Final Report

Blood Bank Donation System

Group No: 2

Student Names:

Pamidi Balaji

Bhavesh Kulkarni

Executive Summary:

The Blood Bank Donation System is designed to streamline the management of blood donations, inventory tracking, hospital requests, and transfusion histories. This system efficiently handles donor registrations, evaluates health eligibility, monitors blood storage and inventory, and processes hospital requests while providing detailed reporting on system performance and resource utilization.

Using a relational database system like MySQL, the system manages structured data, while NoSQL databases like MongoDB handle unstructured data. Python is leveraged for data visualization, enabling comprehensive insights into inventory levels, donor activity, and request fulfilment.

Key entities include Donor, Blood Inventory, Blood Storage, Hospital, and Patient, modeled through EER and UML diagrams. Attributes such as Donor_ID, Blood_ID, and Storage_ID facilitate precise tracking and integration across various modules. This structure ensures seamless operations by linking blood donations to inventory, associating inventory with storage facilities, and fulfilling hospital requests in real-time.

The system addresses critical challenges such as maintaining adequate blood supply, optimizing storage capacities, and monitoring transfusion records. Recommendations include rigorous testing before deployment, regular updates to improve efficiency, and potential integration with external systems for enhanced functionality.

Overall, the Blood Bank Donation System enhances operational efficiency, improves resource management, and ensures better healthcare outcomes through effective blood bank operations.

I. Introduction

Blood donation is a vital act that saves lives by supplying vital blood supplies to patients and hospitals. However, overseeing blood donation procedures, which include donor registration, inventory control, and guaranteeing on-time delivery, is difficult and resource-intensive. Patient safety may be at risk due to delays, shortages, or waste caused by inefficiencies in various areas. In order to properly handle these complications, streamlined methods are required, as evidenced by problems like mismatched blood types or insufficient supplies during emergencies.

By providing a complete solution to automate and optimize blood donation procedures, the Blood Bank Donation System tackles these issues. It effectively handles hospital demands, keeps track of supplies, keeps precise donor records, and guarantees that blood is delivered and stored correctly. Through enhanced data management and communication between hospitals and blood banks, the system lowers the risk of waste and shortages. In order to ensure timely availability and save more lives, this project seeks to improve the effectiveness of blood donation services.

Background information:

