Department of Computer Science

California State University, Channel Islands

COMP 520
Advanced Database Systems

Blood Bank Management System

Term Project Report (Part-III Implementation and Write-up)

By: Kumar Saurav Jha and Uday Kiran Jadi

Introduction

The Blood Bank Management System is a comprehensive database solution designed to modernize and streamline operations for blood banks and healthcare institutions. This system addresses the critical need for efficient blood inventory management while ensuring the security and privacy of donor/recipient information in compliance with medical data protection standards.

Key System Components

The system comprises several integrated modules that work together to provide complete blood bank management:

1. **Donor/Recipient Management**: Centralized database for all personal and medical information
2. **Inventory Tracking**: Real-time monitoring of blood stock levels by blood type
3. **Transaction Processing**: Records all blood donations and transfusions with timestamps
4. **Reporting Tools**: Generates historical reports and statistical analyses

Technical Implementation

Built on a relational database foundation using MySQL, the system features:

- Normalized table structures to minimize data redundancy
- Automated triggers for maintaining data integrity
- Stored procedures for complex operations
- PHP-based web interface for cross-platform accessibility
- Secure data encryption protocols

Benefits and Advantages

This system represents a significant improvement over traditional paper-based or spreadsheet methods by:

- Reducing human errors in blood type matching and inventory tracking
- Providing instant access to critical information during emergencies
- Generating automated alerts for low stock levels
- Maintaining comprehensive audit trails for all transactions
- Ensuring compliance with healthcare data regulations

The Blood Bank Management System serves as a vital tool for healthcare institutions to efficiently manage their blood supply chain, ultimately contributing to better patient care and potentially saving lives through more effective blood resource management.

Implementation Documentation

- Data definition language (DDL)

The database is named blood_bank and uses UTF-8 character encoding (utf8mb4). It consists of 5 tables that manage blood donors, donations, recipients, blood stock, and system users. We haven’t considered the system users too much this is just for login into the application.

Tables and Their Structures

1. Person Table (Core Entity)

- **Purpose**: Stores information about individuals (both donors and recipients)
- **Structure**:
  - person_id (INT, Primary Key, Auto-increment): Unique identifier
  - person_name (VARCHAR(25)): Full name
  - person_phone (CHAR(10)): Phone number (exactly 10 digits)
  - person_dob (DATE): Date of birth
  - person_address (VARCHAR(100), nullable): Physical address
  - person_gender (CHAR(1)): Single character for gender (M/F/O)
  - person_blood_group (ENUM): Blood type (A+, A-, B+, B-, AB+, AB-, O+, O-)
  - person_med_issues (VARCHAR(100), nullable): Medical conditions or issues

NAME	TYPE	NULLABLE	DESCRIPTION
person_id	int(10)	NOT NULL	Unique person identifier (auto-incremented)
person_name	varchar(25)	NOT NULL	Full name of the person
person_phone	char(10)	NOT NULL	10-digit phone number
person_dob	date	NOT NULL	Date of birth
person_address	varchar(100)	NULL	Physical address (can be null)
person_gender	char(1)	NOT NULL	Gender (M/F/O)
person_blood_group	ENUM('A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-')	NOT NULL	Blood group type
person_med_issues	varchar(100)	NULL	Any medical conditions or issues (can be null)

2. Donation Table

- **Purpose**: Records blood donation events
- **Structure**:
  - person_id (INT, Foreign Key): References person.person_id
  - donation_date (DATE): Date of donation
  - donation_time (TIME): Time of donation

    - donation_quantity (INT(1)): Number of units donated (typically 1)
- **Composite Primary Key**: (person_id, donation_date, donation_time)
- **Relationship**: Many-to-one with person table (a person can have multiple donations)

NAME	TYPE	NULLABLE	DESCRIPTION
person_id	int(10)	NOT NULL	Foreign key referencing person.person_id
donation_date	date	NOT NULL	Date when donation was made
donation_time	time	NOT NULL	Time when donation was made
donation_quantity	int(1)	NOT NULL	Number of units donated (typically 1)

3. Receive Table

- **Purpose**: Records blood transfusion events
- **Structure**:
    - person_id (INT, Foreign Key): References person.person_id
    - received_date (DATE): Date of transfusion
    - received_time (TIME): Time of transfusion
    - received_quantity (INT(1)): Number of units received
    - received_hospital (VARCHAR(50)): Hospital name where transfusion occurred
- **Composite Primary Key**: (person_id, received_date, received_time)
- **Relationship**: Many-to-one with person table (a person can receive blood multiple times)

NAME	TYPE	NULLABLE	DESCRIPTION
person_id	int(10)	NOT NULL	Foreign key referencing person.person_id
received_date	date	NOT NULL	Date when blood was received
received_time	time	NOT NULL	Time when blood was received
received_quantity	int(1)	NOT NULL	Number of units received
received_hospital	varchar(50)	NOT NULL	Name of hospital where blood was received

4. Stock Table

- **Purpose**: Tracks current blood inventory levels
- **Structure**:
  - stock_blood_group (ENUM): Blood type (same values as person_blood_group)
  - stock_quantity (INT(5), default 0): Current units in stock
- **Primary Key**: stock_blood_group
- **Initial Data**: Pre-populated with all blood types (A+, A-, etc.) with quantity 0.

NAME	TYPE	NULLABLE	DESCRIPTION
stock_blood_group	ENUM('A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-')	NOT NULL	Blood group type
stock_quantity	int(5)	NOT NULL	Current units in stock (default 0)

5. User Table

- **Purpose**: Stores system user credentials
- **Structure**:
  - username (VARCHAR(10), Primary Key): User login name
  - password (VARCHAR(16)): User password (plaintext in example)
- **Sample Data**: Includes 'SuperAdmin' and 'test_user' accounts.

NAME	TYPE	NULLABLE	DESCRIPTION
username	varchar(10)	NOT NULL	User login name
password	varchar(16)	NOT NULL	User password (stored in plaintext)

Relationships

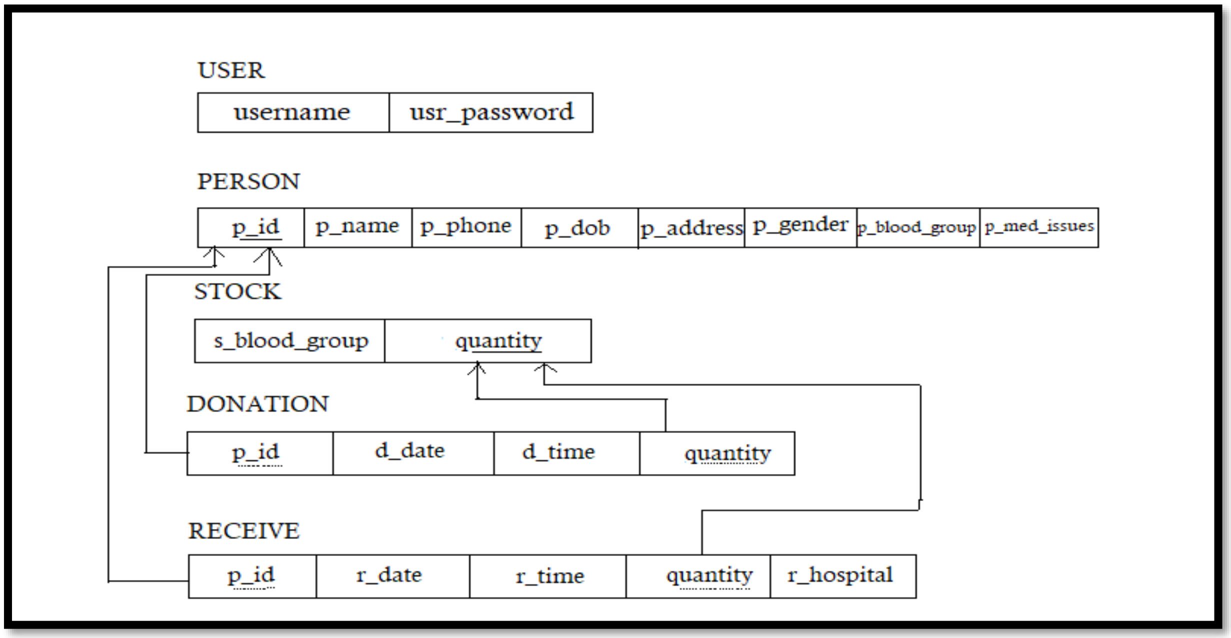
1. **Person-Donation**: One-to-many
   - A person can make multiple donations
   - Each donation is linked to exactly one person
2. **Person-Receive**: One-to-many
   - A person can receive blood multiple times
   - Each transfusion record is linked to exactly one person

Constraints

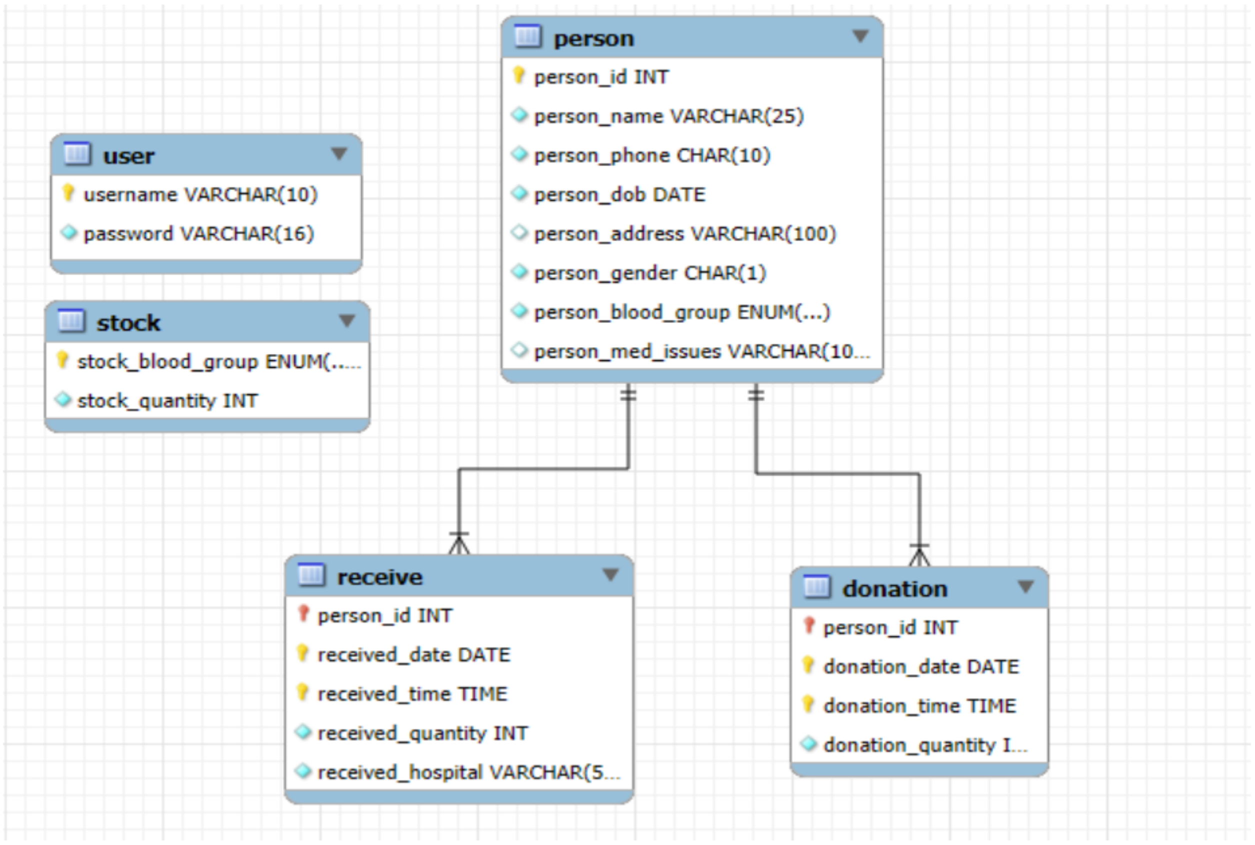
1. **Primary Keys**:
   - person: person_id
   - donation: (person_id, donation_date, donation_time)
   - receive: (person_id, received_date, received_time)
   - stock: stock_blood_group
   - user: username
2. **Foreign Keys**:
   - donation.person_id → person.person_id
   - receive.person_id → person.person_id
3. **Data Constraints**:
   - Blood groups are restricted to specific ENUM values
   - Phone numbers must be exactly 10 characters
   - Donation and receive quantities are integers (though INT(1) is likely meant to represent single units)
   - Person ID auto-increments for new records

Schema Diagram:

We have used short names for fields so it will be easier to fit in the image.



ER-Diagram



Our **Blood Bank Management System** is built on a relational database with five core tables designed to streamline blood donation tracking, inventory management, and user access. The **USER** table (username, password) authenticates system users, with username limited to 10 characters and password to 16, ensuring secure yet concise credentials. The **PERSON** table acts as the central hub, storing comprehensive donor/receiver details: person_id (unique identifier), person_name (25-characterlimit), person_phone (exactly 10 digits for uniformity), person_dob (date of birth), person_address (100-character flexible field), person_gender (single character: M/F), person_blood_group (ENUM: predefined blood types like 'A+', 'O-'), and person_med_issues (100-character notes on health conditions). The **STOCK** table tracks blood inventory with stock_blood_group (matching the ENUM from PERSON) and stock_quantity (integer count of available units), ensuring real-time visibility into supply levels.

Two transactional tables manage blood movement:

1. **DONATION** records each donation with person_id (linking to PERSON), donation_date, donation_time, and donation_quantity (typically 1-2 units, validated via triggers).
2. **RECEIVE** logs transfusions with person_id, received_date, received_time, received_quantity, and received_hospital (50-character field for hospital name), enabling traceability from donor to recipient.

Key Features & Integrity Rules:

- **Referential Integrity**: All person_id fields in DONATION/RECEIVE reference PERSON’s primary key, preventing orphaned records.
- **Data Validation**: ENUMs (e.g., blood groups), fixed-length fields (e.g., person_phone), and triggers (e.g., stock auto-update) enforce consistency.
- **Workflow Support**: Donations increment STOCK quantities via triggers, while RECEIVE entries decrement stock, ensuring inventory accuracy.

This schema balances flexibility (VARCHAR for addresses/hospital names) with strict controls (ENUMs, CHAR limits), making it robust for daily operations (donor registration, blood disbursement) and regulatory compliance (audit trails, eligibility checks via person_med_issues).

- <u>Data Manipulation Language(DML) Views</u>

1. **Active Donors View**

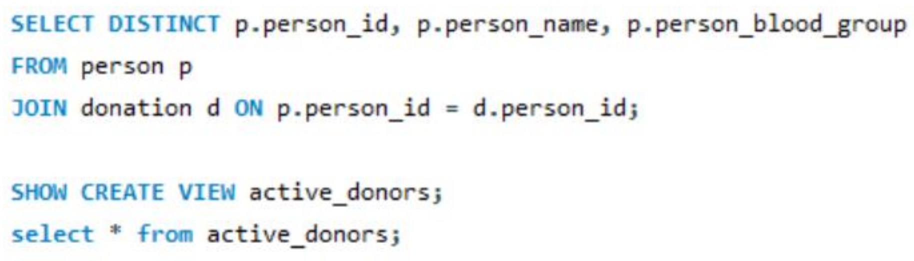
   Lists all people who have donated blood.

