errors (e.g., duplicate Donor ID, missing fields), display an error message indicating the issue and prompt the user to correct it.

Donor Profile Management: Enables donors to update their profile information, including contact details and medical history.

Input

1. Donor ID

Unique identifier for each donor

2. Contact Details

Updated contact details of the donor.

3. Medical History

Medical history of the donor

Processing

1. Checks if the provided Donor ID exists in the database.
2. Updates the contact details and medical history for the specified donor.

Output

1. If the profile update is successful, display a message indicating successful update.
2. If there are any errors (e.g., donor not found), display an error message indicating the issue and prompt the user to correct it.

Donation History: Tracks the donation history of each donor, including the date of donation, quantity donated, and blood type.

Input

1. Donation ID

Unique identifier for each donation

2. Donor ID

Identifier of the donor who made the donation.

3. Donation Date

Date of the donation

4. Quantity Donated

Quantity of blood donated (in liters)

5. Blood Type

Blood type donated.

Processing

1. Check if the provided Donor ID exists in the database.
2. Insert the provided donation information into the database.

Output

1. If the donation record is successfully added, display a message indicating successful donation recording.
2. If there are any errors (e.g., donor not found), display an error message indicating the issue and prompt the user to correct it.

3.2 Recipient Management Module

Recipient Registration: Allows recipients to register by providing their personal information such as name, contact details, blood type, gender, and date of birth.

Input

1. Recipient ID

Unique identifier for each recipient

2. Name

Name of the recipient

3. Contact Details

Contact details of the recipient (phone number, email, etc.)

4. Blood Type

Blood type of the recipient

5. Gender

Gender of the recipient

6. Date of Birth

Date of birth of the recipient

<u>Processing</u>

1. Check if the provided Recipient ID is unique.
2. Ensure that all required fields (Name, Contact Details, Blood Type, Gender, Date of Birth) are filled out.
3. Insert the provided recipient information into the database.

<u>Output</u>

1. If the recipient registration is successful, display a message indicating successful registration.
2. If there are any errors (e.g., duplicate Recipient ID, missing fields), display an error message indicating the issue and prompt the user to correct it.

<u>Recipient Profile Management</u>: Enables recipients to update their profile information, including contact details and medical history.

<u>Input</u>

1. Recipient ID

Unique identifier for each recipient

2. Contact Details

Updated contact details of the recipient

3. Medical History

Medical history of the recipient

<u>Processing</u>

1. Check if the provided Recipient ID exists in the database.
2. Update the contact details and medical history for the specified recipient.

<u>Output</u>

1. If the profile update is successful, display a message indicating successful update.
2. If there are any errors (e.g., recipient not found), display an error message indicating the issue and prompt the user to correct it.

3.3 Blood Inventory Management Module:

Blood Donation Recording Table: Records details of each blood donation, including donor information, donation date, blood type, and quantity donated.

Input

1. Donation ID

Unique identifier for each donation

2. Donor ID

Identifier of the donor who made the donation.

3. Donation Date

Date of the donation

4. Blood Type

Blood type donated.

5. Quantity Donated

Quantity of blood donated (in liters)

Processing

1. Check if the provided Donor ID exists in the database.
2. Insert the provided donation information into the database.

Output

1. If the donation record is successfully added, display a message indicating successful donation recording.
2. If there are any errors (e.g., donor not found), display an error message indicating the issue and prompt the user to correct it.

Inventory Tracking Table: Tracks the quantity of each blood type available in the inventory and updates it based on donations and usage, manages the storage locations of blood units.

Input

1. Inventory ID

Unique identifier for each inventory entry

2. Blood Type

Blood type

3. Quantity

Quantity of blood available (in liters)

4. Storage Location

Location where the blood is stored.

Processing

1. Check if the provided Donation ID is unique.
2. Ensure that all required fields (Blood Type, Quantity, Storage Location) are filled out.
3. Insert the provided inventory information into the database.

Output

1. If the inventory entry is successfully added, display a message indicating successful entry.

2. If there are any errors (e.g., duplicate Inventory ID, missing fields), display an error message indicating the issue and prompt the user to correct it.

3.4 Blood Request Processing Module:

Blood Request Management: Receives and processes blood requests submitted by recipients, ensuring timely fulfilment based on availability and compatibility, tracks the status of each blood request from submission to fulfilment.

Input

1. Request ID

Unique identifier for each blood request

2. RecipientID

Identifier of the recipient making the request

3. Blood Type

Requested blood type.

4. Request Date

Date of the request

5. Status

Current Status of the request

Processing

1. Check if the provided RecipientID exists in the database.
2. Insert the provided blood request information into the database.

Output

1. If the blood request is successfully processed, display a message indicating successful processing.
2. If there are any errors (e.g., recipient not found), display an error message indicating the issue and prompt the user to correct it.

3.5 Blood Testing Module:

Blood Sample Testing: Conducts screening tests on donated blood samples to ensure they meet safety and quality standards, records screening results for each blood sample, indicating whether it meets the required criteria for transfusion.

Input

1. Screening ID

Unique identifier for each screening

2. Donation ID

Identifier of the donation being screened.

3. Screening Date

Date of the screening

4. Screening Result

Result of the screening test

Processing

1. Check if the provided Donation ID exists in the database.
2. Update the screening result for the specified blood sample.

Output

1. If the screening result is successfully recorded, display a message indicating successful recording.
2. If there are any errors (e.g., donation not found), display an error message indicating the issue and prompt the user to correct it.

3.6 Reporting Module:

Gives an overview of any donation, request, recipient, inventory, testing information.

Input

1. Report ID

Unique identifier for each report

2. Report Name

Name of the report

3. Report Type

Type of the report (donor, recipient, etc.)

4. Report Data

Data included in the report.

Processing

1. Retrieve data from the database based on the specified report parameters.
2. Generate the report based on the processed data.

Output

1. Provide the generated report to the user for viewing and analysis.
2. If there are any errors (e.g., report generation failure), display an error message indicating the issue and prompt the user to try again.

