BLOOD AND ORGAN DONATION NETWORK

PROJECT REPORT

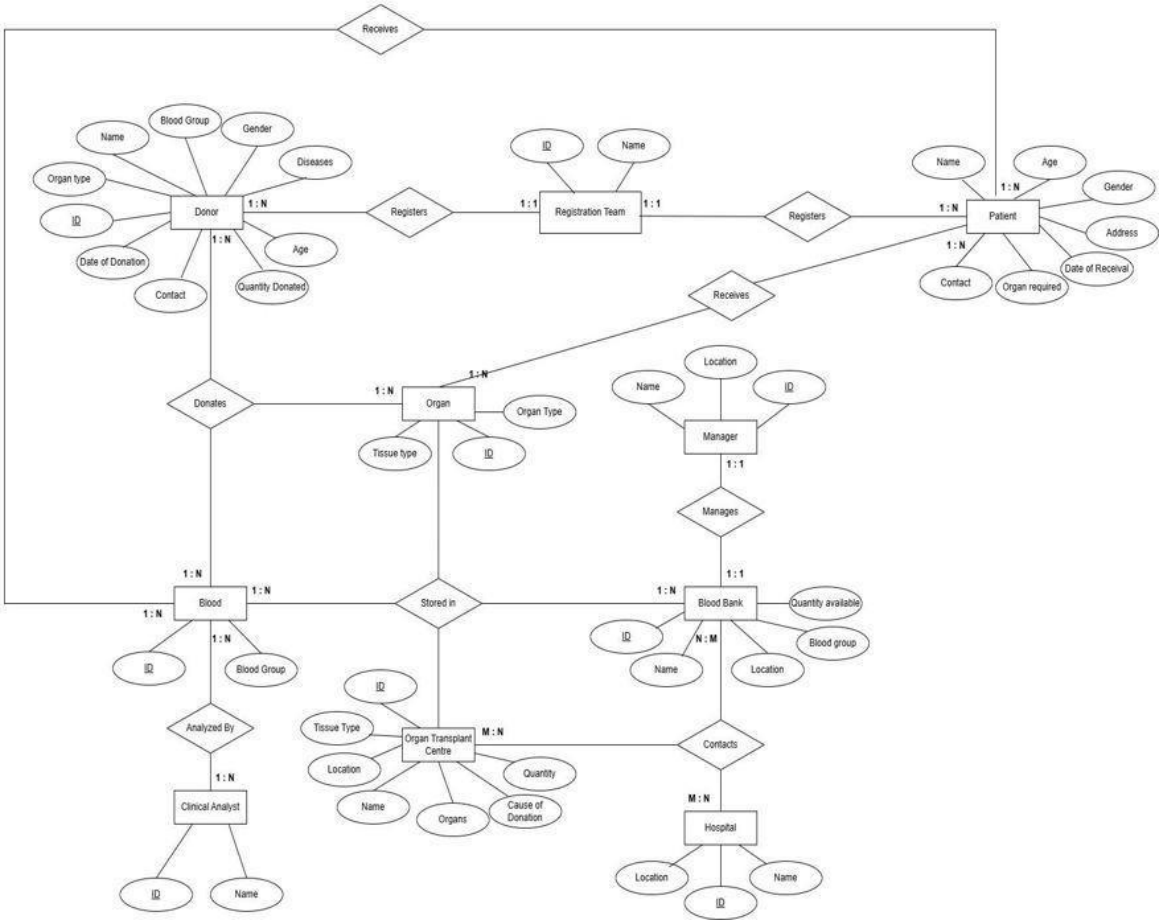
Pranathi Bombay
+1 (857) 384-9844

bombay.p@northeastern.edu

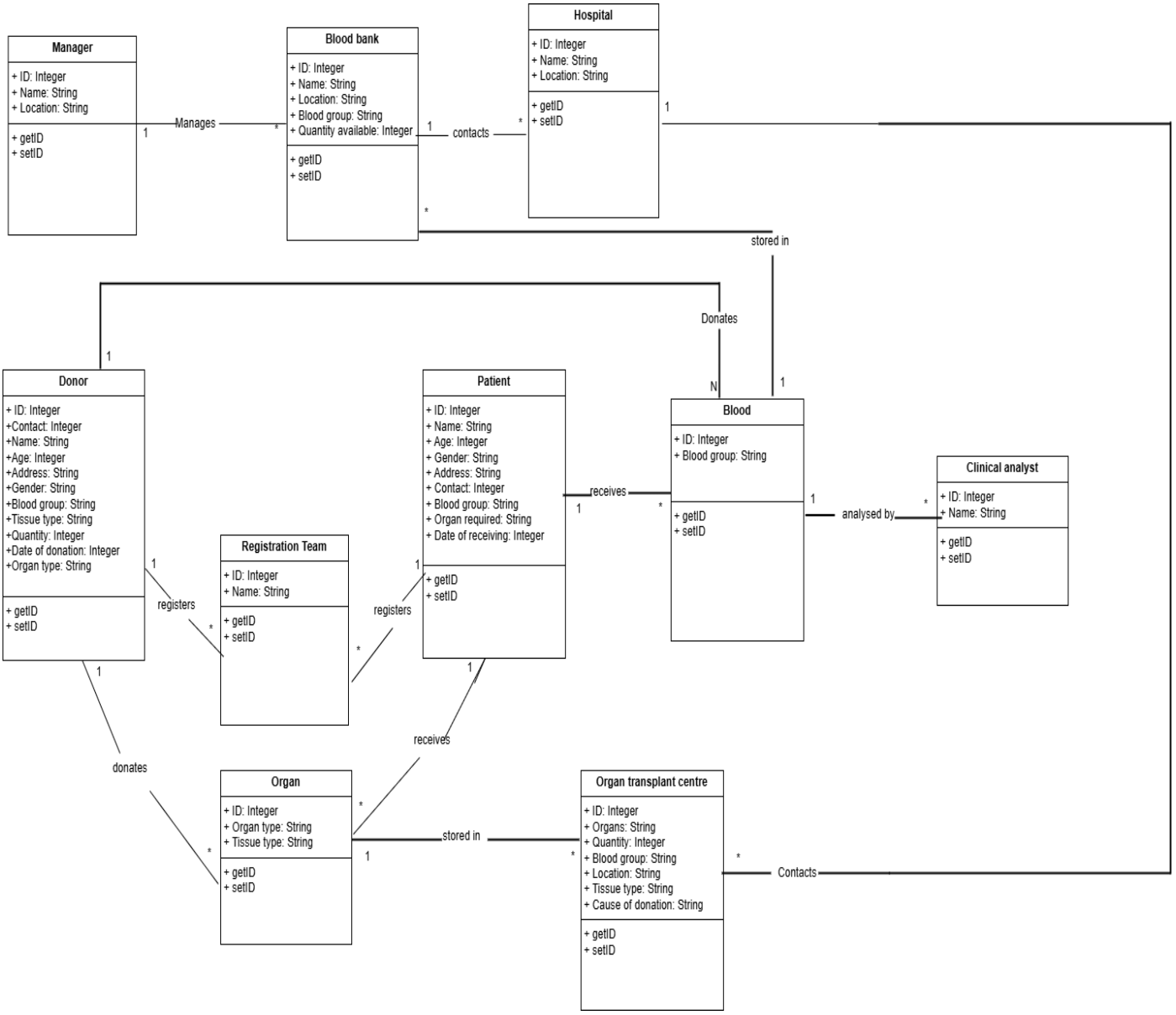
PROBLEM STATEMENT

The objective of this project is to develop a Blood and Organ Donation Network that simplifies the management of donor records and ensures the efficient allocation of blood and organs to hospitals based on their requirements. This system is designed to collect, organize, and analyze data related to blood banks and organ transplant centers, centralizing donor and recipient information to improve accessibility and operational efficiency. Key features include maintaining detailed donor records, tracking blood stock levels across locations, monitoring donation and transplant processes, and managing hospital requests for blood and organs. The system will provide real-time updates on the availability of blood types and organs, streamline the matching of organ donors with recipients based on compatibility criteria, and track ongoing and completed transplants for