   CREATE VIEW active_donors AS

   SELECT DISTINCT p.person_id, p.person_name, p.person_blood_group

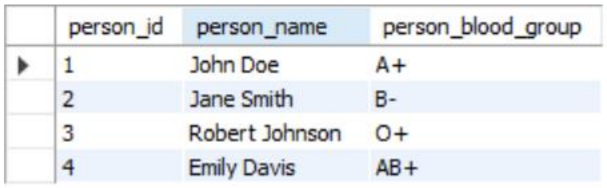
   FROM person p

   JOIN donation d ON p.person_id = d.person_id;



```
SELECT DISTINCT p.person_id, p.person_name, p.person_blood_group
FROM person p
JOIN donation d ON p.person_id = d.person_id;

SHOW CREATE VIEW active_donors;
select * from active_donors;
```

   Output:



person_id	person_name	person_blood_group
1	John Doe	A+
2	Jane Smith	B-
3	Robert Johnson	O+
4	Emily Davis	AB+

2. **Blood Group Stock View**

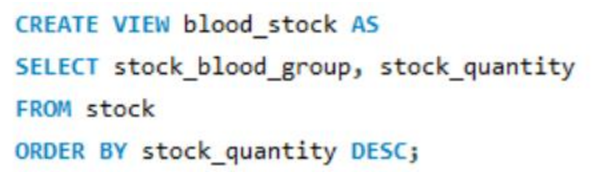
   Shows current blood stock quantities

   CREATE VIEW blood_stock AS

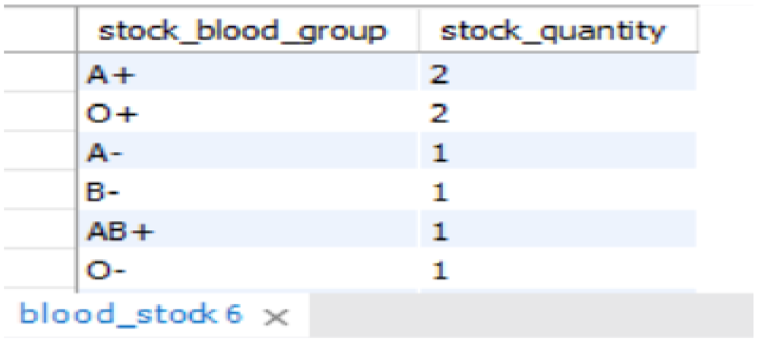
   SELECT stock_blood_group, stock_quantity

FROM stock

ORDER BY stock_quantity DESC;



```
CREATE VIEW blood_stock AS
SELECT stock_blood_group, stock_quantity
FROM stock
ORDER BY stock_quantity DESC;
```



	stock_blood_group	stock_quantity
	A+	2
	O+	2
	A-	1
	B-	1
	AB+	1
	O-	1

blood_stock 6

3. **Recent Donations View**
   Shows donations from the last 30 days

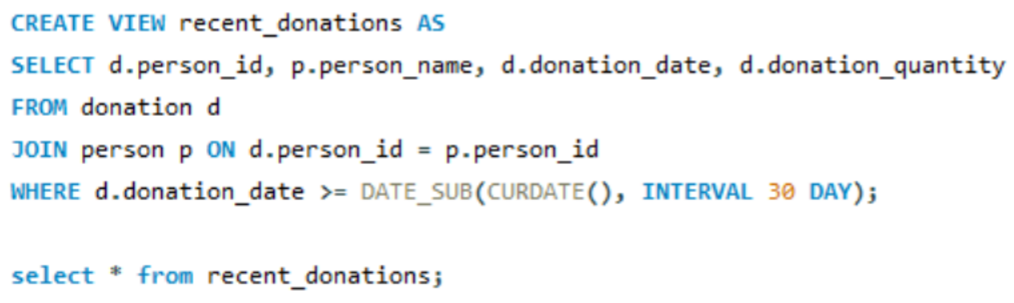
   CREATE VIEW recent_donations AS

   SELECT d.person_id, p.person_name, d.donation_date, d.donation_quantity

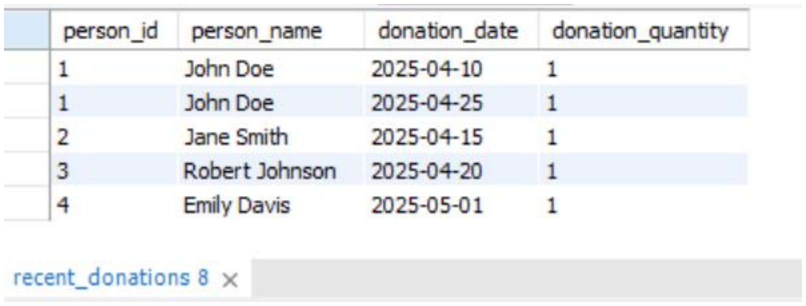
   FROM donation d

   JOIN person p ON d.person_id = p.person_id

   WHERE d.donation_date >= DATE_SUB(CURDATE(), INTERVAL 30 DAY);



```
CREATE VIEW recent_donations AS
SELECT d.person_id, p.person_name, d.donation_date, d.donation_quantity
FROM donation d
JOIN person p ON d.person_id = p.person_id
WHERE d.donation_date >= DATE_SUB(CURDATE(), INTERVAL 30 DAY);

select * from recent_donations;
```



person_id	person_name	donation_date	donation_quantity
1	John Doe	2025-04-10	1
1	John Doe	2025-04-25	1
2	Jane Smith	2025-04-15	1
3	Robert Johnson	2025-04-20	1
4	Emily Davis	2025-05-01	1

4. **Blood Receivers View**

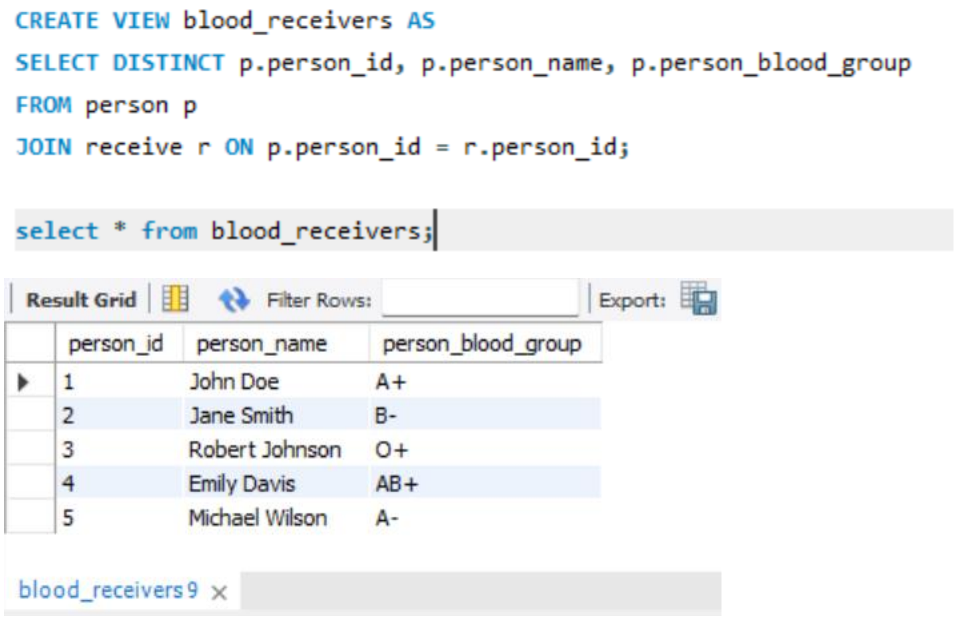
   Lists all people who received blood

   CREATE VIEW blood_receivers AS

   SELECT DISTINCT p.person_id, p.person_name, p.person_blood_group

   FROM person p

   JOIN receive r ON p.person_id = r.person_id;



```
CREATE VIEW blood_receivers AS
SELECT DISTINCT p.person_id, p.person_name, p.person_blood_group
FROM person p
JOIN receive r ON p.person_id = r.person_id;

select * from blood_receivers;
```

person_id	person_name	person_blood_group
1	John Doe	A+
2	Jane Smith	B-
3	Robert Johnson	O+
4	Emily Davis	AB+
5	Michael Wilson	A-

5.Donor Count by Blood Group

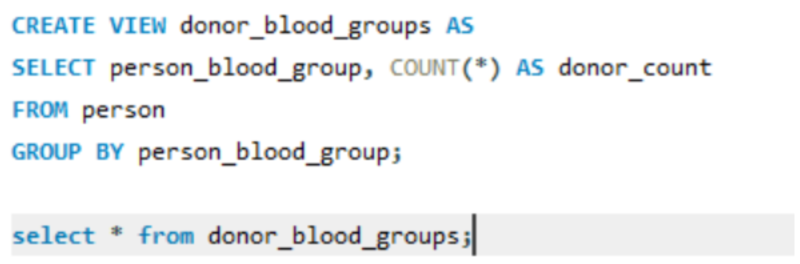
Counts donors for each blood type

CREATE VIEW donor_blood_groups AS

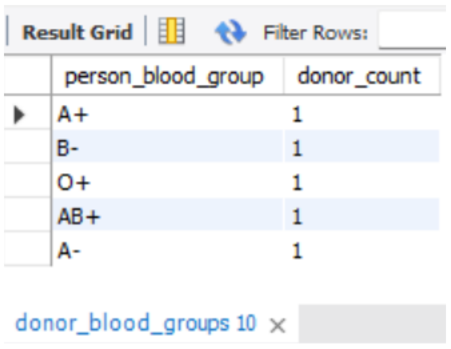
SELECT person_blood_group, COUNT(*) AS donor_count

FROM person

GROUP BY person_blood_group;



```
CREATE VIEW donor_blood_groups AS
SELECT person_blood_group, COUNT(*) AS donor_count
FROM person
GROUP BY person_blood_group;

select * from donor_blood_groups;
```



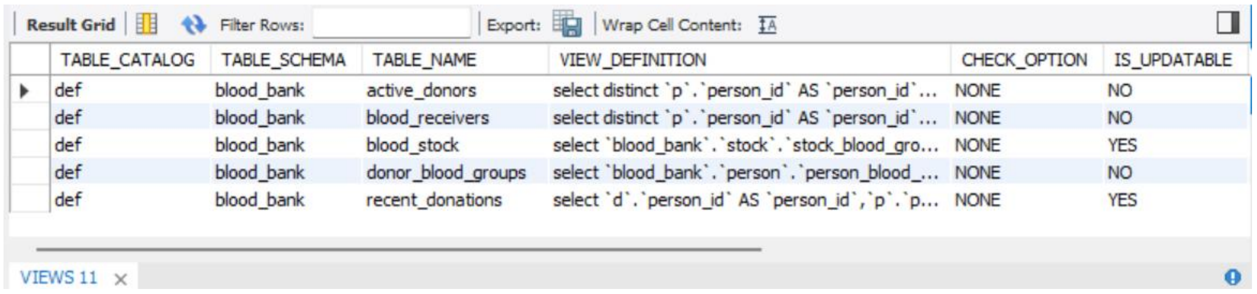
person_blood_group	donor_count
A+	1
B-	1
O+	1
AB+	1
A-	1

Here is the image for all the views that exist in our database.

SELECT *

FROM information_schema.VIEWS

WHERE TABLE_SCHEMA = 'blood_bank';



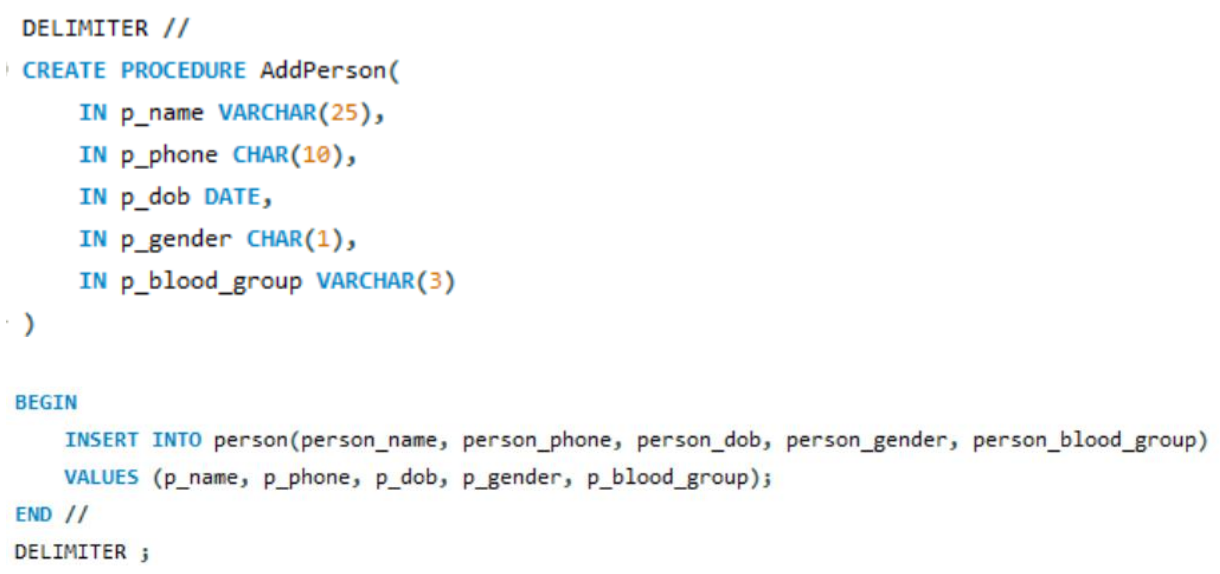
TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	VIEW_DEFINITION	CHECK_OPTION	IS_UPDATABLE
def	blood_bank	active_donors	select distinct `p`.`person_id` AS `person_id`...	NONE	NO
def	blood_bank	blood_receivers	select distinct `p`.`person_id` AS `person_id`...	NONE	NO
def	blood_bank	blood_stock	select `blood_bank`.`stock`.`stock_blood_gro...	NONE	YES
def	blood_bank	donor_blood_groups	select `blood_bank`.`person`.`person_blood_...	NONE	NO
def	blood_bank	recent_donations	select `d`.`person_id` AS `person_id`,`p`.`p...	NONE	YES

- Data Manipulation Language(DML) Queries(Stored Procedures)

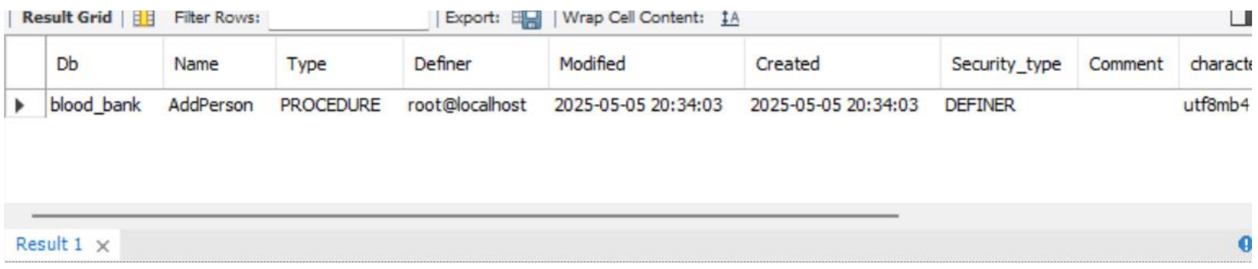
1. Add New Person

Creates new donor/recipient records with essential details.
Business Use: Patient registration and donor onboarding.
Admin Use: Maintains clean demographic data for all system users.