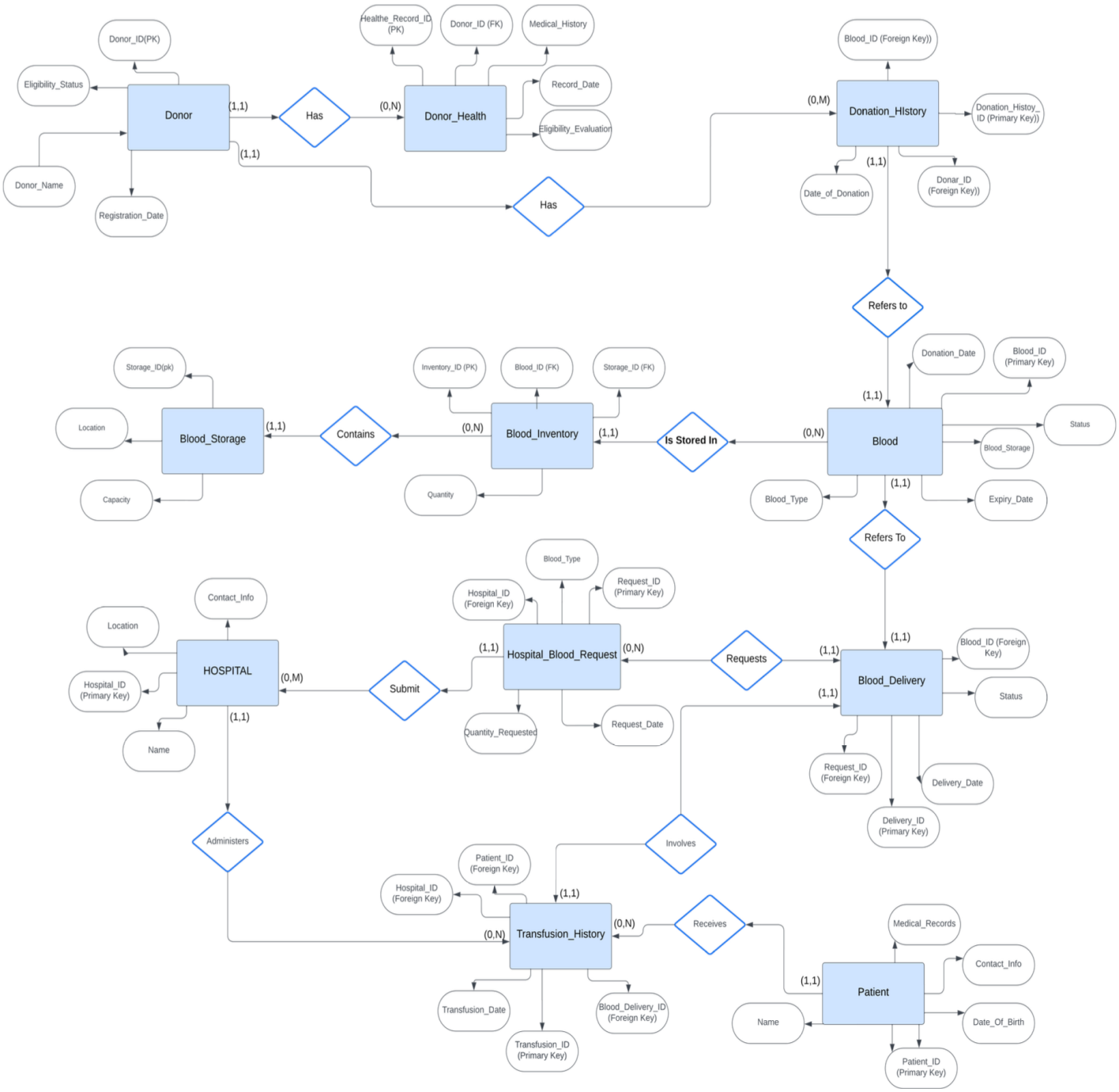
Blood donation is essential for saving lives, providing critical blood supplies to hospitals and patients in need. However, managing the blood donation process presents challenges, such as maintaining accurate donor records, managing inventory, ensuring blood type compatibility, and coordinating timely deliveries. Inefficiencies can lead to delayed emergency responses, insufficient stock, or wasted resources due to expired blood units, compromising healthcare delivery and posing risks to patient safety. A lack of available blood in critical situations can have severe consequences, emphasizing the need for a streamlined system to ensure proper utilization and availability of blood supplies when and where needed.

Other Requirements:

1. A donor can have multiple health records, but each health record is linked to one specific donor.
2. A donor can make multiple donations, but each donation is associated with one specific donor.
3. Each record in the donation history corresponds to one blood unit, and each blood unit is linked to one donation record.
4. Each blood unit is included in a single inventory record, but an inventory can hold multiple blood units.
5. A storage location can hold multiple inventory units, but each inventory unit is stored in a single location.
6. A hospital can submit multiple blood requests, but each request is tied to one specific hospital.
7. A single blood request can result in multiple deliveries, but each delivery is tied to one specific request.
8. Each blood delivery involves one specific blood unit, but a blood unit can be part of multiple deliveries.
9. A hospital can conduct multiple transfusions, but each transfusion is associated with one specific hospital.
10. A patient may receive multiple transfusions, but each transfusion is tied to one specific patient.
11. Each transfusion corresponds to one specific blood delivery, but one blood delivery can serve multiple transfusions.