Chapter 4
Detailed Design

4.1 ER Diagram

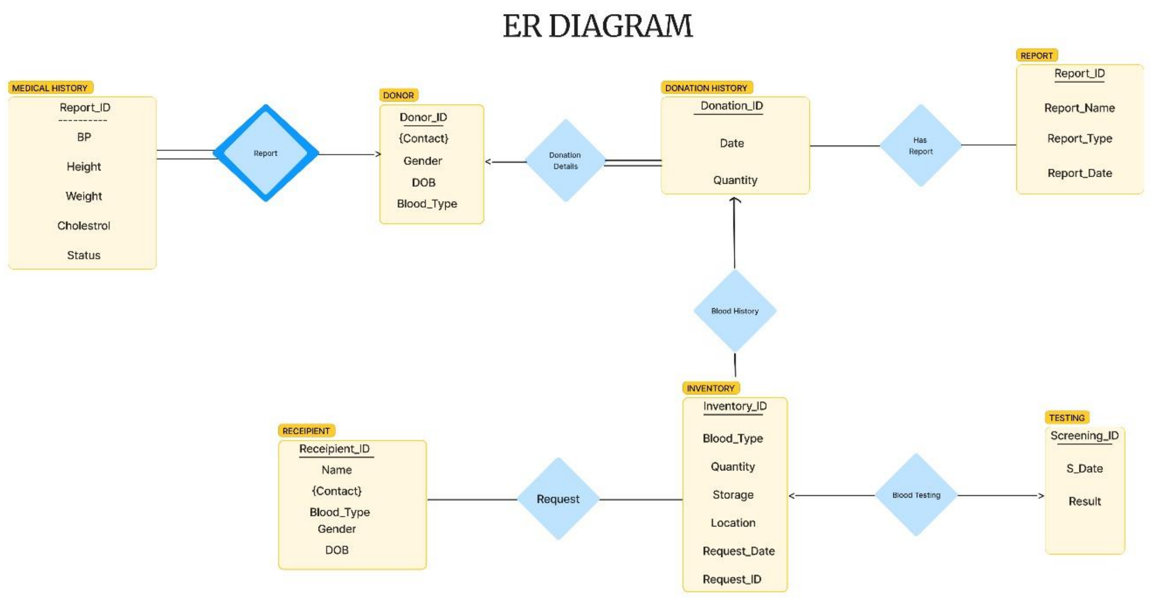


Figure 4.1 ER Diagram

4.2 Schema Diagram

Relational Schema

Donor (donor_id, gender, dob, blood_type)

Medical_History (report_id, donor _id, bp, height, weight, cholesterol, status)

Donation_History (donation_id, date, quantity, donor_id)

Report (report_id,report_name, report_type, report_date)

Has_Report(donation_id,report_id**)**

Receipient(Receipient_id, name, blood_type, gender, dob)

Inventory (inventory_id, donation_id, blood_type, quantity, storage, location, request_date, request_id)

Request(Receipient_id, inventory_id)

Testing (screening_id, inventory_id, s_date, result)

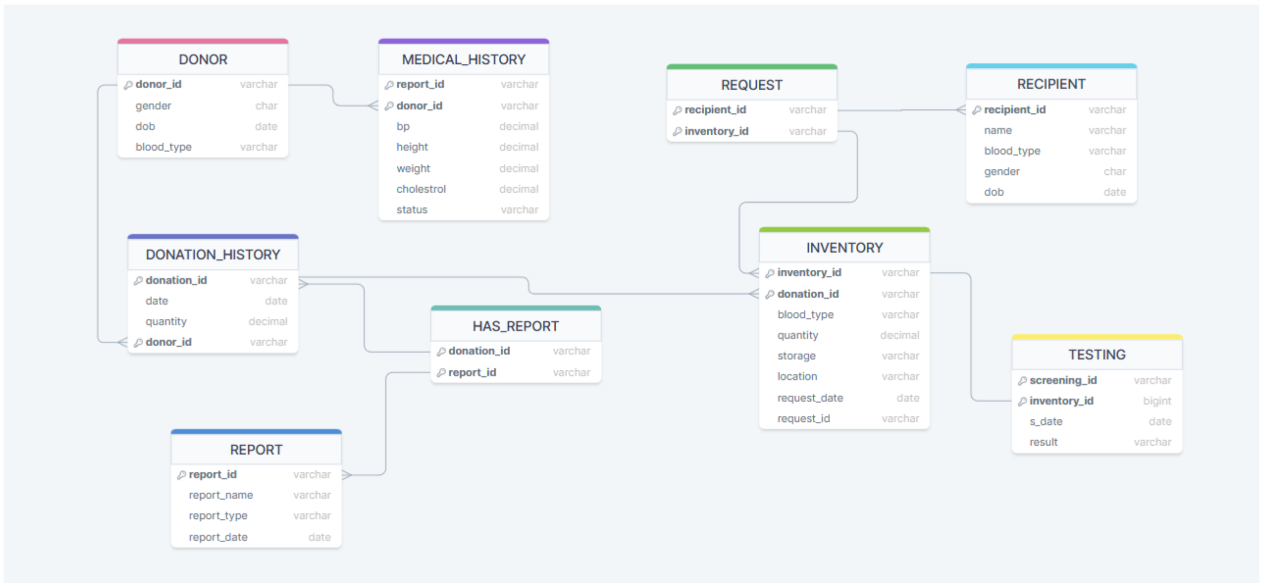


Figure 4.2 Schema Diagram

4.3 Data Dictionary

Donor:

1. donor_id:
   - Datatype: VARCHAR(10)
   - Constraint: Primary Key
   - Constraint Name: PK_Donor
2. gender:
   - Datatype: CHAR(1)
3. dob:
   - Datatype: DATE

4. blood_type:
   - Datatype: VARCHAR(5)

Medical_History:

1. report_id:
   - Datatype: VARCHAR(10)
   - Constraint: Primary Key
   - Constraint Name: PK_Report_Medical_History

2. donor_id:
   - Datatype: VARCHAR(10)
   - Constraint: Foreign Key (References Donor)
   - Constraint Name: FK_Donor_Medical_History

3. bp:

- Datatype: VARCHAR(10)

4. height:
   - Datatype: NUMBER(5,2)