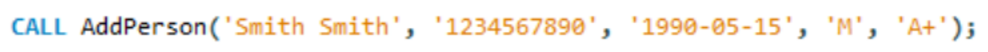
```
 DELIMITER //
CREATE PROCEDURE AddPerson(
    IN p_name VARCHAR(25),
    IN p_phone CHAR(10),
    IN p_dob DATE,
    IN p_gender CHAR(1),
    IN p_blood_group VARCHAR(3)
)
BEGIN
    INSERT INTO person(person_name, person_phone, person_dob,
person_gender, person_blood_group)
    VALUES (p_name, p_phone, p_dob, p_gender, p_blood_group);
END //
DELIMITER ;
```



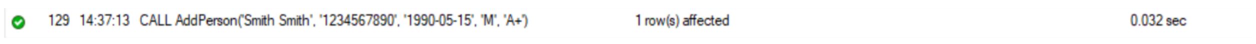
```
DELIMITER //
CREATE PROCEDURE AddPerson(
    IN p_name VARCHAR(25),
    IN p_phone CHAR(10),
    IN p_dob DATE,
    IN p_gender CHAR(1),
    IN p_blood_group VARCHAR(3)
)

BEGIN
    INSERT INTO person(person_name, person_phone, person_dob, person_gender, person_blood_group)
    VALUES (p_name, p_phone, p_dob, p_gender, p_blood_group);
END //
DELIMITER ;
```



Db	Name	Type	Definer	Modified	Created	Security_type	Comment	characte
blood_bank	AddPerson	PROCEDURE	root@localhost	2025-05-05 20:34:03	2025-05-05 20:34:03	DEFINER		utf8mb4



```
CALL AddPerson('Smith Smith', '1234567890', '1990-05-15', 'M', 'A+');
```



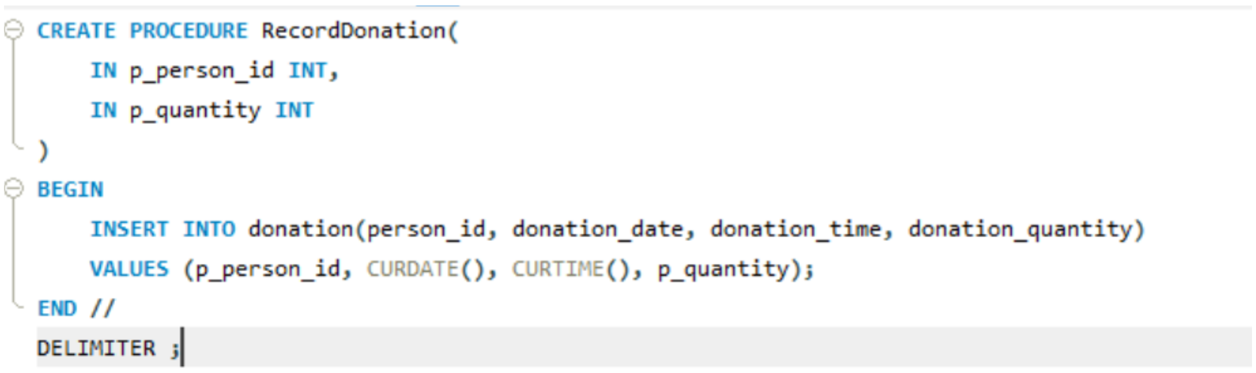
```
129  14:37:13  CALL AddPerson('Smith Smith', '1234567890', '1990-05-15', 'M', 'A+')    1 row(s) affected    0.032 sec
```

2. Record Blood Donation

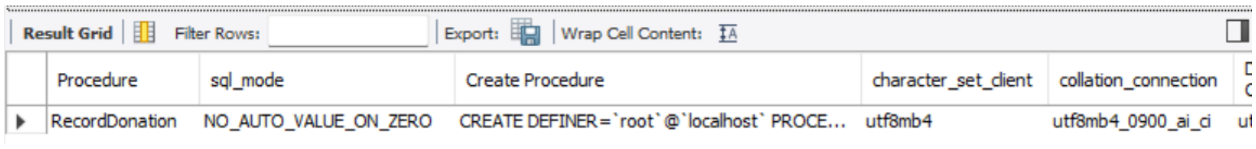
Tracks blood donations with timestamps and quantities.
Business Use: Documents collection events for inventory tracking.
Admin Use: Provides audit for blood product lifecycle.

```
DELIMITER //
CREATE PROCEDURE RecordDonation(
    IN p_person_id INT,
    IN p_quantity INT
)
BEGIN
```

```
    INSERT INTO donation(person_id, donation_date, donation_time,
donation_quantity)
    VALUES (p_person_id, CURDATE(), CURTIME(), p_quantity);
END //
DELIMITER ;
```



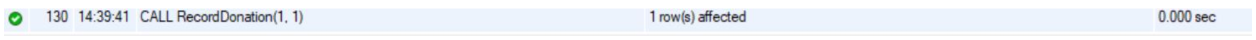
```
CREATE PROCEDURE RecordDonation(
    IN p_person_id INT,
    IN p_quantity INT
)
BEGIN
    INSERT INTO donation(person_id, donation_date, donation_time, donation_quantity)
    VALUES (p_person_id, CURDATE(), CURTIME(), p_quantity);
END //
DELIMITER ;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Procedure	sql_mode	Create Procedure	character_set_client	collation_connection	D
RecordDonation	NO_AUTO_VALUE_ON_ZERO	CREATE DEFINER=`root`@`localhost` PROCE...	utf8mb4	utf8mb4_0900_ai_ci	ut

```
CALL RecordDonation(1, 1);
```



```
130  14:39:41  CALL RecordDonation(1, 1)        1 row(s) affected        0.000 sec
```

3. Record Blood Receival

Records blood transfusions to patients at hospitals.
Business Use: Manages distribution logistics and usage tracking.
Admin Use: Supports product recall capabilities if needed.

```
DELIMITER //
CREATE PROCEDURE RecordReceival(
    IN p_person_id INT,
    IN p_quantity INT,
    IN p_hospital VARCHAR(50)
)
```

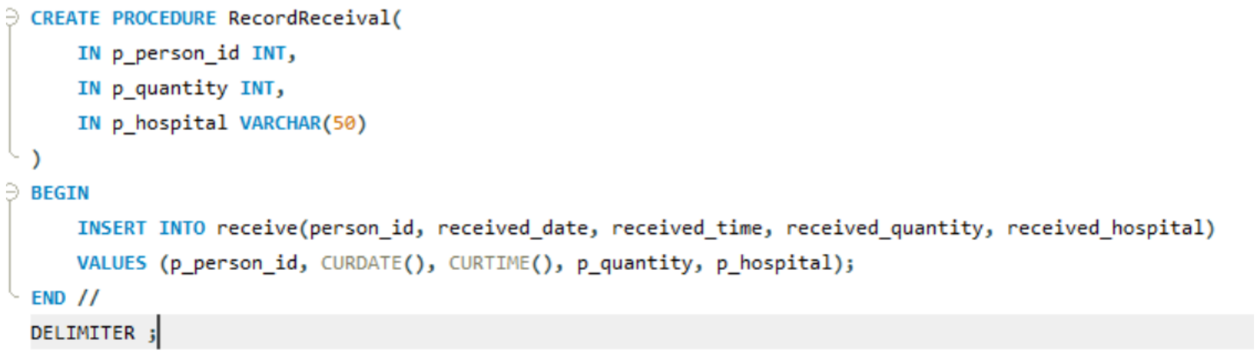
BEGIN

    INSERT INTO receive(person_id, received_date, received_time, received_quantity, received_hospital)

    VALUES (p_person_id, CURDATE(), CURTIME(), p_quantity, p_hospital);

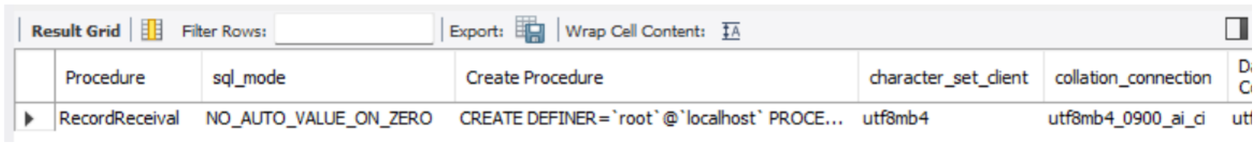
END //

DELIMITER ;



```
CREATE PROCEDURE RecordReceival(
    IN p_person_id INT,
    IN p_quantity INT,
    IN p_hospital VARCHAR(50)
)
BEGIN
    INSERT INTO receive(person_id, received_date, received_time, received_quantity, received_hospital)
    VALUES (p_person_id, CURDATE(), CURTIME(), p_quantity, p_hospital);
END //
DELIMITER ;
```

CALL RecordReceival(1, 2, “”);



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Procedure	sql_mode	Create Procedure	character_set_client	collation_connection	D C
RecordReceival	NO_AUTO_VALUE_ON_ZERO	CREATE DEFINER=`root`@`localhost` PROCE...	utf8mb4	utf8mb4_0900_ai_ci	ut

4. Update Blood Stock
Adjusts inventory counts for specific blood types.
Business Use: Real-time stock level management.
Admin Use: Corrects discrepancies during physical inventory checks.

DELIMITER //

CREATE PROCEDURE UpdateStock(

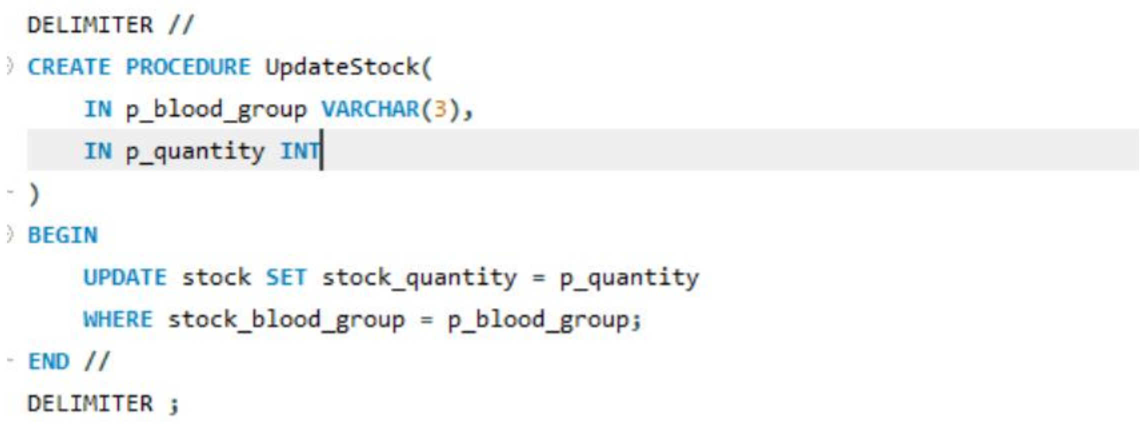
    IN p_blood_group VARCHAR(3),

    IN p_quantity INT

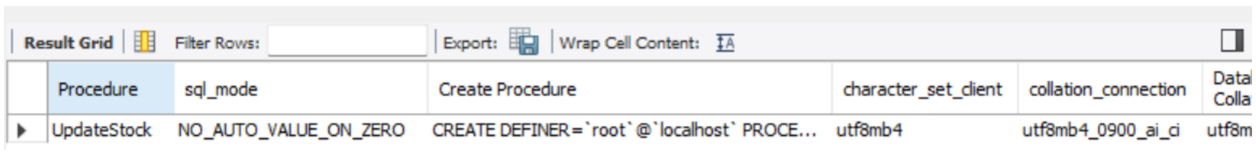
)

BEGIN

```
    UPDATE stock SET stock_quantity = p_quantity
    WHERE stock_blood_group = p_blood_group;
END //
DELIMITER ;
```



```
DELIMITER //
CREATE PROCEDURE UpdateStock(
    IN p_blood_group VARCHAR(3),
    IN p_quantity INT
)
BEGIN
    UPDATE stock SET stock_quantity = p_quantity
    WHERE stock_blood_group = p_blood_group;
END //
DELIMITER ;
```

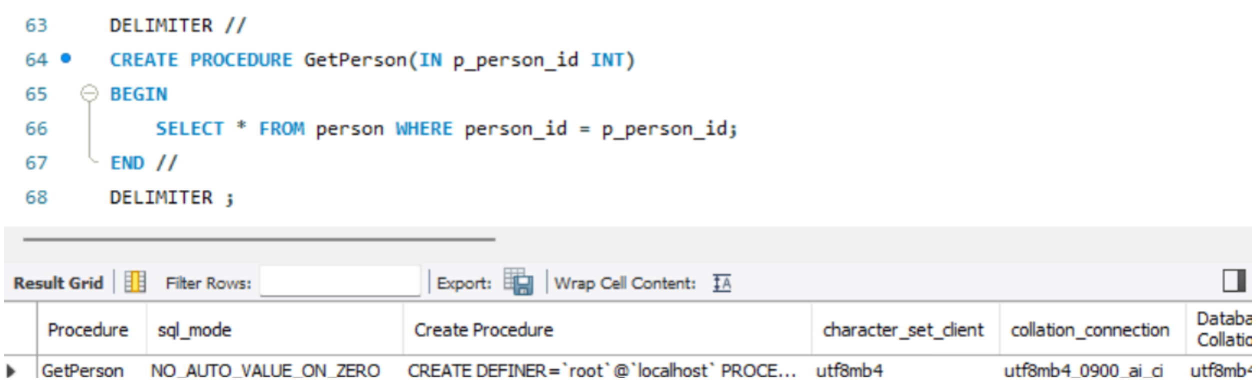


Procedure	sql_mode	Create Procedure	character_set_client	collation_connection	Data Colla
UpdateStock	NO_AUTO_VALUE_ON_ZERO	CREATE DEFINER=`root`@`localhost` PROCE...	utf8mb4	utf8mb4_0900_ai_ci	utf8m