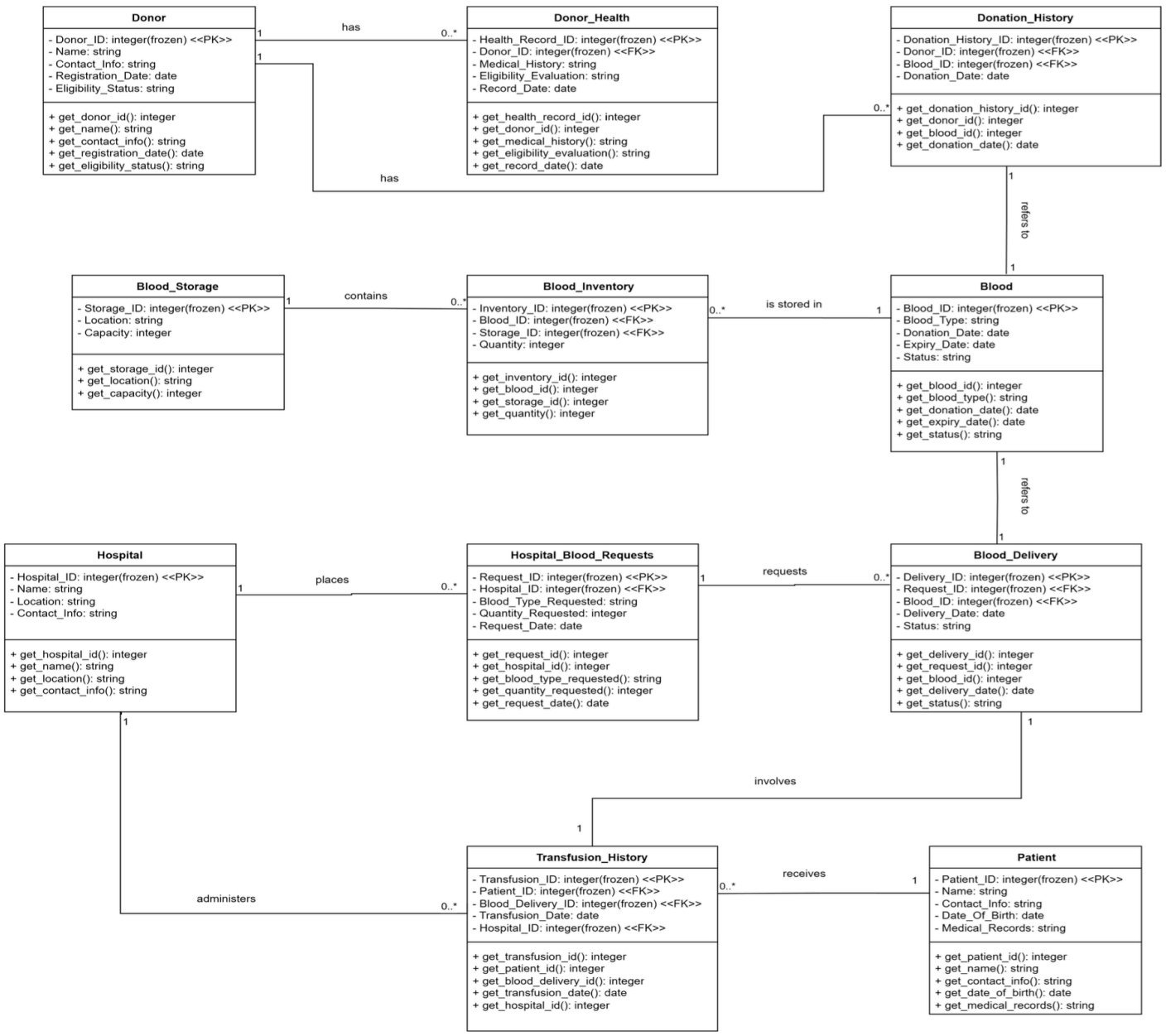
II. Conceptual Data Modeling

EER Diagram:



UML Diagram:

UML



III. Mapping Conceptual Model to Relational Model

DONOR(Donor_ID (PK),Name,Contact_Info,Registration_Date,Eligibility_Status)

DONOR_HEALTH(Health_Record_ID (PK),Donor_ID (FK),Medical_History,Eligibility_Evaluation, Record_Date)
• FOREIGN KEY Donor_ID REFERENCES DONOR(Donor_ID) NOT NULL

DONATION_HISTORY(Donation_History_ID (PK),Donor_ID (FK),Blood_ID (FK),Donation_Date)
• FOREIGN KEY Donor_ID REFERENCES DONOR(Donor_ID) NOT NULL
• FOREIGN KEY Blood_ID REFERENCES BLOOD(Blood_ID) NOT NULL

BLOOD(Blood_ID (PK),Blood_Type,Donation_Date,Expiry_Date,Status)
BLOOD_INVENTORY(Inventory_ID (PK),Blood_ID (FK),Storage_ID (FK),Quantity)
• FOREIGN KEY Blood_ID REFERENCES BLOOD(Blood_ID) NOT NULL
• FOREIGN KEY Storage_ID REFERENCES BLOOD_STORAGE(Storage_ID) NOT NULL