accountability. Additionally, it aims to raise awareness about blood and organ donation by identifying donation trends and sending reminders to donors for future eligibility. Built using a MySQL database, NOSQL And Python for the application in visual studio code. The Blood and Organ Donation Network ensures secure, scalable, and reliable data handling, improving transparency, accessibility, and efficiency in saving lives.

EER DIAGRAM


Receives
Name
Blood Group
Gender
Diseases
Organ type
ID
Donor
1 : N
Registers
1 : 1
ID
Name
Registration Team
1 : 1
Registers
1 : N
Name
Age
Gender
Address
Patient
Date of Receival
Organ required
Contact
1 : N
Date of Donation
Contact
Quantity Donated
Age
Receives
1 : N
Donates
1 : N
Organ
Organ Type
Tissue type
ID
Location
Name
ID
Manager
1 : 1
Manages
1 : N
1 : N
Blood
1 : N
1 : N
ID
Blood Group
Stored in
1 : N
Blood Bank
1 : 1
Quantity available
ID
Name
N : M
Location
Blood group
Analyzed By
1 : N
Clinical Analyst
ID
Name
ID
Tissue Type
Location
Name
Organ Transplant Centre
M : N
Quantity
Organs
Cause of Donation
Contacts
M : N
Hospital
Location
Name
ID