```
CALL UpdateStock('AB+', 8)
```

5. Get Person Details
Retrieves complete profile for any registered individual.
Business Use: Donor/recipient verification at facilities.
Admin Use: Supports customer service inquiries.

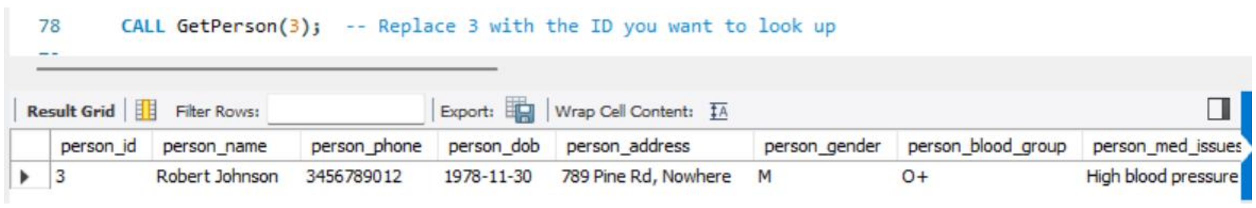
```
DELIMITER //
CREATE PROCEDURE GetPerson(IN p_person_id INT)
BEGIN
    SELECT * FROM person WHERE person_id = p_person_id;
END //
DELIMITER ;
```



```
DELIMITER //
CREATE PROCEDURE GetPerson(IN p_person_id INT)
BEGIN
    SELECT * FROM person WHERE person_id = p_person_id;
END //
DELIMITER ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Procedure	sql_mode	Create Procedure	character_set_client	collation_connection	Databa Collatic
GetPerson	NO_AUTO_VALUE_ON_ZERO	CREATE DEFINER=`root`@`localhost` PROCE...	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4



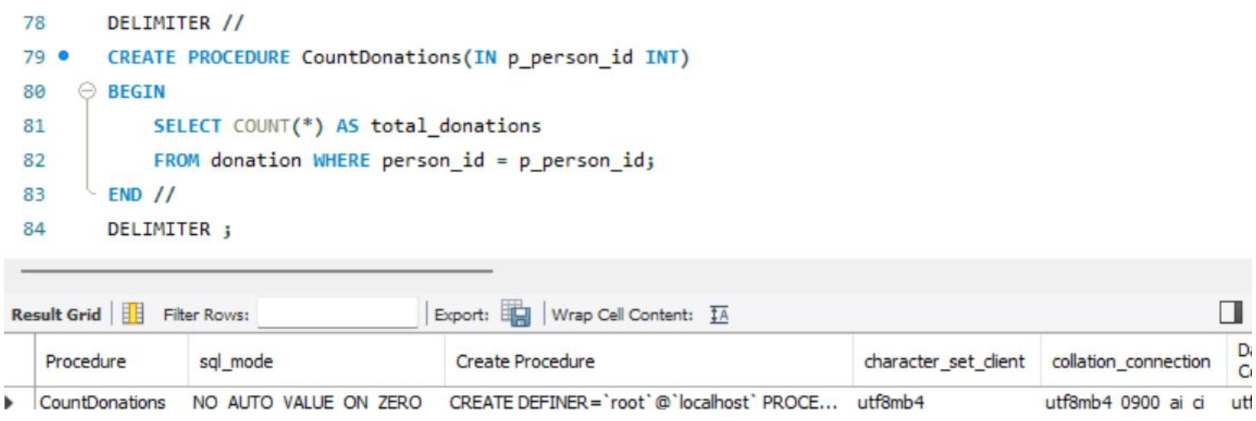
```
CALL GetPerson(3);  -- Replace 3 with the ID you want to look up
--
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

person_id	person_name	person_phone	person_dob	person_address	person_gender	person_blood_group	person_med_issues
3	Robert Johnson	3456789012	1978-11-30	789 Pine Rd, Nowhere	M	O+	High blood pressure

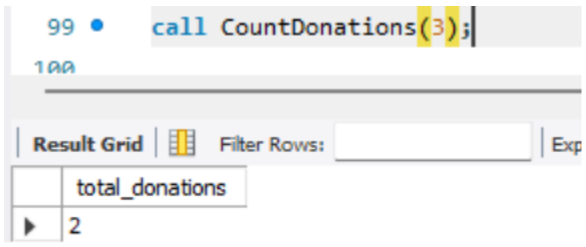
6. Count Donations by Person
Calculates lifetime donations per donor.
Business Use: Identifies frequent donors for recognition programs.
Admin Use: Measures donor engagement metrics.

```
DELIMITER //

CREATE PROCEDURE CountDonations(IN p_person_id INT)

BEGIN

    SELECT COUNT(*) AS total_donations

    FROM donation WHERE person_id = p_person_id;

END //

DELIMITER ;
```



```sql
DELIMITER //
CREATE PROCEDURE CountDonations(IN p_person_id INT)
BEGIN
    SELECT COUNT(*) AS total_donations
    FROM donation WHERE person_id = p_person_id;
END //
DELIMITER ;
```

	Procedure	sql_mode	Create Procedure	character_set_client	collation_connection	D C
▶	CountDonations	NO AUTO VALUE ON ZERO	CREATE DEFINER=`root`@`localhost` PROCE...	utf8mb4	utf8mb4 0900 ai ci	utf



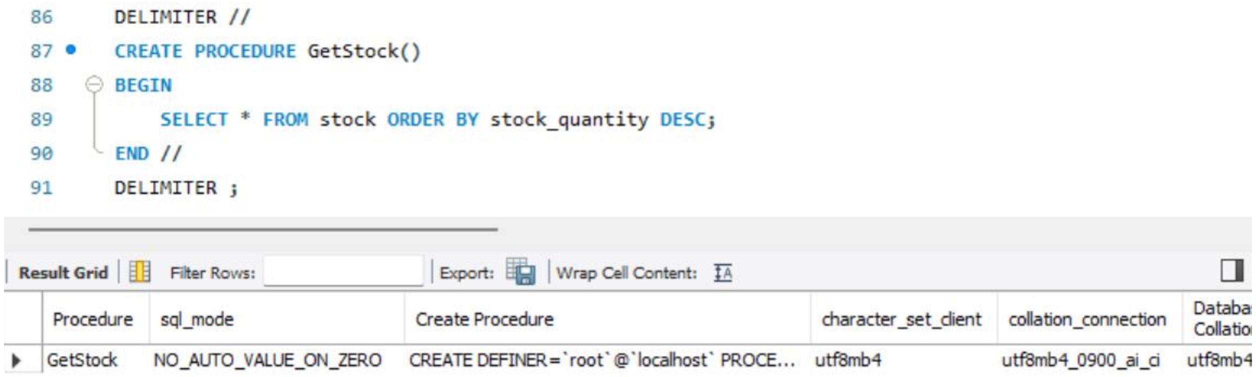
```sql
call CountDonations(3);
```

	total_donations
▶	2

7. Get Current Stock Levels
Displays current inventory across all blood types.
Business Use: Daily operational planning.
Admin Use: Compliance reporting for regulatory requirements.

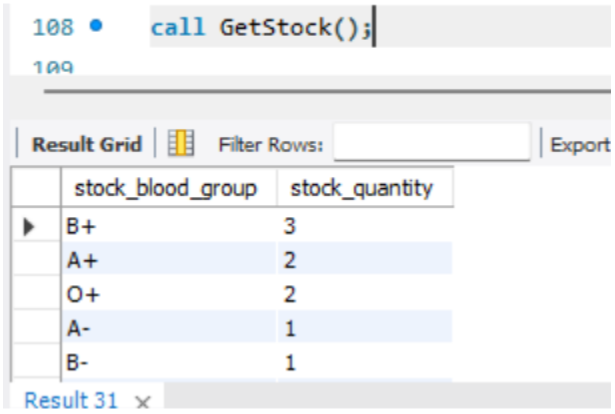
```sql
DELIMITER //

CREATE PROCEDURE GetStock()

BEGIN

    SELECT * FROM stock ORDER BY stock_quantity DESC;

END //
```

DELIMITER ;



```
DELIMITER //
CREATE PROCEDURE GetStock()
BEGIN
    SELECT * FROM stock ORDER BY stock_quantity DESC;
END //
DELIMITER ;
```

	Procedure	sql_mode	Create Procedure	character_set_client	collation_connection	Databa Collatio
▸	GetStock	NO_AUTO_VALUE_ON_ZERO	CREATE DEFINER=`root`@`localhost` PROCE...	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4



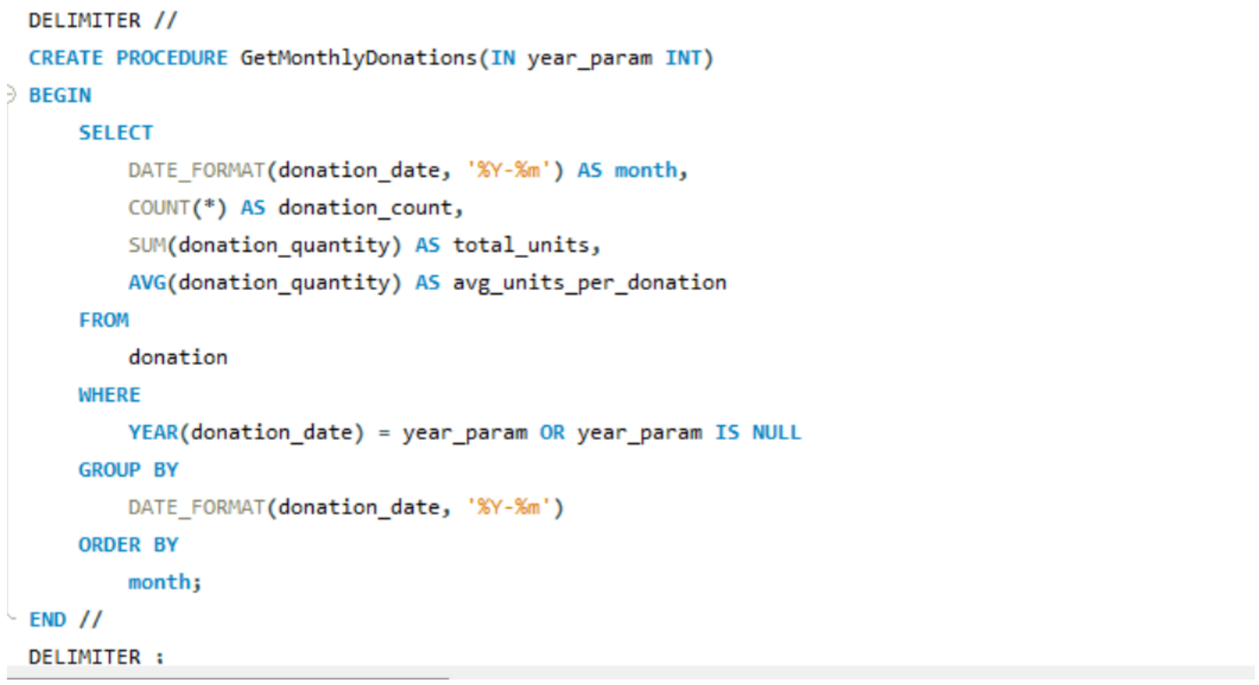
```
call GetStock();
```

	stock_blood_group	stock_quantity
▸	B+	3
	A+	2
	O+	2
	A-	1
	B-	1

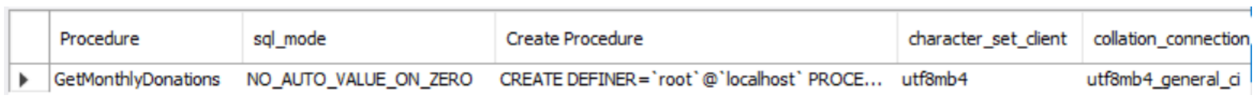
Result 31

8. GetMonthlyDonations
Shows donation statistics aggregated by month.
Business Use: Reveals seasonal patterns to optimize blood drive scheduling and donor retention campaigns.
Admin Use: Months collection performance metrics and staff allocation efficiency.

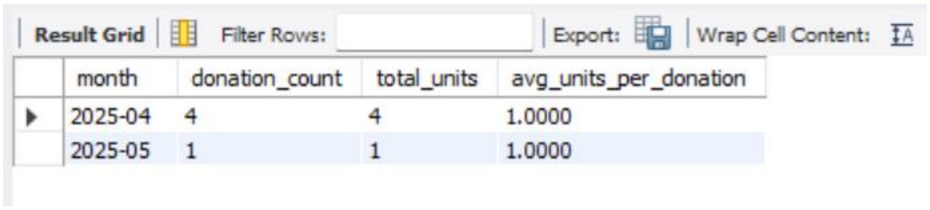
```
DELIMITER //

CREATE PROCEDURE GetMonthlyDonations(IN year_param INT)

BEGIN

    SELECT

        DATE_FORMAT(donation_date, '%Y-%m') AS month,

        COUNT(*) AS donation_count,

        SUM(donation_quantity) AS total_units,
        AVG(donation_quantity) AS avg_units_per_donation
    FROM
        donation
    WHERE
        YEAR(donation_date) = year_param OR year_param IS NULL
    GROUP BY
        DATE_FORMAT(donation_date, '%Y-%m')
    ORDER BY
        month;
END //
DELIMITER ;
```