BLOOD_STORAGE(Storage_ID (PK),Location,Capacity)

HOSPITAL(Hospital_ID (PK),Name,Location,Contact_Info)

HOSPITAL_BLOOD_REQUESTS(Request_ID (PK),Hospital_ID (FK),Blood_Type_Requested, Quantity_Requested,Request_Date)
• FOREIGN KEY Hospital_ID REFERENCES HOSPITAL(Hospital_ID) NOT NULL

BLOOD_DELIVERY(Delivery_ID (PK),Request_ID (FK),Blood_ID (FK),Delivery_Date,Status (Delivered, Pending))
• FOREIGN KEY Request_ID REFERENCES HOSPITAL_BLOOD_REQUESTS(Request_ID) NOT NULL
• FOREIGN KEY Blood_ID REFERENCES BLOOD(Blood_ID) NOT NULL**PATIENT**(Patient_ID (PK),Name,Contact_Info,Date_Of_Birth,Medical_Records)

TRANSFUSION_HISTORY(Transfusion_ID (PK),Patient_ID (FK),Blood_Delivery_ID (FK),Hospital_ID (FK),Transfusion_Date)
• FOREIGN KEY Patient_ID REFERENCES PATIENT(Patient_ID) NOT NULL
• FOREIGN KEY Blood_Delivery_ID REFERENCES BLOOD_DELIVERY(Delivery_ID) NOT NULL
• FOREIGN KEY Hospital_ID REFERENCES HOSPITAL(Hospital_ID) NOT NULL

IV. Implementation of Relation Model via MySQL and NoSQL

MySQL Implementation :

The database was created in MySQL and the following queries were performed :

Query 1: Find the details of blood donations made by eligible donors.

```
SELECT d.Name, b.Blood_Type, dh.Donation_Date
FROM DONOR d
JOIN DONATION_HISTORY dh ON d.Donor_ID =
dh.Donor_ID
JOIN BLOOD b ON dh.Blood_ID = b.Blood_ID
WHERE d.Eligibility_Status = 'Eligible';
```

Name	Blo...	Donation_Date
Alice Johnson	A+	2023-01-20
Bob Smith	O-	2023-02-25
Diana Prince	AB+	2023-04-10
Edward Elric	A-	2023-05-05
Fiona Gallagher	O+	2023-06-20
Hannah Baker	AB-	2023-08-30
Isaac Clarke	A+	2023-09-15
Jenna Ortega	O-	2023-10-01
Laura Croft	AB+	2023-12-15
Michael Scott	A-	2023-01-10
Nancy Drew	O+	2023-02-05
Oscar Isaac	B-	2023-03-20

Query 2: Count of Eligible vs. Not Eligible Donors

```
SELECT Eligibility_Status, COUNT(*) AS Donor_Cou
FROM DONOR GROUP BY Eligibility_Status;
```

Eligibility_Status	Donor_Count
Eligible	12
Not Eligible	3

Query 3: Retrieve donor names and the blood types they have donated

```
SELECT d.Name AS Donor_Name, b.Blood_Type, dh.Donation_Date
FROM DONOR d
INNER JOIN DONATION_HISTORY dh ON d.Donor_ID = dh.Donor_ID
INNER JOIN BLOOD b ON dh.Blood_ID = b.Blood_ID;
```

Donor_Name	Blo...	Donation_Date
Alice Johnson	A+	2023-01-20
Bob Smith	O-	2023-02-25
Charlie Brown	B+	2023-03-15
Diana Prince	AB+	2023-04-10
Edward Elric	A-	2023-05-05
Fiona Gallagher	O+	2023-06-20
George Weasley	B-	2023-07-25
Hannah Baker	AB-	2023-08-30
Isaac Clarke	A+	2023-09-15
Jenna Ortega	O-	2023-10-01
Kevin Hart	B+	2023-11-20
Laura Croft	AB+	2023-12-15
Michael Scott	A-	2023-01-10
Nancy Drew	O+	2023-02-05
Oscar Isaac	B-	2023-03-20

Query 4: Find The Donors Who Donated Blood with Type 'A+'

```
SELECT Name
FROM DONOR
WHERE Donor_ID IN (
   SELECT Donor_ID
   FROM DONATION_HISTORY
   WHERE Blood_ID IN (
      SELECT Blood_ID
      FROM BLOOD
      WHERE Blood_Type = 'A+'
   )
);
```