5. weight:
   - Datatype: NUMBER(5,2)

6. cholesterol:
   - Datatype: NUMBER(5,5)

7. status:
   - Datatype: VARCHAR(50)

Donation_History:

1. donation_id:
   - Datatype: VARCHAR(10)
   - Constraint: Primary Key
   - Constraint Name: PK_Donation_Donation_History

2. date:
   - Datatype: DATE
3. quantity:
   - Datatype: NUMBER(5,2)
4. donor_id:
   - Datatype: VARCHAR(10)
   - Constraint: Foreign Key (References Donor)
   - Constraint Name: FK_Donor_Donation_History

Report:

1. report_id:
   - Datatype: VARCHAR(10)
   - Constraint: Primary Key
   - Constraint Name: PK_Report_Report

2. report_info:
   - Datatype: VARCHAR(100)

3. report_type:
   - Datatype: VARCHAR(20)

4 report_date:
   - Datatype: DATE

Has_Report:

1. donation_id:
   - Datatype: VARCHAR(10)
   - Constraint: Foreign Key (References Donation_History)
   - Constraint Name: FK_Donation_Has_Report

2. report_id:
   - Datatype: VARCHAR(10)
   - Constraint: Foreign Key (References Report)
   - Constraint Name: FK_Report_Has_Report

Recipient:

1. recipient_id:
   - Datatype: VARCHAR(10)
   - Constraint: Primary Key
   - Constraint Name: PK_Recipient

2. name:
   - Datatype: VARCHAR(50)

3. blood_type:
   - Datatype: VARCHAR(5)

4. gender:
   - Datatype: CHAR(1)

5. dob:
   - Datatype: DATE

Inventory:

1. inventory_id:
   - Datatype: VARCHAR(10)
   - Constraint: Primary Key
   - Constraint Name: PK_Inventory_Inventory

2. donation_id:
   - Datatype: VARCHAR(10)
   - Constraint: Foreign Key (References Donation_History)
   - Constraint Name: FK_Donation_Inventory

3. blood_type:
   - Datatype: VARCHAR(5)

4. quantity:
   - Datatype: NUMBER(5,2)

5. storage:
   - Datatype: VARCHAR(50)

6. location:
   - Datatype: VARCHAR(50)

7. request_date:
   - Datatype: DATE

8. request_id:
   - Datatype: VARCHAR(10)

<u>Request</u>:
1. recipient_id:
    - Datatype: VARCHAR(10)
    - Constraint: Foreign Key (References Recipient)
    - Constraint Name: FK_Recipient_Request

2. inventory_id:
    - Datatype: VARCHAR(10)
    - Constraint: Foreign Key (References Inventory)
    - Constraint Name: FK_Inventory_Request

<u>Testing</u>:
1. screening_id:
    - Datatype: VARCHAR(10)
    - Constraint: Primary Key
    - Constraint Name: PK_Screening_Testing

2. inventory_id:
    - Datatype: VARCHAR(10)
    - Constraint: Foreign Key (References Inventory)
    - Constraint Name: FK_Inventory_Testing

3. screening_date:
    - Datatype: DATE

4. result:
    - Datatype: VARCHAR(20)

4.4 Relational Model Implementation

```
-- Donor table
CREATE TABLE Donor (
   donor_id VARCHAR(10),
   gender CHAR(1),
   dob DATE,
   blood_type VARCHAR(5),
   CONSTRAINT PK_Donor PRIMARY KEY (donor_id)
);

-- Medical_History table
CREATE TABLE Medical_History (
   report_id VARCHAR(10),
   donor_id VARCHAR(10),
   bp VARCHAR(10),
   height NUMBER(5,2),
   weight NUMBER(5,2),
   cholesterol NUMBER(5,2),
   status VARCHAR(50),
   CONSTRAINT PK_Report_Medical_History PRIMARY KEY (report_id),
   CONSTRAINT FK_Donor_Medical_History FOREIGN KEY (donor_id) REFERENCES
Donor(donor_id)
);

-- Donation_History table
CREATE TABLE Donation_History (
   donation_id VARCHAR(10),
   donation_date DATE,
   quantity NUMBER(5,2),
   donor_id VARCHAR(10),
   CONSTRAINT PK_Donation_Donation_History PRIMARY KEY (donation_id),
   CONSTRAINT FK_Donor_Donation_History FOREIGN KEY (donor_id) REFERENCES
Donor(donor_id)
);

-- Report table
CREATE TABLE Report (
   report_id VARCHAR(10),
   report_info VARCHAR(100),
   report_type VARCHAR(20),
   report_date DATE,
   CONSTRAINT PK_Report_Report PRIMARY KEY (report_id)
);

-- Has_Report table
CREATE TABLE Has_Report (
   donation_id VARCHAR(10),
   report_id VARCHAR(10),
   CONSTRAINT FK_Donation_Has_Report FOREIGN KEY (donation_id) REFERENCES
Donation_History(donation_id),
   CONSTRAINT FK_Report_Has_Report FOREIGN KEY (report_id) REFERENCES
Report(report_id)
);

-- Recipient table
CREATE TABLE Recipient (
   recipient_id VARCHAR(10),
   name VARCHAR(50),
   blood_type VARCHAR(5),
   gender CHAR(1),
   dob DATE,
   CONSTRAINT PK_Recipient PRIMARY KEY (recipient_id)
);

-- Inventory table
CREATE TABLE Inventory (
   inventory_id VARCHAR(10),
   donation_id VARCHAR(10),
   blood_type VARCHAR(5),
   quantity NUMBER(5,2),
   storage VARCHAR(50),
   location VARCHAR(50),
   request_date DATE,
   request_id VARCHAR(10),
   CONSTRAINT PK_Inventory_Inventory PRIMARY KEY (inventory_id),
   CONSTRAINT FK_Donation_Inventory FOREIGN KEY (donation_id) REFERENCES
Donation_History(donation_id)
);

-- Request table
CREATE TABLE Request (
   recipient_id VARCHAR(10),
   inventory_id VARCHAR(10),
   CONSTRAINT FK_Recipient_Request FOREIGN KEY (recipient_id) REFERENCES
Recipient(recipient_id),
   CONSTRAINT FK_Inventory_Request FOREIGN KEY (inventory_id) REFERENCES
Inventory(inventory_id)
);

-- Testing table
CREATE TABLE Testing (
   screening_id VARCHAR(10),
   inventory_id VARCHAR(10),
   screening_date DATE,
   result VARCHAR(20),
   CONSTRAINT PK_Screening_Testing PRIMARY KEY (screening_id),
   CONSTRAINT FK_Inventory_Testing FOREIGN KEY (inventory_id) REFERENCES
Inventory(inventory_id)
);
```