```
DELIMITER //
CREATE PROCEDURE GetMonthlyDonations(IN year_param INT)
BEGIN
    SELECT
        DATE_FORMAT(donation_date, '%Y-%m') AS month,
        COUNT(*) AS donation_count,
        SUM(donation_quantity) AS total_units,
        AVG(donation_quantity) AS avg_units_per_donation
    FROM
        donation
    WHERE
        YEAR(donation_date) = year_param OR year_param IS NULL
    GROUP BY
        DATE_FORMAT(donation_date, '%Y-%m')
    ORDER BY
        month;
END //
DELIMITER ;
```



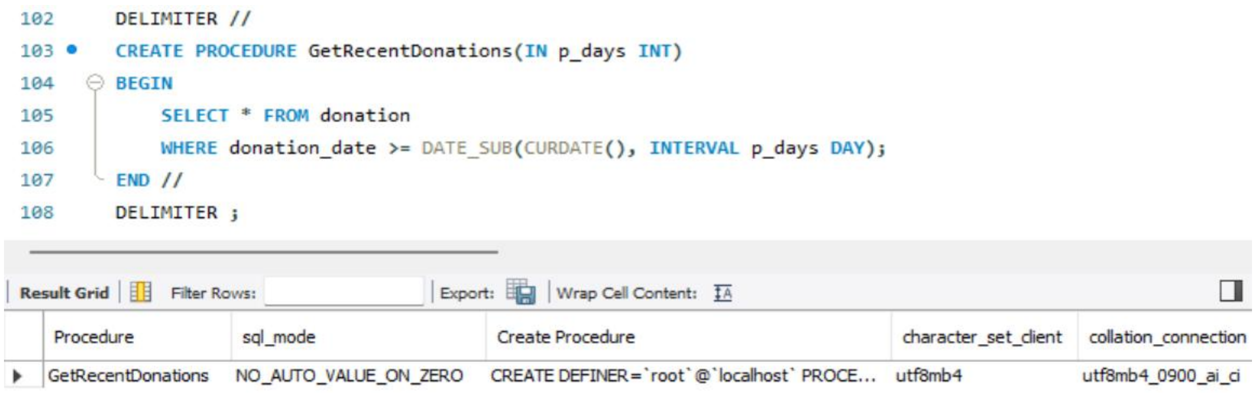
	Procedure	sql_mode	Create Procedure	character_set_client	collation_connection
▶	GetMonthlyDonations	NO_AUTO_VALUE_ON_ZERO	CREATE DEFINER=`root`@`localhost` PROCE...	utf8mb4	utf8mb4_general_ci


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

month	donation_count	total_units	avg_units_per_donation
2025-04	4	4	1.0000
2025-05	1	1	1.0000

9. Get Recent Donations
Shows donations within a customizable time window.
Business Use: Identifies active vs. lapsed donors.
Admin Use: Tracks collection team productivity.

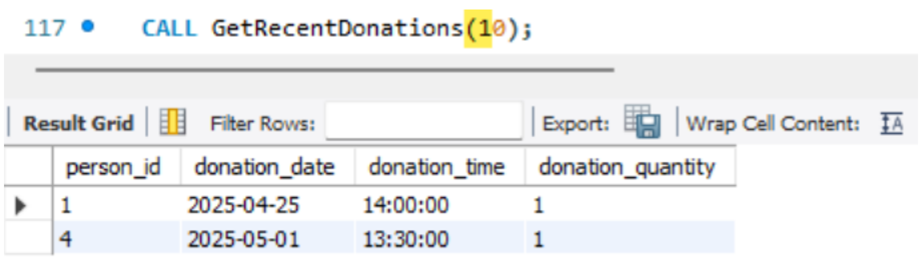
```
DELIMITER //
CREATE PROCEDURE GetRecentDonations(IN p_days INT)
BEGIN
    SELECT * FROM donation
    WHERE donation_date >= DATE_SUB(CURDATE(), INTERVAL p_days
DAY);
END //
DELIMITER ;
```



```
DELIMITER //
CREATE PROCEDURE GetRecentDonations(IN p_days INT)
BEGIN
    SELECT * FROM donation
    WHERE donation_date >= DATE_SUB(CURDATE(), INTERVAL p_days DAY);
END //
DELIMITER ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Procedure	sql_mode	Create Procedure	character_set_client	collation_connection
GetRecentDonations	NO_AUTO_VALUE_ON_ZERO	CREATE DEFINER=`root`@`localhost` PROCE...	utf8mb4	utf8mb4_0900_ai_ci



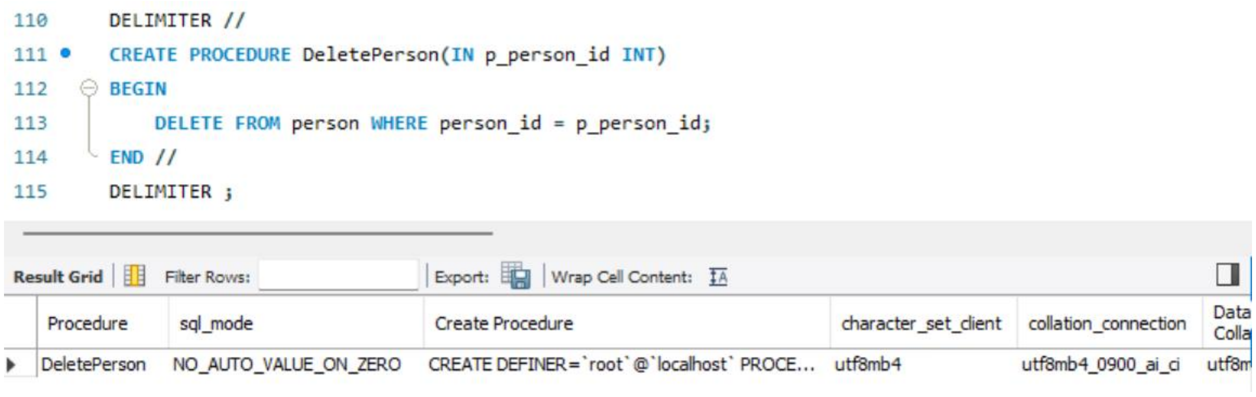
```
CALL GetRecentDonations(10);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

person_id	donation_date	donation_time	donation_quantity
1	2025-04-25	14:00:00	1
4	2025-05-01	13:30:00	1

10. Delete Person Record
Removes individual records with all associated data.
Business Use: GDPR compliance for right-to-be-forgotten requests.
Admin Use: Database maintenance and cleanup.

```
DELIMITER //

CREATE PROCEDURE DeletePerson(IN p_person_id INT)

BEGIN

    DELETE FROM person WHERE person_id = p_person_id;

END //

DELIMITER ;
```



```
DELIMITER //
CREATE PROCEDURE DeletePerson(IN p_person_id INT)
BEGIN
    DELETE FROM person WHERE person_id = p_person_id;
END //
DELIMITER ;
```

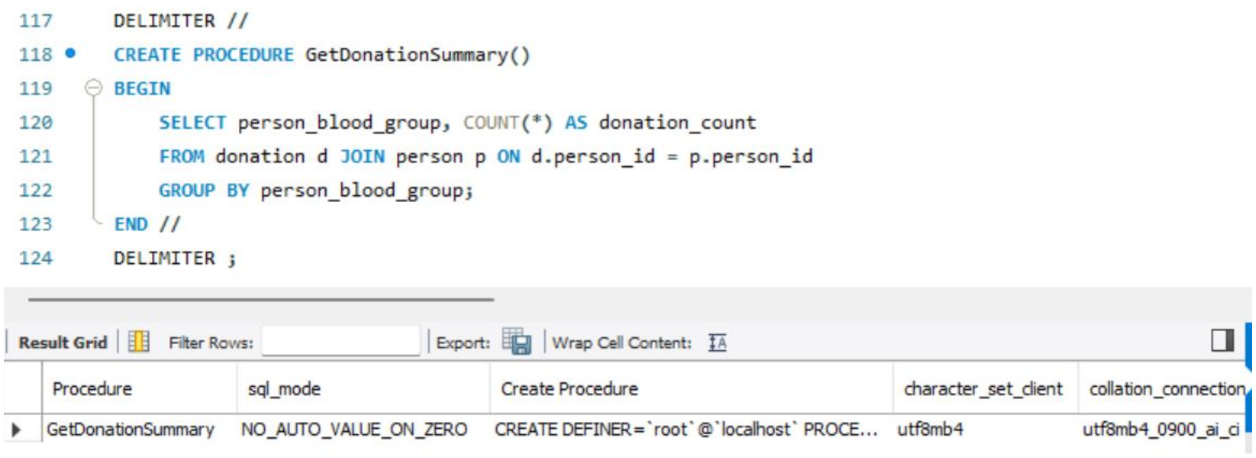
Procedure	sql_mode	Create Procedure	character_set_client	collation_connection	Data Colla
DeletePerson	NO_AUTO_VALUE_ON_ZERO	CREATE DEFINER=`root`@`localhost` PROCE...	utf8mb4	utf8mb4_0900_ai_ci	utf8m

11. Get Donation Summary
Aggregates donations by blood type categories.
Business Use: Strategic collection planning.
Admin Use: Blood group utilization analytics.

```
DELIMITER //

CREATE PROCEDURE GetDonationSummary()

BEGIN

    SELECT person_blood_group, COUNT(*) AS donation_count

    FROM donation d JOIN person p ON d.person_id = p.person_id

    GROUP BY person_blood_group;
```

END //

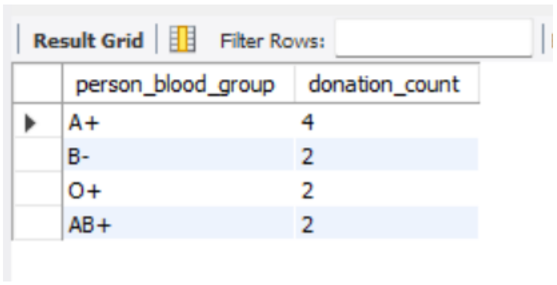
DELIMITER ;



```
DELIMITER //
CREATE PROCEDURE GetDonationSummary()
BEGIN
    SELECT person_blood_group, COUNT(*) AS donation_count
    FROM donation d JOIN person p ON d.person_id = p.person_id
    GROUP BY person_blood_group;
END //
DELIMITER ;
```

Procedure	sql_mode	Create Procedure	character_set_client	collation_connection
GetDonationSummary	NO_AUTO_VALUE_ON_ZERO	CREATE DEFINER=`root`@`localhost` PROCE...	utf8mb4	utf8mb4_0900_ai_ci

CALL GetDonationSummary();



person_blood_group	donation_count
A+	4
B-	2
O+	2
AB+	2

12. Check Blood Shortages
Identifies blood types below minimum thresholds.
Business Use: Triggers emergency collection campaigns.
Admin Use: Inventory risk management and forecasting.

DELIMITER //

CREATE PROCEDURE GetBloodShortages(IN min_threshold INT)

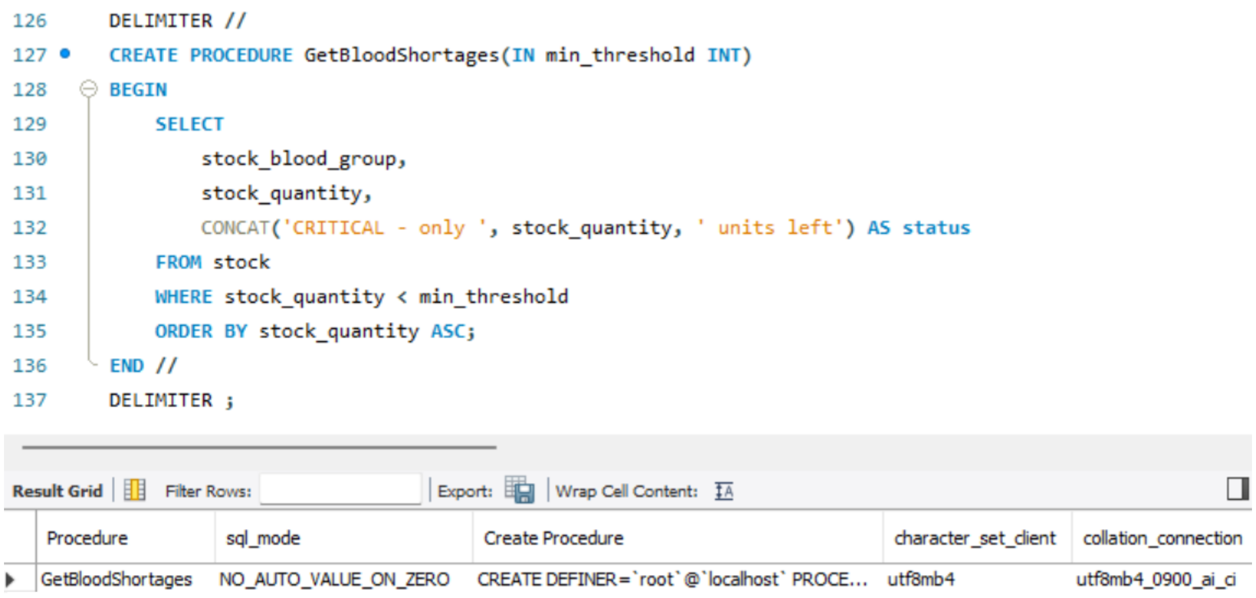
BEGIN

    SELECT

        stock_blood_group,

        stock_quantity,

```
        CONCAT('CRITICAL - only ', stock_quantity, ' units left') AS status
    FROM stock
    WHERE stock_quantity < min_threshold
    ORDER BY stock_quantity ASC;
END //
DELIMITER ;
```

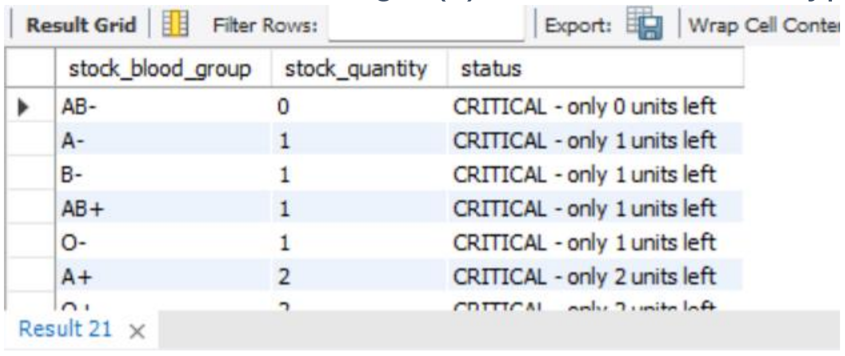


```
DELIMITER //
CREATE PROCEDURE GetBloodShortages(IN min_threshold INT)
BEGIN
    SELECT
        stock_blood_group,
        stock_quantity,
        CONCAT('CRITICAL - only ', stock_quantity, ' units left') AS status
    FROM stock
    WHERE stock_quantity < min_threshold
    ORDER BY stock_quantity ASC;
END //
DELIMITER ;
```

Procedure	sql_mode	Create Procedure	character_set_client	collation_connection
GetBloodShortages	NO_AUTO_VALUE_ON_ZERO	CREATE DEFINER=`root`@`localhost` PROCE...	utf8mb4	utf8mb4_0900_ai_ci

Identifies blood groups with stock below minimum threshold (e.g., <5 units).

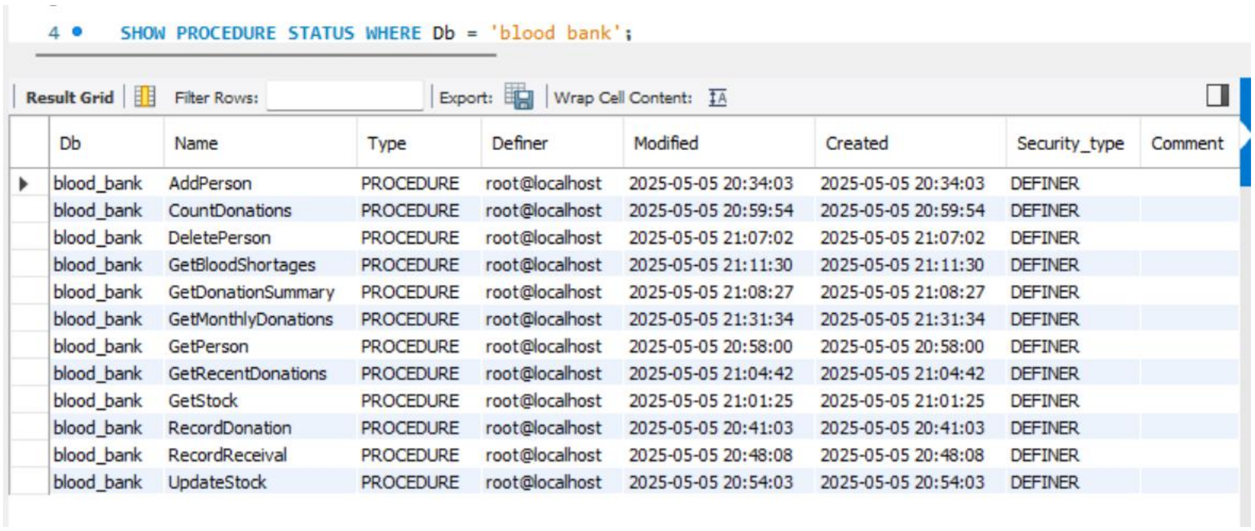
CALL GetBloodShortages(5); -- Finds all blood types with less than 5 units



stock_blood_group	stock_quantity	status
AB-	0	CRITICAL - only 0 units left
A-	1	CRITICAL - only 1 units left
B-	1	CRITICAL - only 1 units left
AB+	1	CRITICAL - only 1 units left
O-	1	CRITICAL - only 1 units left
A+	2	CRITICAL - only 2 units left
O+	2	CRITICAL - only 2 units left

Finally, we are calling all the stored procedures in our database

SHOW PROCEDURE STATUS WHERE Db = 'blood_bank';