Name
Alice Johnson
Isaac Clarke

Query 5: Find Blood Types with Donations More Than the Average Quantity

```
SELECT Blood_Type
FROM BLOOD b
WHERE (
   SELECT SUM(Quantity)
   FROM BLOOD_INVENTORY bi
   WHERE bi.Blood_ID = b.Blood_ID
) > (
   SELECT AVG(SUM_Quantity)
   FROM (
      SELECT SUM(Quantity) AS SUM_Quantity
      FROM BLOOD_INVENTORY
      GROUP BY Blood_ID
   ) AS SubQuery
);
```

Blood_Type
A+
O-
B+

Query 6: Find hospitals that have not made any blood requests.

```
SELECT Name
FROM HOSPITAL h
WHERE NOT EXISTS (
   SELECT 1
   FROM HOSPITAL_BLOOD_REQUESTS hbr
   WHERE hbr.Hospital_ID = h.Hospital_ID
);
```

Name
South End Community Health Center
Harborview Medical Center
Boston Children's Hospital
Brigham and Women's Hospital
Massachusetts General Hospital
Boston Medical Center
South End Community Health Center

Query 7: Get a list of all locations (both blood storage locations and hospital locations), including duplicates.

```
SELECT Location
FROM BLOOD_STORAGE
UNION ALL
SELECT Location
FROM HOSPITAL;
```

Location
Central Blood Bank
Northside Clinic
Eastside Hospital
Westside Health Center
City General Hospital
Downtown
Northside
Eastside
Westside
Uptown
Boston, MA
South Boston
Longwood Medical Area
Mission Hill
West End
South End
Boston, MA

Query 8: Find the hospitals with the highest quantity of blood requests.

```
SELECT h. Name, h.Location, hr.Total_Requested
FROM HOSPITAL h
JOIN (
    SELECT Hospital_ID, SUM(Quantity_Requested) AS Total_Requested
    FROM HOSPITAL_BLOOD_REQUESTS
    GROUP BY Hospital_ID
) hr ON h.Hospital_ID = hr.Hospital_ID
ORDER BY hr.Total_Requested DESC;
```

Name	Location	Total_Request...
City Hospital	Downtown	10
Eastside Community Hospital	Eastside	8
County Hospital	Uptown	6
Northside Medical Center	Northside	5
Westside General Hospital	Westside	4

NoSQL Implementation:

Query 1: Get the top 3 donors with the earliest registration dates

```
db.donor.find().sort({ Registration_Date: 1 }).
limit(3);
```


```
{
  _id: ObjectId('673f722cc2584a3c7e8b0fbc'),
  Donor_ID: 1,
  Name: 'Alice Johnson',
  Contact_Info: 'alice.j@example.com',
  Registration_Date: 2023-01-15T00:00:00.000Z,
  Eligibility_Status: 'Eligible'
}
{
  _id: ObjectId('673f722cc2584a3c7e8b0fbd'),
  Donor_ID: 2,
  Name: 'Bob Smith',
  Contact_Info: 'bob.smith@example.com',
  Registration_Date: 2023-02-20T00:00:00.000Z,
  Eligibility_Status: 'Eligible'
}
{
  _id: ObjectId('673f722cc2584a3c7e8b0fbe'),
  Donor_ID: 3,
  Name: 'Charlie Brown',
  Contact_Info: 'charlie.b@example.com',
  Registration_Date: 2023-03-10T00:00:00.000Z,
  Eligibility_Status: 'Not Eligible'
}
```