5. Implementation

5.1 Queries

```
--1)Retrieve Donors with Blood Type A+:
SELECT * FROM Donor WHERE blood_type = 'A+';

--2)Retrieve Medical History for a Specific Donor:
SELECT * FROM Medical_History WHERE donor_id = 'D1004';

--3)Update Cholesterol Level for a Report:
UPDATE Medical_History SET cholesterol = 5.2 WHERE report_id = 'R1004';

--4)Retrieve Donations Made by Donor 'D1001':
SELECT * FROM Donation_History WHERE donor_id = 'D1001';

--5)Retrieve Reports of info 'Blood Test Report':
SELECT * FROM Report WHERE report_info = 'Blood Test Report';

--6)Retrieve Donations with Associated Reports:
SELECT * FROM Has_Report;

--7)Retrieve Recipient with ID 'RC1001':
SELECT * FROM Recipient WHERE recipient_id = 'RC1001';

--8)Update Gender of Recipient 'RC1001' to 'F':
UPDATE Recipient SET gender = 'F' WHERE recipient_id = 'RC1001';

--9)Retrieve Inventory Items with Blood Type 'AB-' and Quantity > 10:
SELECT * FROM Inventory WHERE blood_type = 'AB-' AND quantity > 10;

--10)Retrieve Requests Made by Recipient 'RC1001':
SELECT * FROM Request WHERE recipient_id = 'RC1001';
```

-- 11)Select the expired inventory items

```
SELECT * FROM Inventory
WHERE request_date < (CURRENT_DATE - INTERVAL '56' DAY);
```

--12)Select rows from the Testing table that correspond to inventory items in the Inventory table where the request_date is older than 56 days from the current date.

```
 SELECT *
FROM Testing
WHERE inventory_id IN (
    SELECT inventory_id
    FROM Inventory
    WHERE request_date < (CURRENT_DATE - INTERVAL '56' DAY));
```

--13)Delete rows from the request table that correspond to inventory items in the Inventory table where the request_date is older than 56 days from the current date.

```
DELETE FROM request
WHERE inventory_id IN (
    SELECT inventory_id
    FROM Inventory
    WHERE request_date < (CURRENT_DATE - INTERVAL '56' DAY));
```

--14)Delete rows from the testing table that correspond to inventory items in the Inventory table where the request_date is older than 56 days from the current date.

```
DELETE FROM Testing
WHERE inventory_id IN (
    SELECT inventory_id
    FROM Inventory
    WHERE request_date < (CURRENT_DATE - INTERVAL '56' DAY));
```

--15)Delete the expired records from the Inventory table.

```
DELETE FROM Inventory
WHERE request_date < (CURRENT_DATE - INTERVAL '56' DAY);
```

--16)Retrieve Donors Who Donated the Maximum Quantity of Blood:

```
SELECT d.donor_id, d.gender, d.dob, d.blood_type, dh.quantity
FROM Donor d
JOIN Donation_History dh ON d.donor_id = dh.donor_id
WHERE dh.quantity = (SELECT MAX(quantity) FROM Donation_History);
```

--17)Calculate Average Cholesterol Level of Male and Female Donors:

```
SELECT d.gender, AVG(mh.cholesterol) AS avg_cholesterol
FROM Donor d
JOIN Medical_History mh ON d.donor_id = mh.donor_id
GROUP BY d.gender;
```

--18)List Recipients Who Haven't Received Any Blood Donation:

```
SELECT r.recipient_id, r.name, r.blood_type
FROM Recipient r
LEFT JOIN Request rq ON r.recipient_id = rq.recipient_id
WHERE rq.inventory_id IS NULL;
```

--19)Find the Total Quantity of Blood Donated by Each Blood Type:

```
SELECT d.blood_type, SUM(dh.quantity) AS total_donation_quantity
FROM Donor d
JOIN Donation_History dh ON d.donor_id = dh.donor_id
GROUP BY d.blood_type;
```

--20)Calculate the Total Quantity of Blood in Inventory for Each Blood Type:

```
SELECT blood_type, SUM(quantity) AS total_quantity
FROM Inventory
GROUP BY blood_type;
```

--<u>21)Find Donors Who Haven't Donated Blood in the Last 6 Months:</u>

```
SELECT d.donor_id, d.gender, d.dob, d.blood_type
FROM Donor d
WHERE NOT EXISTS (
   SELECT 1
   FROM Donation_History dh
   WHERE dh.donor_id = d.donor_id
   AND dh.donation_date >= (CURRENT_DATE - INTERVAL '6' MONTH))
```

5.2 Triggers

Trigger-1

```
CREATE OR REPLACE TRIGGER update_inventory_on_donation_delete
AFTER DELETE ON Donation_History
FOR EACH ROW
DECLARE
   inventory_quantity NUMBER(5,2);
BEGIN
   SELECT quantity INTO inventory_quantity
   FROM Inventory
   WHERE donation_id = :OLD.donation_id;
   UPDATE Inventory
   SET quantity = quantity - :OLD.quantity
   WHERE donation_id = :OLD.donation_id;
   IF inventory_quantity = :OLD.quantity THEN
      DELETE FROM Inventory
      WHERE donation_id = :OLD.donation_id;
   END IF;
END;
/
```

Trigger-2

```
CREATE TRIGGER update_medical_history_status
AFTER INSERT ON Report
FOR EACH ROW
BEGIN
   UPDATE Medical_History
   SET status = 'Report Added'
   WHERE donor_id = (SELECT donor_id FROM Donation_History WHERE donation_id =
(SELECT donation_id FROM Has_Report WHERE report_id = :NEW.report_id));
END; /
```