```
SHOW PROCEDURE STATUS WHERE Db = 'blood bank';
```

Db	Name	Type	Definer	Modified	Created	Security_type	Comment
blood_bank	AddPerson	PROCEDURE	root@localhost	2025-05-05 20:34:03	2025-05-05 20:34:03	DEFINER	
blood_bank	CountDonations	PROCEDURE	root@localhost	2025-05-05 20:59:54	2025-05-05 20:59:54	DEFINER	
blood_bank	DeletePerson	PROCEDURE	root@localhost	2025-05-05 21:07:02	2025-05-05 21:07:02	DEFINER	
blood_bank	GetBloodShortages	PROCEDURE	root@localhost	2025-05-05 21:11:30	2025-05-05 21:11:30	DEFINER	
blood_bank	GetDonationSummary	PROCEDURE	root@localhost	2025-05-05 21:08:27	2025-05-05 21:08:27	DEFINER	
blood_bank	GetMonthlyDonations	PROCEDURE	root@localhost	2025-05-05 21:31:34	2025-05-05 21:31:34	DEFINER	
blood_bank	GetPerson	PROCEDURE	root@localhost	2025-05-05 20:58:00	2025-05-05 20:58:00	DEFINER	
blood_bank	GetRecentDonations	PROCEDURE	root@localhost	2025-05-05 21:04:42	2025-05-05 21:04:42	DEFINER	
blood_bank	GetStock	PROCEDURE	root@localhost	2025-05-05 21:01:25	2025-05-05 21:01:25	DEFINER	
blood_bank	RecordDonation	PROCEDURE	root@localhost	2025-05-05 20:41:03	2025-05-05 20:41:03	DEFINER	
blood_bank	RecordReceival	PROCEDURE	root@localhost	2025-05-05 20:48:08	2025-05-05 20:48:08	DEFINER	
blood_bank	UpdateStock	PROCEDURE	root@localhost	2025-05-05 20:54:03	2025-05-05 20:54:03	DEFINER	

- Data Manipulation Language(DML) Triggers

1. Automatic Stock Update on Donation
Function: Automatically increases blood stock when a donation is recorded
Importance: Ensures inventory is always up-to-date without manual updates
Example: When 1 unit of A+ blood is donated, the A+ stock increases by 1

This trigger **automatically updates blood stock levels** whenever a new donation is recorded.

How It Works:

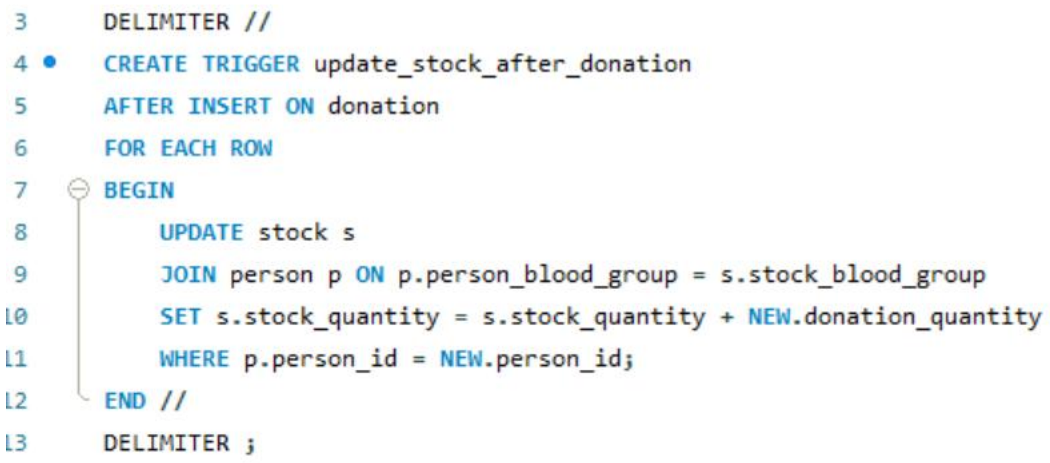
1. **Activates After Insertion** – Runs right after a new row is added to the donation table.
2. **Finds Donor's Blood Group** – Joins the person table to get the donor's blood type.
3. **Updates Stock** – Increases the matching blood group's quantity in the stock table by the donated amount.

Example:

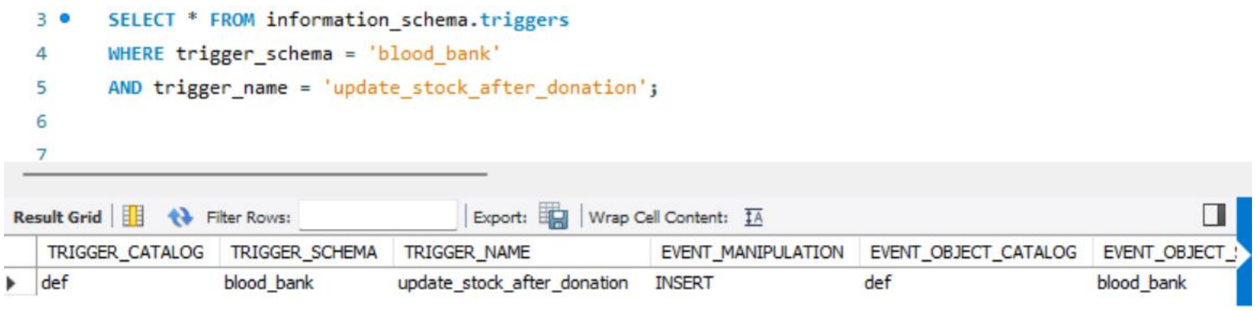
- If **Person ID 5 (Blood: A+)** donates **2 units**, the trigger:
    - Checks their blood type (A+).
    - Adds **2** to the **A+ stock count**.

```
DELIMITER //

CREATE TRIGGER update_stock_after_donation

AFTER INSERT ON donation

FOR EACH ROW

BEGIN
```

```
    UPDATE stock s
    JOIN person p ON p.person_blood_group = s.stock_blood_group
    SET s.stock_quantity = s.stock_quantity + NEW.donation_quantity
    WHERE p.person_id = NEW.person_id;
END //
DELIMITER ;
```



```
DELIMITER //
CREATE TRIGGER update_stock_after_donation
AFTER INSERT ON donation
FOR EACH ROW
BEGIN
    UPDATE stock s
    JOIN person p ON p.person_blood_group = s.stock_blood_group
    SET s.stock_quantity = s.stock_quantity + NEW.donation_quantity
    WHERE p.person_id = NEW.person_id;
END //
DELIMITER ;
```



```
SELECT * FROM information_schema.triggers
WHERE trigger_schema = 'blood_bank'
AND trigger_name = 'update_stock_after_donation';
```

TRIGGER_CATALOG	TRIGGER_SCHEMA	TRIGGER_NAME	EVENT_MANIPULATION	EVENT_OBJECT_CATALOG	EVENT_OBJECT_
def	blood_bank	update_stock_after_donation	INSERT	def	blood_bank

2. Prevent Invalid Donation Quantity

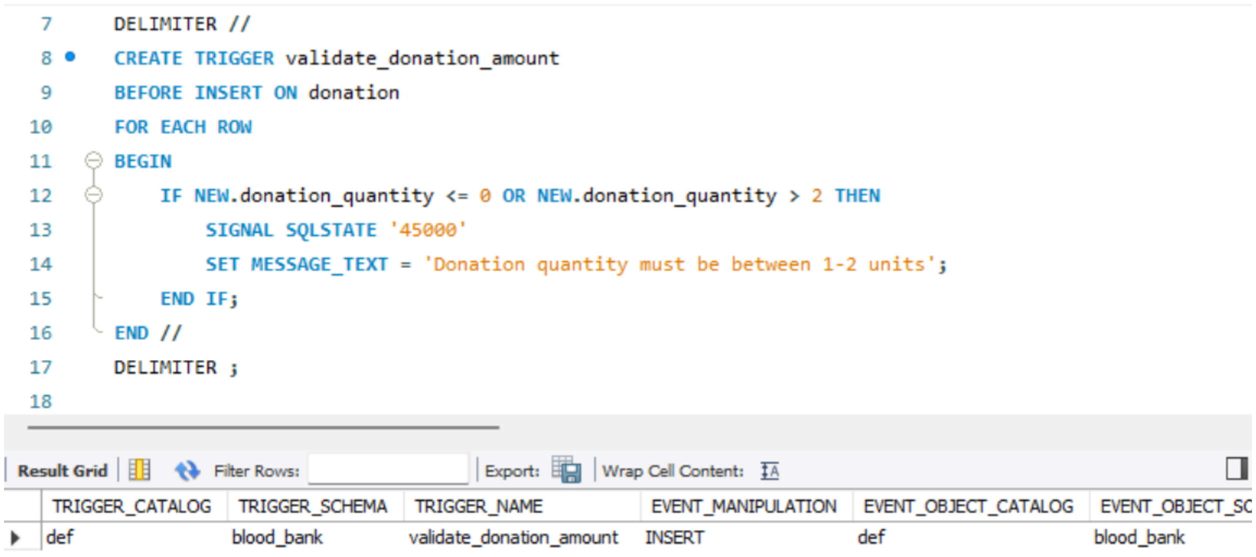
Function: Validates donation amounts before they're recorded
Importance: Ensures only realistic donation quantities (1-2 units) are stored
Example: Blocks insertion of a 5-unit donation which would be medically unsafe

This trigger **blocks unrealistic or unsafe blood donations** before they’re saved in the database.

How It Works:

1. **Activates Before Insertion** – Runs **before** a new donation is recorded.
2. **Checks Quantity Rules**:
    - **Rejects donations ≤ 0** (invalid)
    - **Rejects donations > 2 units** (medically unsafe)
3. **Throws an Error** if the rules are broken, stopping the invalid data from being saved.

```
DELIMITER //
CREATE TRIGGER validate_donation_amount
BEFORE INSERT ON donation
FOR EACH ROW
BEGIN
   IF NEW.donation_quantity <= 0 OR NEW.donation_quantity > 2 THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Donation quantity must be between 1-2
units';
   END IF;
END //
DELIMITER ;
```



```
DELIMITER //
CREATE TRIGGER validate_donation_amount
BEFORE INSERT ON donation
FOR EACH ROW
BEGIN
    IF NEW.donation_quantity <= 0 OR NEW.donation_quantity > 2 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Donation quantity must be between 1-2 units';
    END IF;
END //
DELIMITER ;
```

TRIGGER_CATALOG	TRIGGER_SCHEMA	TRIGGER_NAME	EVENT_MANIPULATION	EVENT_OBJECT_CATALOG	EVENT_OBJECT_SC
def	blood_bank	validate_donation_amount	INSERT	def	blood_bank

3. Track Last Donation Date
Function: Records when each donor last gave blood
Importance: Helps enforce 8-week minimum between donations
Example: After donation on 2023-10-15, the donor's record shows this date

1. **Activates After Insertion** – Runs *after* a new donation is recorded in the donation table.

2.Updates the Donor's Record – Sets the last_donation_date in the person table to the new donation's date.

Example Scenario:

- **Donor ID 5** gives blood on **May 20, 2024**.
- The trigger updates their record in the person table:

DELIMITER //

CREATE TRIGGER update_last_donation_date

AFTER INSERT ON donation

FOR EACH ROW

BEGIN

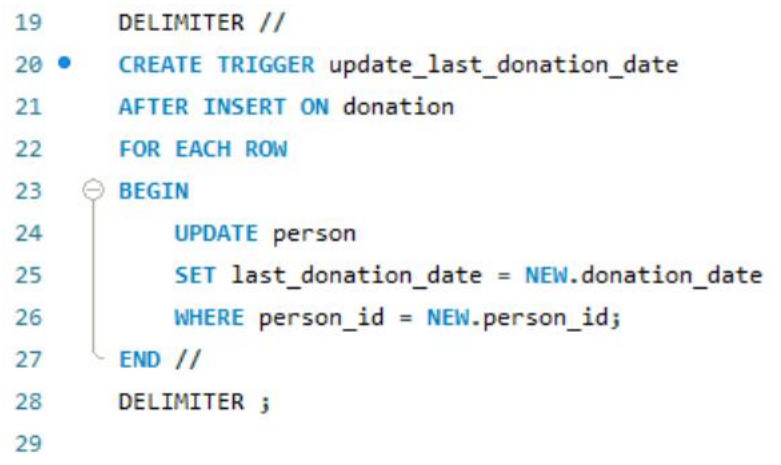
UPDATE person

SET last_donation_date = NEW.donation_date

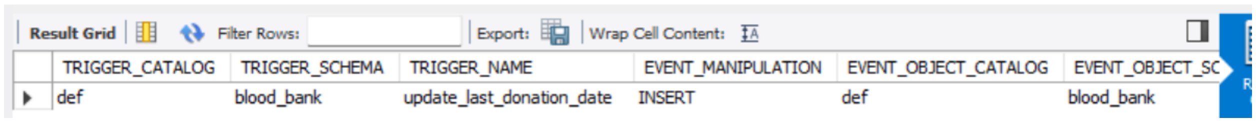
WHERE person_id = NEW.person_id;

END //

DELIMITER ;

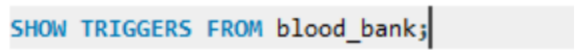