Query 2: To Fetch all blood donations made in 2023, including donor name, blood type, and hospital name.

```
db.donation_history.find({
  Donation_Date: { $gte: ISODate("2023-01-01T00:00:00Z"), $lt: ISODate("2024-01-01T00:00:00Z") }
})
  .forEach(function(donation) {
    var donor = db.donor.findOne({ Donor_ID: donation.Donor_ID });
    var blood = db.blood.findOne({ Donor_ID: donation.Blood_ID });
    var hospitalRequests = db.hospital_blood_requests.find
({ Blood_Type_Requested: blood.Blood_Type });
    print("Donation Date: " + donation.Donation_Date);
    print("Donor Name: " + donor.Name);
    print("Blood Type: " + blood.Blood_Type);

    hospitalRequests.forEach(function(request) {
      var hospital = db.hospital.findOne({ Hospital_ID:
request.Hospital_ID });
      print("Hospital: " + hospital.Name);
    });
    print("\n----------------------------------------\n");
  });
```

```
Donation Date: Thu Jan 19 2023 19:00:00 GMT-0500 (Eastern Standard Time)
Donor Name: Alice Johnson
Blood Type: A+

----------------------------------------
Donation Date: Fri Feb 24 2023 19:00:00 GMT-0500 (Eastern Standard Time)
Donor Name: Bob Smith
Blood Type: O-

----------------------------------------
Donation Date: Tue Mar 14 2023 20:00:00 GMT-0400 (Eastern Daylight Time)
Donor Name: Charlie Brown
Blood Type: B+

----------------------------------------
Donation Date: Sun Apr 09 2023 20:00:00 GMT-0400 (Eastern Daylight Time)
Donor Name: Diana Prince
Blood Type: AB+

----------------------------------------
Donation Date: Thu May 04 2023 20:00:00 GMT-0400 (Eastern Daylight Time)
Donor Name: Edward Elric
Blood Type: A-
```

Query 3: To find the total number of donations made by each donor:

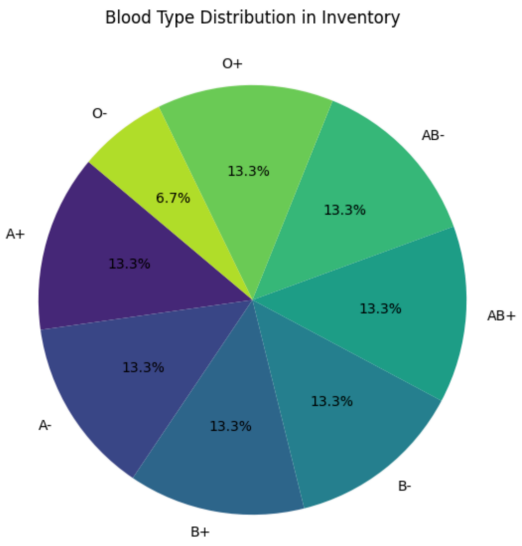
```
db.donation_history.aggregate([
 {
   $group: {
     _id: "$Donor_ID",    // Group by Donor_ID
     total_donations: { $sum: 1 }  // Count the number of donations per
 donor  }
 },{   $lookup: {
     from: "donor",             // Join with the donor collection
     localField: "_id",         // Field to match with Donor_ID
     foreignField: "Donor_ID",   // Field in donor collection
     as: "donor_info"           // Add donor information
   }},{   $unwind: "$donor_info"       // Unwind the donor_info array
 },{  $project: {
     Donor_Name: "$donor_info.Name",  // Get donor's name
     Total_Donations: "$total_donations",  // Total donations made by the
donor
     _id: 0   // Exclude the default _id field
   }
 }]);
```

```
{
  Donor_Name: 'Diana Prince',
  Total_Donations: 1
}
{
  Donor_Name: 'Alice Johnson',
  Total_Donations: 1
}
{
  Donor_Name: 'Bob Smith',
  Total_Donations: 1
}
{
  Donor_Name: 'Charlie Brown',
  Total_Donations: 1
}
{
  Donor_Name: 'Edward Elric',
  Total_Donations: 1
}
```