5.3 Stored Procedures

Procedure-1

```
CREATE OR REPLACE PROCEDURE add_new_donor(
    p_donor_id VARCHAR2,
    p_gender VARCHAR2,
    p_dob_str VARCHAR2,
    p_blood_type VARCHAR2
)
IS
BEGIN
    -- Convert the string to a DATE data type
    INSERT INTO Donor (donor_id, gender, dob, blood_type)
    VALUES (p_donor_id, p_gender, TO_DATE(p_dob_str, 'YYYY-MM-DD'), p_blood_type);
END;
/
```

Procedure-2

```
CREATE OR REPLACE PROCEDURE CheckBloodAvailability (
  p_blood_type VARCHAR2
)
IS
  available_quantity NUMBER;
BEGIN
  SELECT SUM(quantity) INTO available_quantity
  FROM Inventory
  WHERE blood_type = p_blood_type
  AND request_id IS NULL;
  IF available_quantity > 0 THEN
    DBMS_OUTPUT.PUT_LINE('Sufficient blood of type ' || p_blood_type || ' available.');
  ELSE
```

```
      DBMS_OUTPUT.PUT_LINE('Insufficient blood of type ' || p_blood_type || ' available.');
    END IF;
  END;
  /
```

5.4 Stored Functions

<u>1. Function for CalculatinBMI and checking donor's status</u>

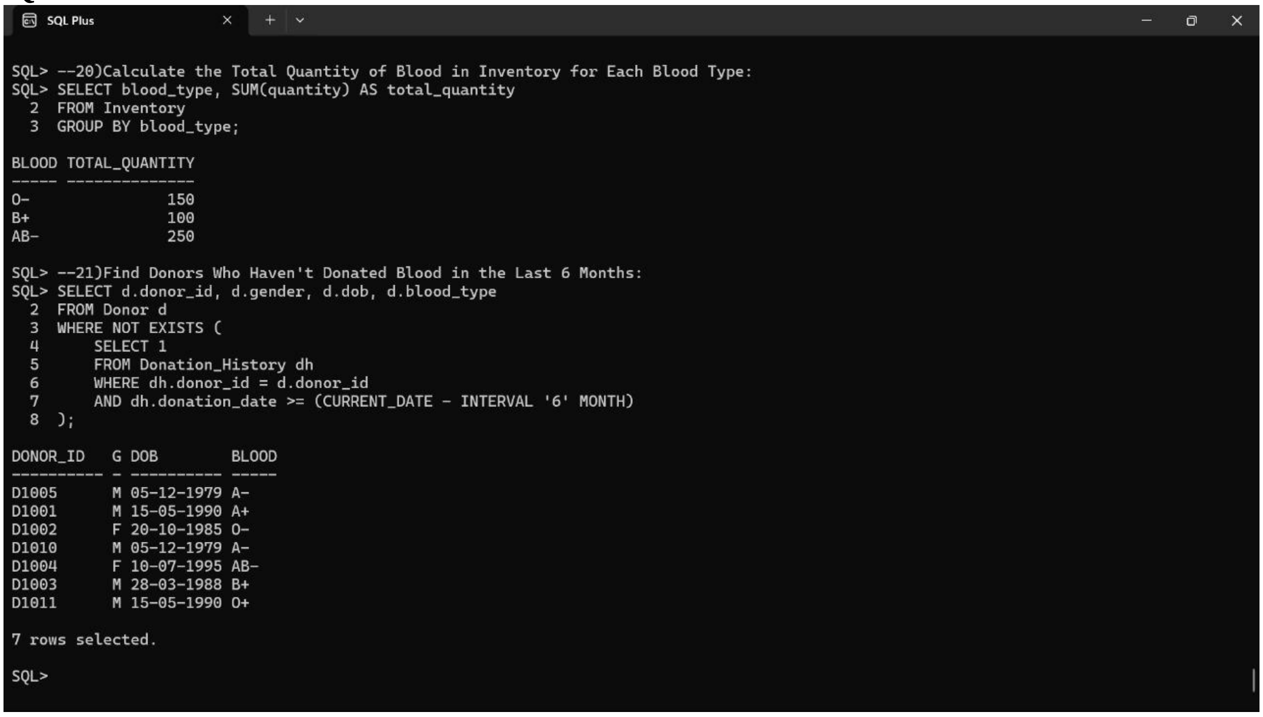
```
CREATE OR REPLACE FUNCTION CalculateBMIAndStatus
  (
    p_donor_id DONOR.DONOR_ID%TYPE
  )
RETURN VARCHAR2
AS
  v_height   NUMBER;
  v_weight   NUMBER;
  v_bmi      NUMBER;
  v_status   VARCHAR2(20);
BEGIN
  SELECT height, weight
    INTO v_height, v_weight
    FROM Medical_History
   WHERE donor_id = p_donor_id
   ORDER BY report_id DESC
   FETCH FIRST 1 ROW ONLY;

  v_height := v_height / 100; -- Convert height from centimeters to meters
  v_bmi := v_weight / (v_height * v_height);

  IF v_bmi < 18.5 THEN
    v_status := 'Underweight';
  ELSIF v_bmi >= 18.5 AND v_bmi < 25 THEN
    v_status := 'Healthy';
  ELSIF v_bmi >= 25 AND v_bmi < 30 THEN
    v_status := 'Overweight';
  ELSE
    v_status := 'Obese';
  END IF;

  RETURN 'BMI: ' || ROUND(v_bmi, 2) || ', Status: ' || v_status;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN 'No medical history found';
END;
/
```