```
DELIMITER //
CREATE TRIGGER update_last_donation_date
AFTER INSERT ON donation
FOR EACH ROW
BEGIN
    UPDATE person
    SET last_donation_date = NEW.donation_date
    WHERE person_id = NEW.person_id;
END //
DELIMITER ;
```

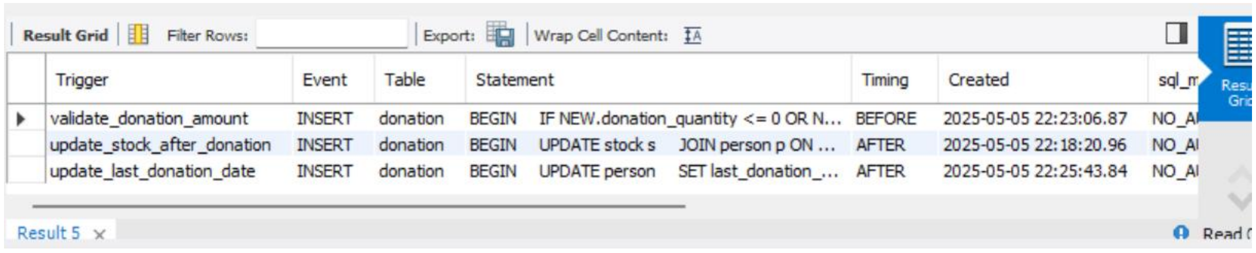


TRIGGER_CATALOG	TRIGGER_SCHEMA	TRIGGER_NAME	EVENT_MANIPULATION	EVENT_OBJECT_CATALOG	EVENT_OBJECT_SC
def	blood_bank	update_last_donation_date	INSERT	def	blood_bank

Here are all triggers existing in out database.
SHOW TRIGGERS FROM blood_bank;