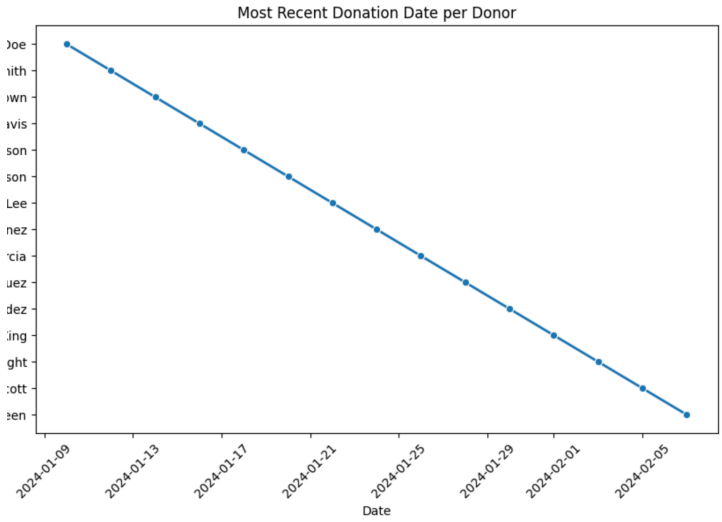
V. Database Access Via Python

The database is accessed using Python and visualization of analyzed data is shown below.
MySQL is connected to Python using MySQL.connector, followed by the cursor.
To execute to run and fetch all from the query, followed by converting the list into a data frame using pandas library and using matplotlib to plot the graphs for the analytics.

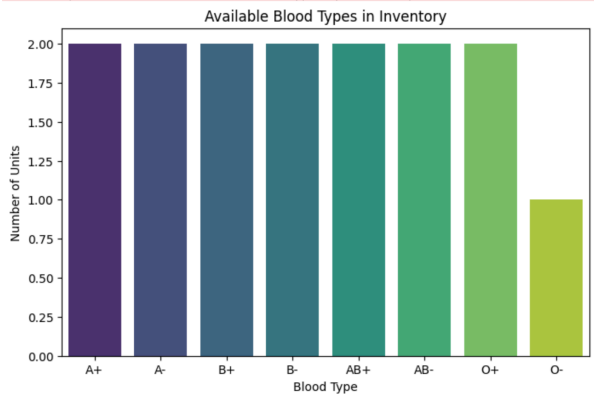
Graph 1:Blood Type Distribution



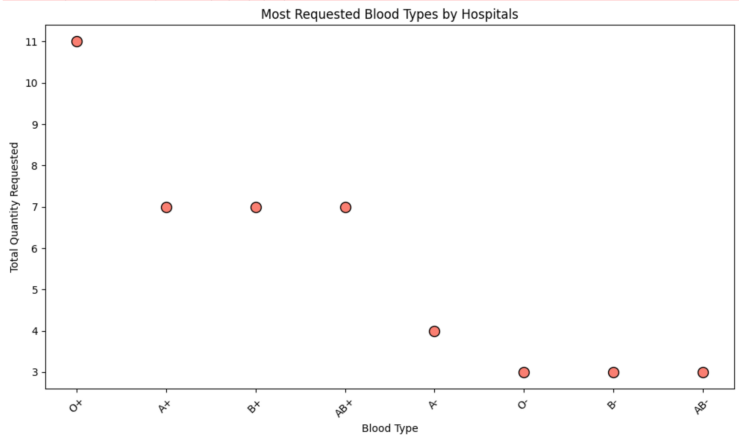
Graph 2: Most Recent Donation



Graph 3: Available Blood In Inventory



Graph 4: Most Requested Blood By Hospital



VII. Summary and Recommendation

The Blood Donation and Management System database, designed using MySQL, is a comprehensive and industry-ready relational database tailored for streamlining blood bank operations. It effectively manages donor information, blood inventory tracking, hospital requests, and transfusion history, ensuring data integrity and operational efficiency. The relational design reduces redundancies, automates key processes, and provides analytics capabilities for inventory optimization and demand forecasting. Visualizations created using Python further highlight its ability to provide actionable insights into donor activity, hospital demands, and inventory status.

To enhance the database, data governance measures should be implemented to maintain data quality, particularly for attributes like donor eligibility, blood type, and inventory status. Automation tools like Optical Character Recognition (OCR) can streamline data entry by validating donor information during registration. Additionally, predictive algorithms can help monitor and forecast blood demand trends, further improving resource management.

Explorations into implementing the database on MongoDB, a NoSQL graph database, revealed challenges due to the highly relational nature of the data. While graph-based systems are ideal for handling dynamic relationships, the real-time update requirements for attributes such as donation history and inventory make the relational model more suitable for this application. Additional research and testing are necessary to overcome these challenges and fully leverage NoSQL’s potential in such use cases.

In conclusion, the MySQL implementation of the database provides a robust and efficient solution for blood bank operations, demonstrating scalability and effectiveness in handling complex queries and processes. While future advancements could explore hybrid models combining relational and NoSQL features, the current design meets industry needs and offers significant value for real-world applications.