2. Blood Compatibility Function

```
CREATE OR REPLACE FUNCTION CheckBloodCompatibility
  (
    p_donor_blood_type   VARCHAR2,
    p_recipient_blood_type VARCHAR2
  )
RETURN VARCHAR2
AS
  v_compatible VARCHAR2(20);
BEGIN
  IF p_donor_blood_type = 'O-' THEN
    v_compatible := 'Compatible';
  ELSIF
        p_donor_blood_type = 'O+' AND
        (p_recipient_blood_type LIKE 'O%' OR
         p_recipient_blood_type LIKE 'A%' OR
         p_recipient_blood_type LIKE 'B%' OR
         p_recipient_blood_type LIKE 'AB%') THEN
         v_compatible := 'Compatible';
  ELSIF
         p_donor_blood_type = 'A-' AND
        (p_recipient_blood_type = 'A+' OR
         p_recipient_blood_type = 'AB+' OR
         p_recipient_blood_type = 'A-' OR
         p_recipient_blood_type = 'AB-') THEN
         v_compatible := 'Compatible';
  ELSIF
         p_donor_blood_type = 'A+' AND
        (p_recipient_blood_type LIKE 'A%' OR
         p_recipient_blood_type LIKE 'AB%') THEN
         v_compatible := 'Compatible';
  ELSIF
         p_donor_blood_type = 'B-' AND
        (p_recipient_blood_type = 'B+' OR
         p_recipient_blood_type = 'AB+' OR
         p_recipient_blood_type = 'B-' OR
         p_recipient_blood_type = 'AB-') THEN
         v_compatible := 'Compatible';
  ELSIF
         p_donor_blood_type = 'B+' AND
        (p_recipient_blood_type LIKE 'B%' OR
         p_recipient_blood_type LIKE 'AB%') THEN
         v_compatible := 'Compatible';
  ELSIF
        p_donor_blood_type = 'AB-' AND
        (p_recipient_blood_type = 'AB+' OR
         p_recipient_blood_type = 'AB-') THEN
         v_compatible := 'Compatible';
  ELSIF
        p_donor_blood_type = 'AB+' AND
        p_recipient_blood_type = 'AB+' THEN
        v_compatible := 'Compatible';
  ELSE
    v_compatible := 'Incompatible';
  END IF;
  RETURN v_compatible;
END;
/
```