```
SHOW TRIGGERS FROM blood_bank;
```



Trigger	Event	Table	Statement	Timing	Created	sql_m
validate_donation_amount	INSERT	donation	BEGIN IF NEW.donation_quantity <= 0 OR N...	BEFORE	2025-05-05 22:23:06.87	NO_A
update_stock_after_donation	INSERT	donation	BEGIN UPDATE stock s JOIN person p ON ...	AFTER	2025-05-05 22:18:20.96	NO_A
update_last_donation_date	INSERT	donation	BEGIN UPDATE person SET last_donation_...	AFTER	2025-05-05 22:25:43.84	NO_A

System

Technologies Used in the Blood Bank Management System

1. Database

- **MySQL** (Relational Database Management System - RDBMS)
  - Supports **SQL queries, triggers, and stored procedures** for automated operations.
  - Ensures **data integrity** with **primary and foreign key constraints**.
- **Dbeaver**
  - Used for storing and managing structured data (donor details, blood stock, transactions, etc.).

2. Frontend-Backend

- **PHP (Hypertext Preprocessor)**
  - Handles **server-side scripting**, database interactions, and business logic.
  - Processes form submissions (donor registration, blood donation/receipt records).
  - Implements **session management** for secure user authentication.
- **Apache HTTP Server** (via **XAMPP**)
  - Serves PHP files and processes HTTP requests.
  - Hosts the web application locally for development/testing.

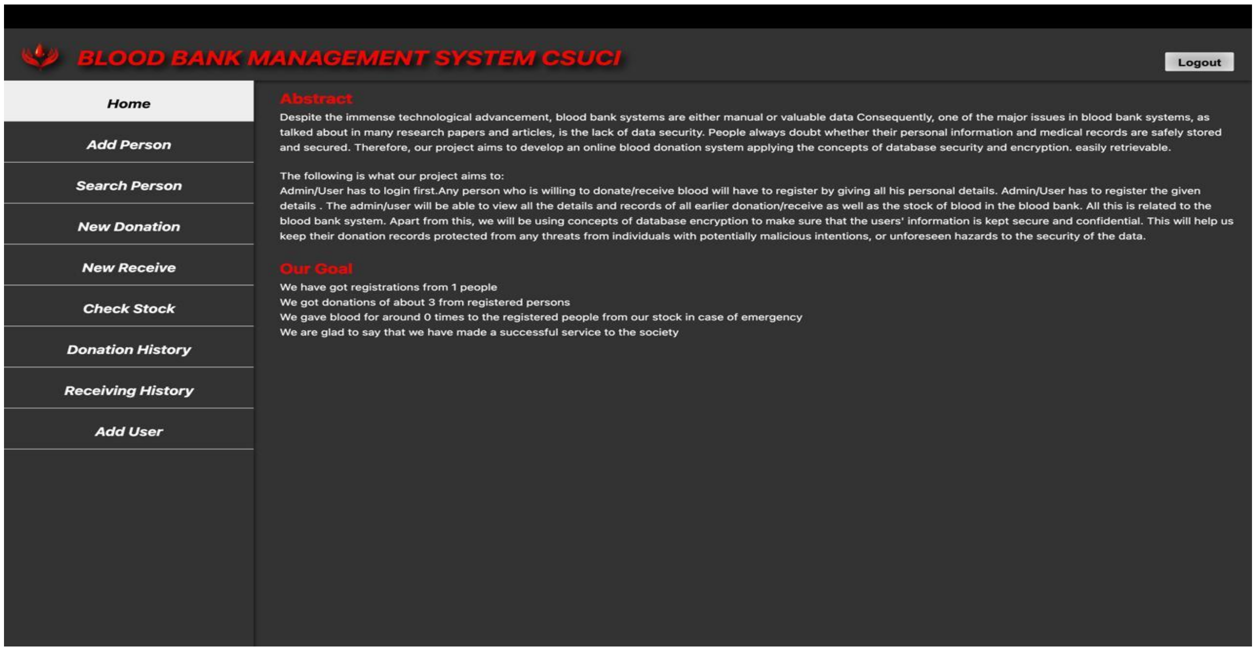
3. Development & Deployment Environment

- **XAMPP** (Cross-Platform Web Server Solution)
  - Includes **Apache (web server), MySQL (database), and PHP** for local development.
  - Allows testing the system before deployment on a live server.

There are few pictures of UI attached below.

1. Home Page

   In this page, we can view the **key statistics and features** of the Blood Bank Management System, including donor registrations, blood donations, and emergency distributions. It also highlights the system’s secure data handling and commitment to societal service



Abstract

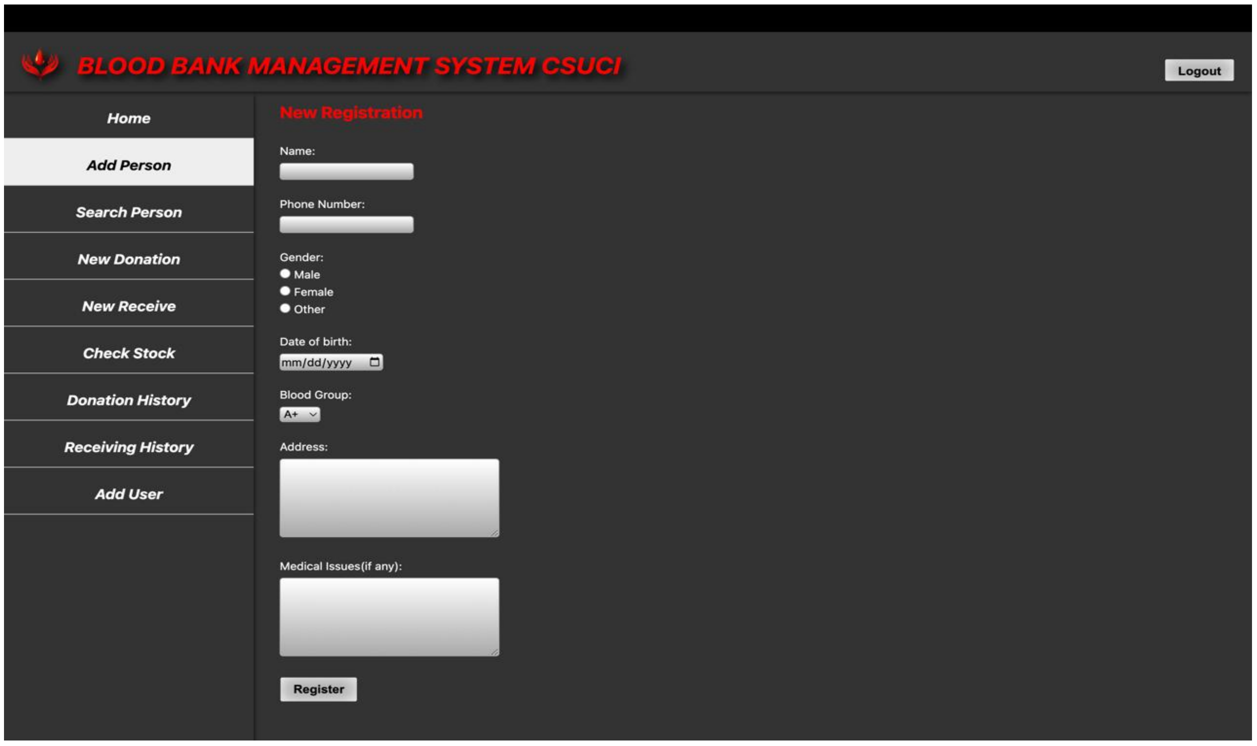
Despite the immense technological advancement, blood bank systems are either manual or valuable data Consequently, one of the major issues in blood bank systems, as talked about in many research papers and articles, is the lack of data security. People always doubt whether their personal information and medical records are safely stored and secured. Therefore, our project aims to develop an online blood donation system applying the concepts of database security and encryption. easily retrievable.

The following is what our project aims to:
Admin/User has to login first.Any person who is willing to donate/receive blood will have to register by giving all his personal details. Admin/User has to register the given details . The admin/user will be able to view all the details and records of all earlier donation/receive as well as the stock of blood in the blood bank. All this is related to the blood bank system. Apart from this, we will be using concepts of database encryption to make sure that the users' information is kept secure and confidential. This will help us keep their donation records protected from any threats from individuals with potentially malicious intentions, or unforeseen hazards to the security of the data.

Our Goal

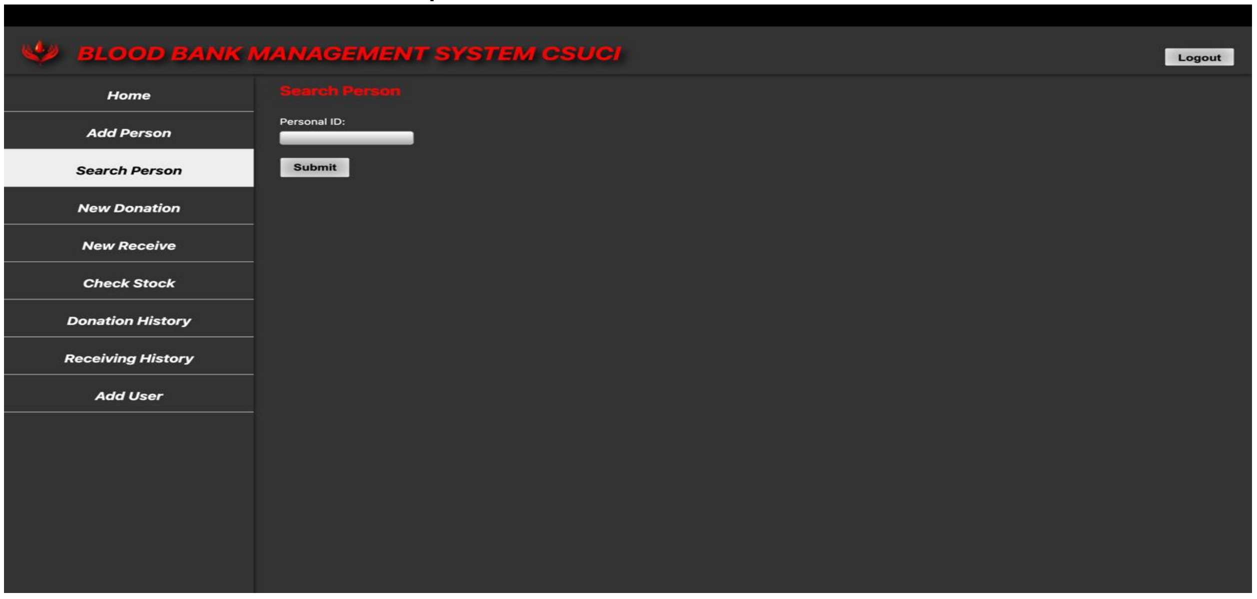
We have got registrations from 1 people
We got donations of about 3 from registered persons
We gave blood for around 0 times to the registered people from our stock in case of emergency
We are glad to say that we have made a successful service to the society

2. Add Person
   The right panel displays a donor registration form with fields for personal details, blood type selection, and medical disclosures. Its structured design ensures accurate data entry, while the "Register" button securely submits information to the database. This streamlined interface maintains data integrity and simplifies the donor process.



3.Search Person

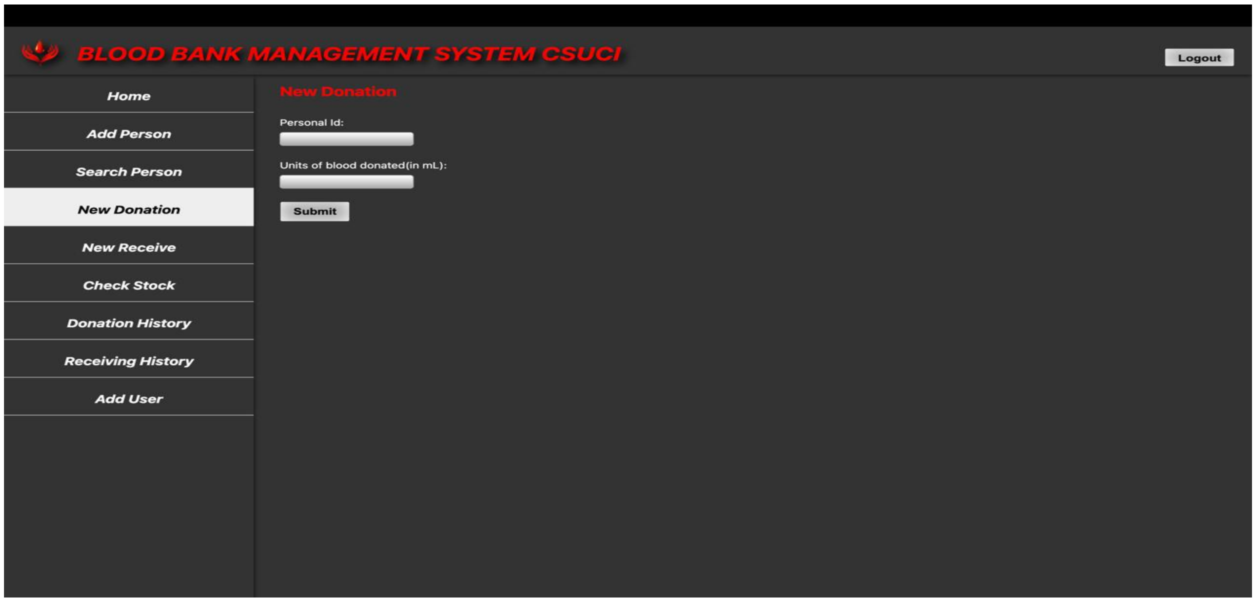
In this we can search the person with Personal ID.



4.New Donation

A unique auto-incremented identifier assigned to each donor/recipient in the system, ensuring accurate record linkage across all blood bank operations (donations, receipts, and medical history).

A numeric field (typically 1-2 units) quantifying blood collected per donation, where 1 unit = 450mL. This standardized measurementtracks inventory and ensures compliance with safe donation limits.



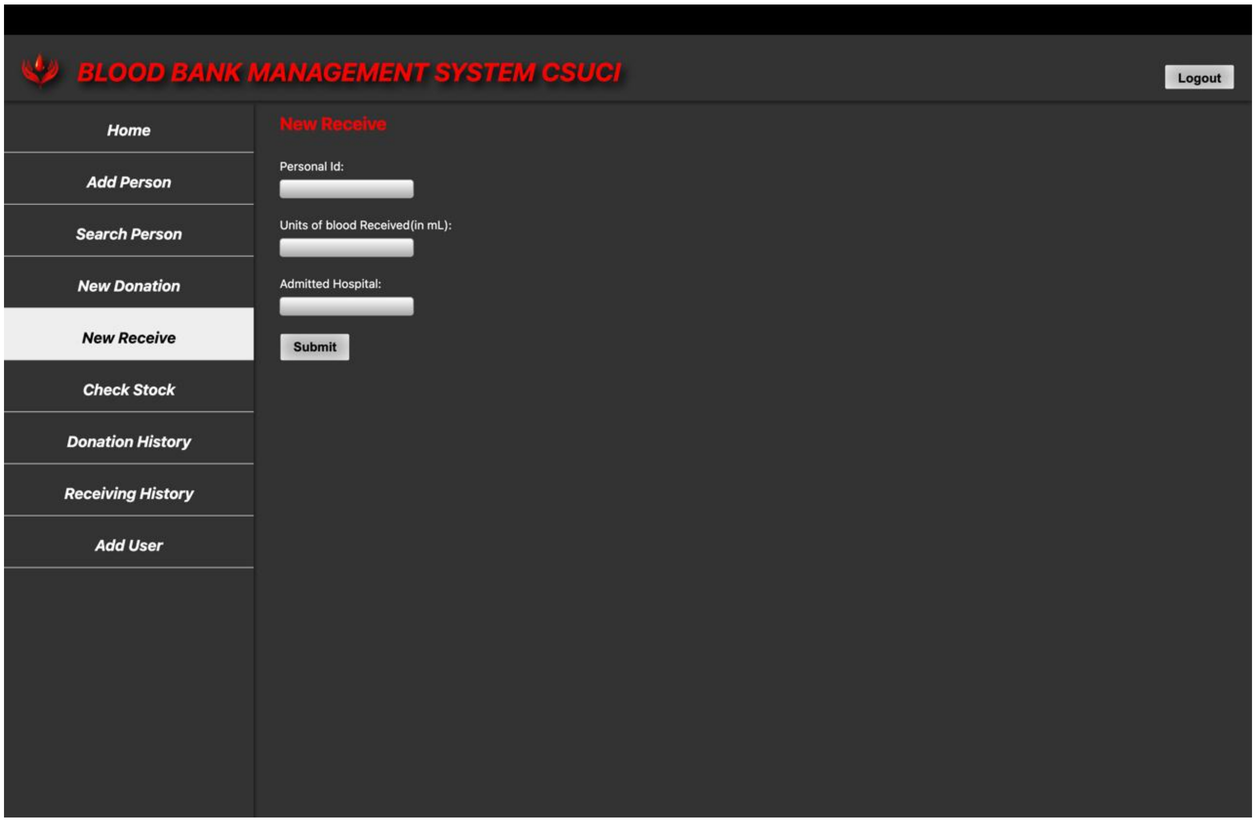
New Donation

Personal Id:

Units of blood donated(in mL):

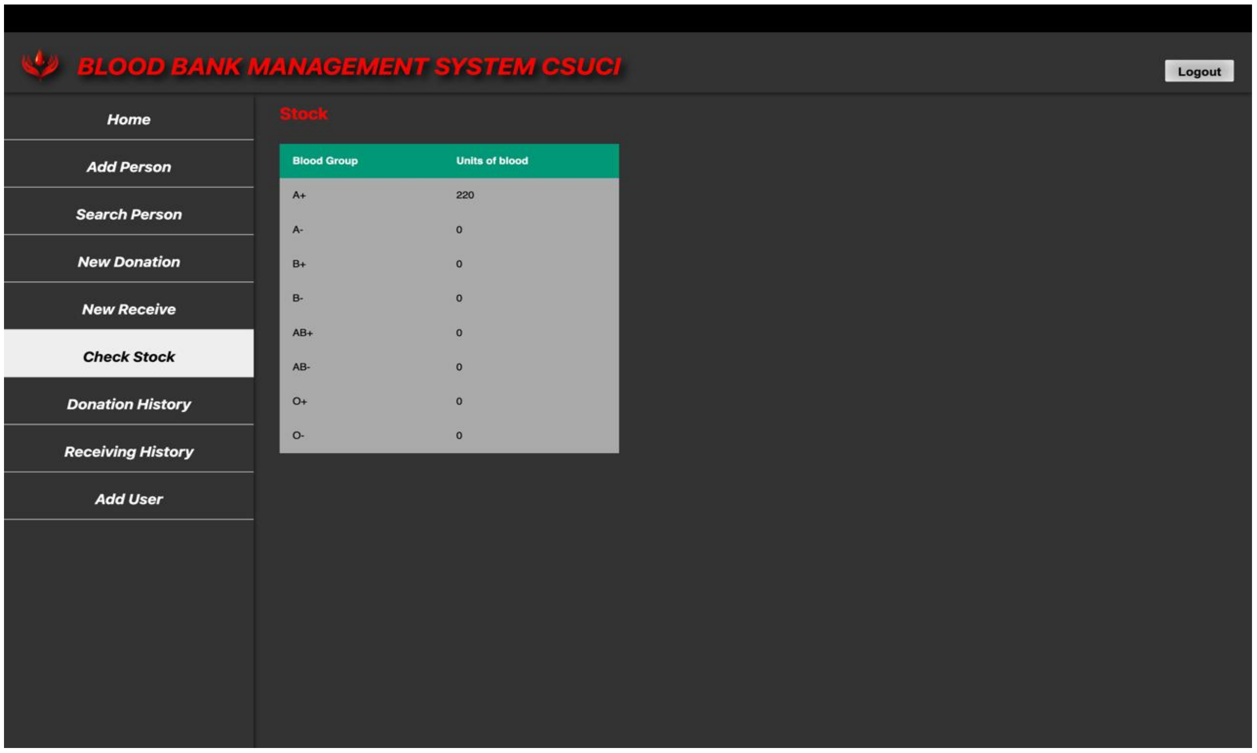
5.New receive

The **Personal ID** uniquely identifies each donor/recipient, while **Units of Blood Received** (1 unit = 450mL) tracks transfusion quantities, and **Admitted Hospital** records the destination facility – together ensuring traceable, compliant blood management. These fields enable precise inventory control and institutional accountability across all transactions.



6.Check Stock

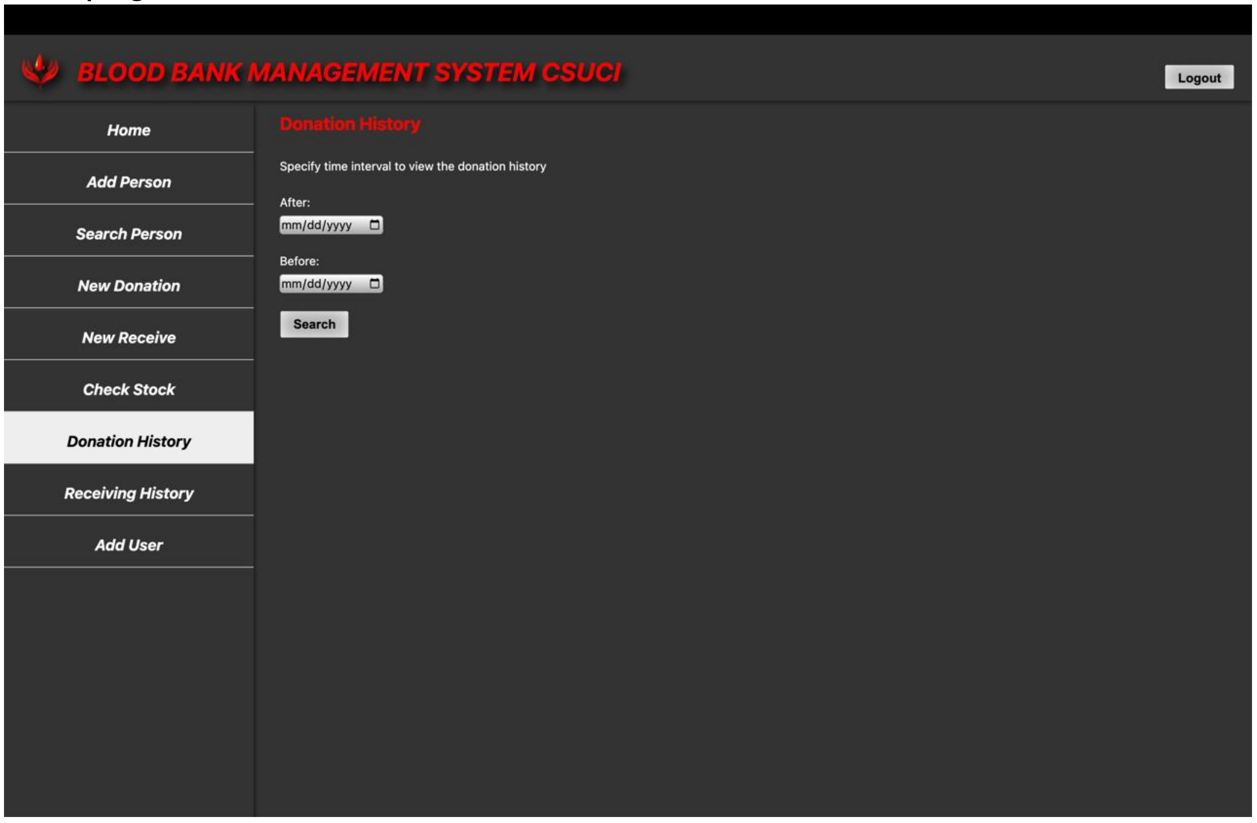
This page displays the **current blood inventory** in a clear table format, showing available units by blood group (e.g., A+: 200 units). It highlights critical shortages (0 units for most types) while providing real-time stock visibility for emergency planning. The minimalist design prioritizes quick readability for medical staff during urgent situations



Stock

Blood Group	Units of blood
A+	220
A-	0
B+	0
B-	0
AB+	0
AB-	0
O+	0
O-	0

7. Donation History
This page shows bloods donated



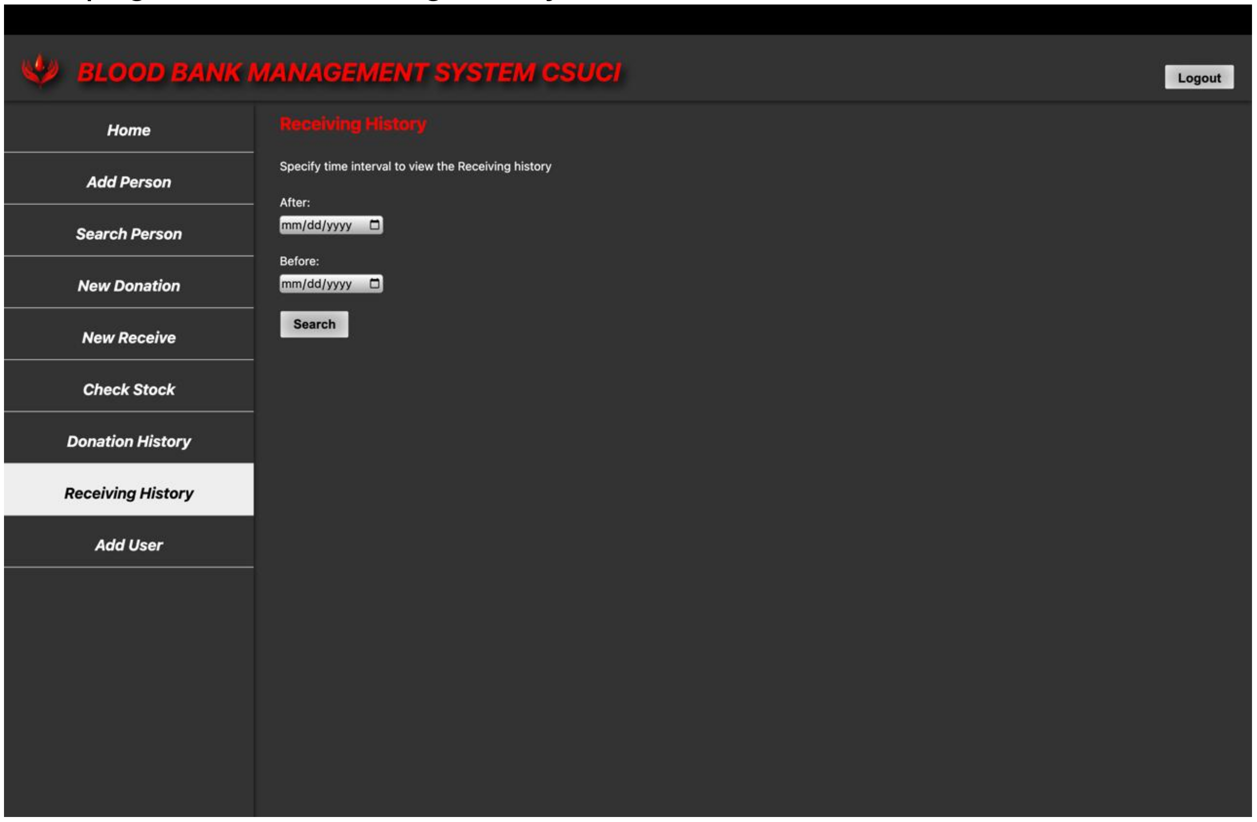
Donation History

Specify time interval to view the donation history

After: mm/dd/yyyy

Before: mm/dd/yyyy

8.Receiving History
This page shows receiving history



Receiving History

Specify time interval to view the Receiving history

After: mm/dd/yyyy

Before: mm/dd/yyyy

Conclusion and Future Scope

The main purpose of our blood management system is to provide blood bank with easier way to store and retrieve data and keep record of the availability of blood in blood bank.
After inserting the data to database, staff need not register of the same person again. They can simply search for recorded data and retrieve them for future blood donation or receiving purpose of that person.
In the nutshell, it can be summarized that the future scope of the project circles around maintaining information regarding:

- ❖ The person can fix their donation schedule using online reservation for donation of blood.
- ❖ The person can search for availability of required blood in the local blood bank in the case of emergency.
- ❖ The blood bank to store the details of the blood donated by person, like RBC, WBC, platelet count etc.

The above mentioned points are the enhancements which can be done to increase the applicability and usage of this project.

References

- Silberschatz Korth and Sudharshan, Database System Concepts, 6th Edition, McGraw Hill, 2013.
- Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.
- Database management systems, Ramakrishna, and Gehrke, 3rd Edition, 2014, McGrawHill, 2013.
- Google
- https://www.w3schools.com