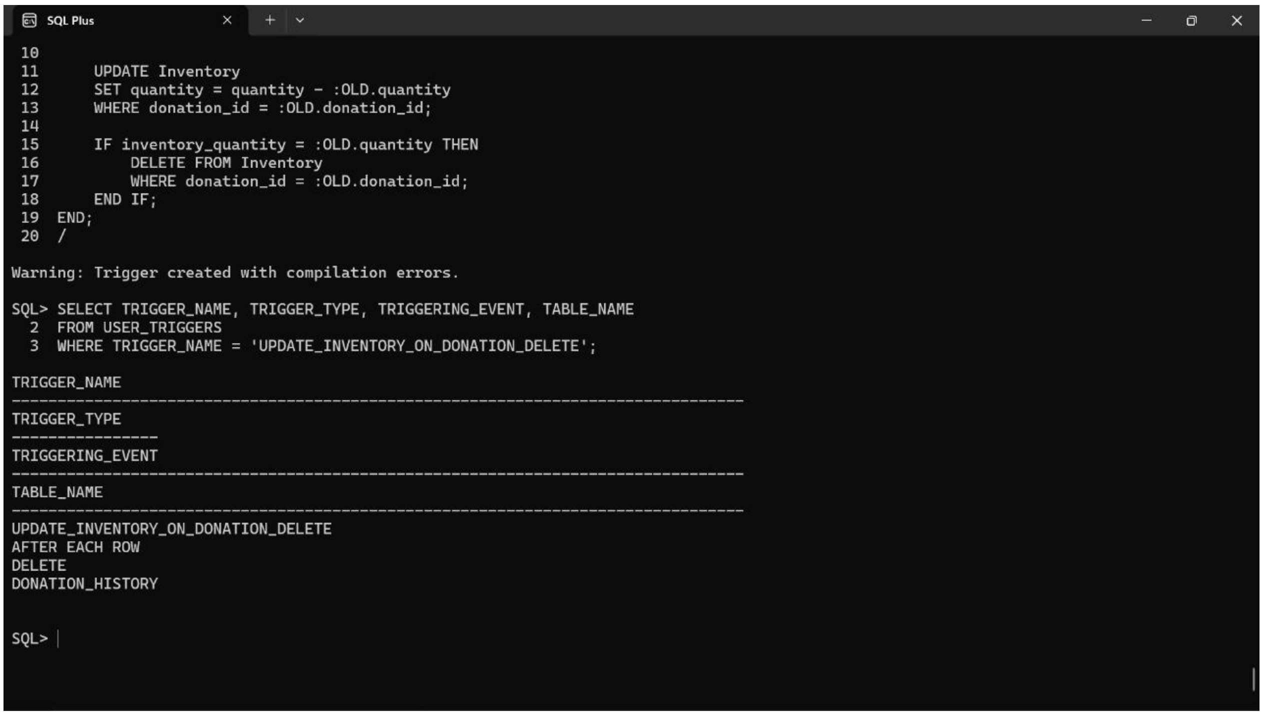
6. Result

QUERIES



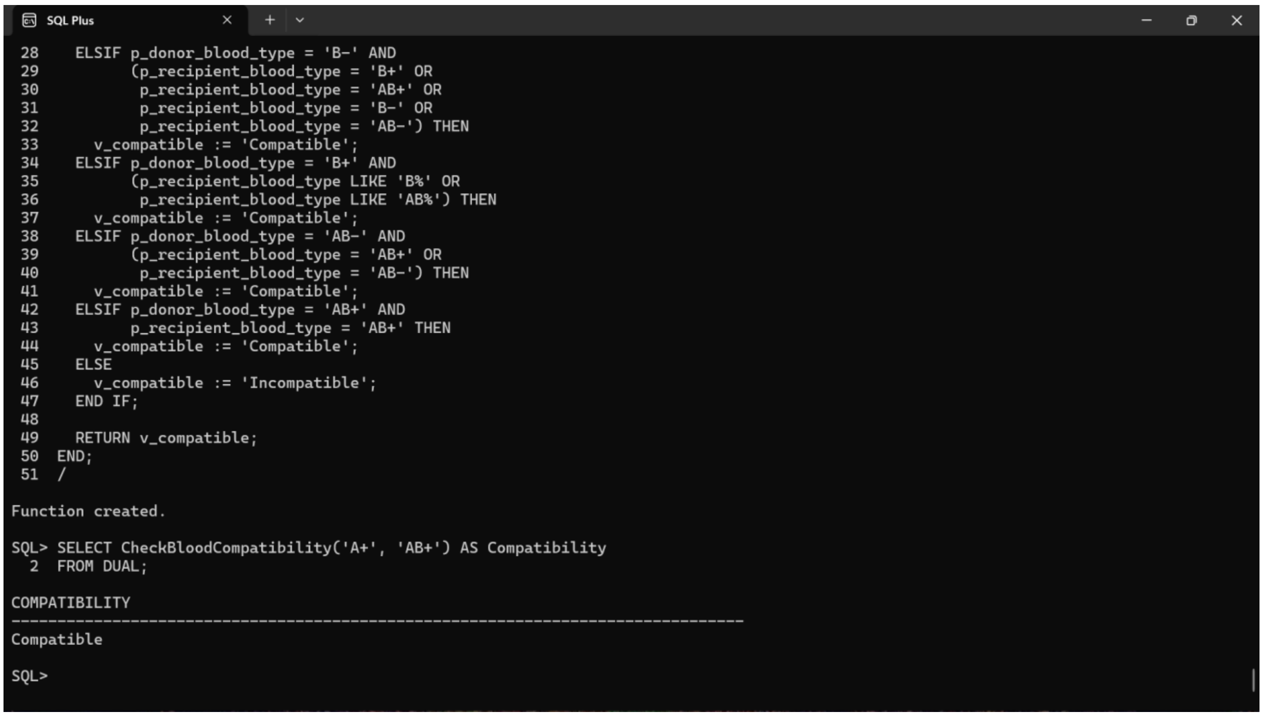
```
SQL> --20)Calculate the Total Quantity of Blood in Inventory for Each Blood Type:
SQL> SELECT blood_type, SUM(quantity) AS total_quantity
  2  FROM Inventory
  3  GROUP BY blood_type;

BLOOD TOTAL_QUANTITY
----- --------------
O-               150
B+               100
AB-              250

SQL> --21)Find Donors Who Haven't Donated Blood in the Last 6 Months:
SQL> SELECT d.donor_id, d.gender, d.dob, d.blood_type
  2  FROM Donor d
  3  WHERE NOT EXISTS (
  4      SELECT 1
  5      FROM Donation_History dh
  6      WHERE dh.donor_id = d.donor_id
  7      AND dh.donation_date >= (CURRENT_DATE - INTERVAL '6' MONTH)
  8  );

DONOR_ID   G DOB        BLOOD
---------- - ---------- -----
D1005      M 05-12-1979 A-
D1001      M 15-05-1990 A+
D1002      F 20-10-1985 O-
D1010      M 05-12-1979 A-
D1004      F 10-07-1995 AB-
D1003      M 28-03-1988 B+
D1011      M 15-05-1990 O+

7 rows selected.

SQL>
```

TRIGGER



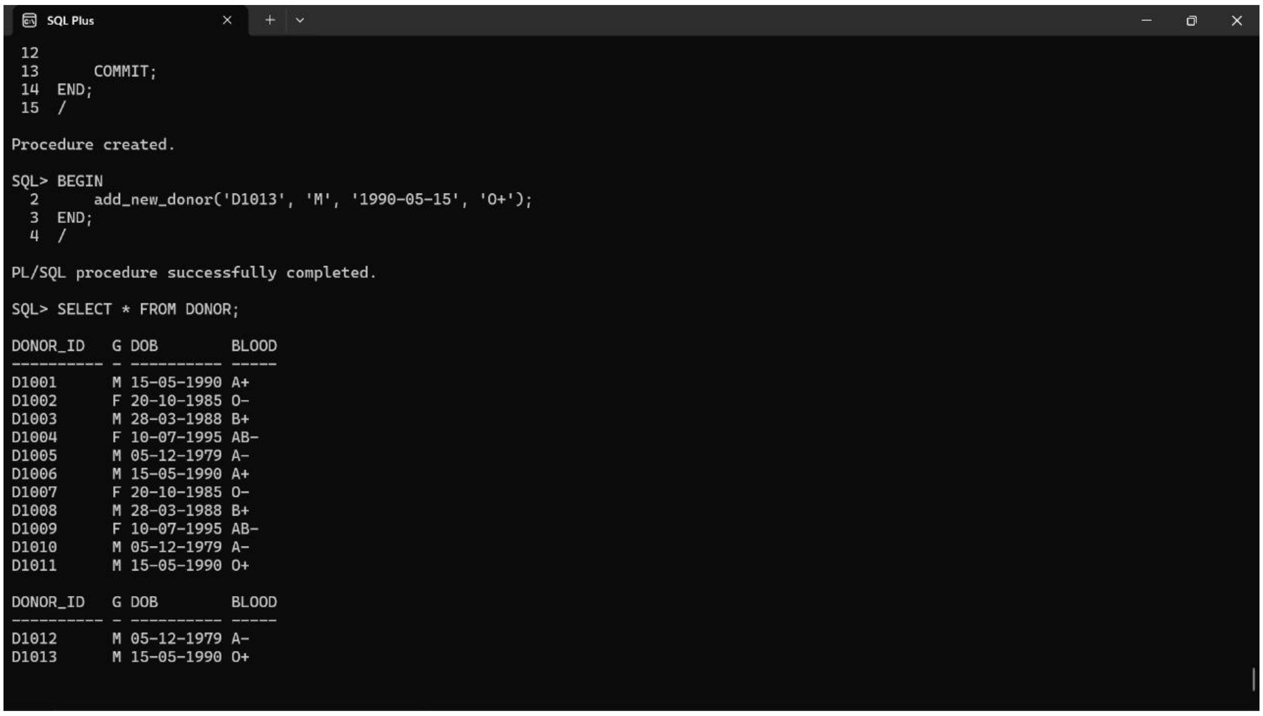
```
 10
 11      UPDATE Inventory
 12      SET quantity = quantity - :OLD.quantity
 13      WHERE donation_id = :OLD.donation_id;
 14
 15      IF inventory_quantity = :OLD.quantity THEN
 16          DELETE FROM Inventory
 17          WHERE donation_id = :OLD.donation_id;
 18      END IF;
 19  END;
 20  /

Warning: Trigger created with compilation errors.

SQL> SELECT TRIGGER_NAME, TRIGGER_TYPE, TRIGGERING_EVENT, TABLE_NAME
  2  FROM USER_TRIGGERS
  3  WHERE TRIGGER_NAME = 'UPDATE_INVENTORY_ON_DONATION_DELETE';

TRIGGER_NAME
--------------------------------------------------------------------------------
TRIGGER_TYPE
----------------
TRIGGERING_EVENT
--------------------------------------------------------------------------------
TABLE_NAME
--------------------------------------------------------------------------------
UPDATE_INVENTORY_ON_DONATION_DELETE
AFTER EACH ROW
DELETE
DONATION_HISTORY


SQL>
```

FUNCTION



```
  ELSIF p_donor_blood_type = 'B-' AND
        (p_recipient_blood_type = 'B+' OR
         p_recipient_blood_type = 'AB+' OR
         p_recipient_blood_type = 'B-' OR
         p_recipient_blood_type = 'AB-') THEN
    v_compatible := 'Compatible';
  ELSIF p_donor_blood_type = 'B+' AND
        (p_recipient_blood_type LIKE 'B%' OR
         p_recipient_blood_type LIKE 'AB%') THEN
    v_compatible := 'Compatible';
  ELSIF p_donor_blood_type = 'AB-' AND
        (p_recipient_blood_type = 'AB+' OR
         p_recipient_blood_type = 'AB-') THEN
    v_compatible := 'Compatible';
  ELSIF p_donor_blood_type = 'AB+' AND
        p_recipient_blood_type = 'AB+' THEN
    v_compatible := 'Compatible';
  ELSE
    v_compatible := 'Incompatible';
  END IF;

  RETURN v_compatible;
END;
/

Function created.

SQL> SELECT CheckBloodCompatibility('A+', 'AB+') AS Compatibility
FROM DUAL;

COMPATIBILITY
--------------------------------------------------------------------------------
Compatible

SQL>
```

PROCEDURE



```

    COMMIT;
END;
/

Procedure created.

SQL> BEGIN
    add_new_donor('D1013', 'M', '1990-05-15', 'O+');
END;
/

PL/SQL procedure successfully completed.

SQL> SELECT * FROM DONOR;

DONOR_ID   G DOB        BLOOD
---------- - ---------- -----
D1001      M 15-05-1990 A+
D1002      F 20-10-1985 O-
D1003      M 28-03-1988 B+
D1004      F 10-07-1995 AB-
D1005      M 05-12-1979 A-
D1006      M 15-05-1990 A+
D1007      F 20-10-1985 O-
D1008      M 28-03-1988 B+
D1009      F 10-07-1995 AB-
D1010      M 05-12-1979 A-
D1011      M 15-05-1990 O+

DONOR_ID   G DOB        BLOOD
---------- - ---------- -----
D1012      M 05-12-1979 A-
D1013      M 15-05-1990 O+
```

7. Conclusion and Future Work

7.1 Conclusion

The Blood Bank Management System represents a significant advancement in the healthcare sector, particularly in the management of blood donations and inventory. By providing a centralized platform for donors, recipients, and blood bank staff, it streamlines processes, enhances efficiency, and ensures the availability of blood when needed.

Key benefits of the system include:

- Efficiency: Automation of processes such as donor registration, inventory management, and blood request processing reduces manual efforts and minimizes errors.

- Accuracy: Centralized storage of donor, recipient, and inventory data ensures accuracy and consistency in records, leading to improved patient safety and care.

- Timeliness: Timely responses to blood requests, facilitated by real-time inventory tracking and automated notifications, ensure that patients receive the required blood promptly, potentially saving lives.

- Accessibility: The system provides easy access to information for both donors and recipients, fostering transparency and trust in the blood donation process.

- Scalability: Designed to be scalable, the system can accommodate increasing numbers of donors, recipients, and blood inventory items without compromising performance or functionality.

Overall, the Blood Bank Management System serves as a critical tool in bridging the gap between blood donors and recipients, ensuring the efficient and effective management of blood donations to meet the needs of patients.

7.2 Scope for future work

While the current system represents a significant advancement, there are several areas for future improvement and expansion:

- Mobile Integration: Developing mobile applications for the system would enhance accessibility, allowing donors and recipients to register, request blood, and access information conveniently from their smartphones. This would broaden the reach of the system and encourage greater participation in blood donation drives.

- AI Integration: Integrating artificial intelligence (AI) algorithms into the system could enable predictive analysis of blood demand based on historical data, seasonal trends, and demographic factors. This would optimize blood inventory management, ensuring adequate stock levels while minimizing wastage.

- Advanced Testing Techniques: Enhancing the blood testing module with advanced screening techniques, such as nucleic acid testing (NAT) for detecting infectious diseases, would further improve the safety and quality of donated blood, reducing the risk of transfusion-transmitted infections.

- Collaboration with External Databases: Integrating the system with external databases, such as national donor registries or electronic health records (EHRs), would facilitate broader donor-recruitment campaigns and enable seamless sharing of donor information between blood banks, healthcare facilities, and regulatory authorities.

- Continuous Improvement: Regular updates and enhancements to the system based on user feedback and technological advancements in the healthcare sector are essential for ensuring its relevance and effectiveness over time. This includes incorporating new features, improving user experience, and staying compliant with regulatory standards and best practices.

Each Team Member Contribution:

Team Member	Contribution
MANDAR CHAUDHARI	Chapter 4
NAMAN MAHESHWARI	Chapter 5, 6 & 7
SHIVLI MATHUR	Chapter 4
BANDI RISHIK REDDY	Chapter 5, 6 & 7
DEVESH AHUJA	Abstract, Chapter 1,2 and 3, Create and Insert Queries & 5.1
VIGNESH SURESH MENON	Abstract, Chapter 1,2 and 3, Create and Insert Queries & 5.1