UML DIAGRAM


Manager
+ ID: Integer
+ Name: String
+ Location: String
+ getID
+ setID
Manages
Blood bank
+ ID: Integer
+ Name: String
+ Location: String
+ Blood group: String
+ Quantity available: Integer
+ getID
+ setID
contacts
Hospital
+ ID: Integer
+ Name: String
+ Location: String
+ getID
+ setID
stored in
Donates
Donor
+ ID: Integer
+Contact: Integer
+Name: String
+Age: Integer
+Address: String
+Gender: String
+Blood group: String
+Tissue type: String
+Quantity: Integer
+Date of donation: Integer
+Organ type: String
+ getID
+ setID
registers
Registration Team
+ ID: Integer
+ Name: String
+ getID
+ setID
registers
Patient
+ ID: Integer
+ Name: String
+ Age: Integer
+ Gender: String
+ Address: String
+ Contact: Integer
+ Blood group: String
+ Organ required: String
+ Date of receiving: Integer
+ getID
+ setID
receives
Blood
+ ID: Integer
+ Blood group: String
+ getID
+ setID
analysed by
Clinical analyst
+ ID: Integer
+ Name: String
+ getID
+ setID
donates
receives
Organ
+ ID: Integer
+ Organ type: String
+ Tissue type: String
+ getID
+ setID
stored in
Organ transplant centre
+ ID: Integer
+ Organs: String
+ Quantity: Integer
+ Blood group: String
+ Location: String
+ Tissue type: String
+ Cause of donation: String
+ getID
+ setID
Contacts

RELATIONAL MODEL

Given the above EER Diagram, the Relational Model is made:

- Primary keys are underlined and the foreign keys are referred in italics.

1. Donor:
   (Donor ID, Name, Age, Gender, Blood Group, Organ Type, Disease, Date of Donation, Contact, *REGID, BloodID*)
   - Donor ID is the primary key.
   - *REGID* is the foreign key referencing Registration Team.
   - *BloodID* is the foreign key referencing Blood.

2. Patient:
   (Patient ID, Name, Age, Gender, Blood Group, Organ Required, Date of Receival, Contact, *REGID*, *OrganID*)
   - Patient ID is the primary key.
   - *OrganID* is the foreign key referencing Organ.
   - *REGID* is a foreign key referencing Registration Team

3. Registration Team:
   (REGID, Name)
   - REGID is the primary key.

4. Organ:
   (Organ ID, Name, Tissue Type, *Donor ID, Patient ID*)
   - Organ ID is the primary key.
   - *Donor ID* is a foreign key referencing Donor.
   - *Patient ID* is a foreign key referencing Recipient.

5. Blood:
   (BloodID, Blood Group, *DonorID, PatientID, Clinical Analyst ID*, *Blood Bank ID*)
   - *Blood ID* Is the primary key.
   - *DonorID* is the foreign key referencing Donor.
   - *PatientID* is the foreign key referencing Patient.
   - *Clinical Analyst ID* is the foreign key referencing Clinical Analyst
   - *Blood Bank ID* is the foreign key referencing Blood Bank.

6. Blood Bank:
   (Blood Bank ID, Name, Location, Blood Group, Quantity Available)
   - Blood Bank ID is the primary key.

7. Manager:
   (MGRID, Name, Location, SR)
   - MGRID is the primary key.

8. Clinical Analyst:
   (Clinical Analyst ID, Name, *BloodID*)
   - Clinical Analyst ID is a primary key.
   - *BloodID* is the foreign key referencing Blood.

9. Hospital:
   (Hospital ID, Name, Location, Contact)
   - Hospital ID is the primary key.

10. Organ Transplant Centre:
    (OTCID, Name, Location)
    - OTCID is the primary key.

11. Contacts 1: Blood bank → Hospital
    (*Blood Bank ID, Blood ID, Hospital ID, Donor ID*, quantity available)

    - *Blood Bank ID* is a foreign key referencing Blood Bank
    - *Blood ID* is a foreign key referencing Blood
    - *Donor ID* is a foreign key referencing Donor.
    - *Hospital ID* is a foreign key referencing Hospital.

12. Contacts 2 : Organ Transplant Centre → Hospital
    (*OTC ID, Organ ID, Donor ID, Hospital ID*, Cause of Donation)

    - *OTC ID* is a foreign key referencing Organ Transplant Centre.
    - *Organ ID* is a foreign key referencing Organ.
    - *Donor ID* is a foreign key referencing Donor.
    - *Hospital ID* is a foreign key referencing Hospital.

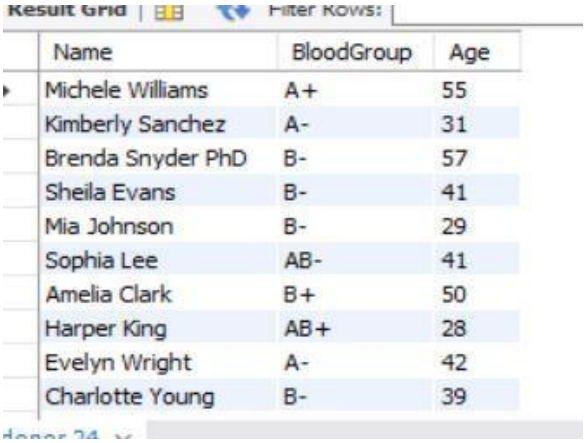
Relationships:

- A donor can donate multiple organs, but each organ belongs to only one donor.
- A recipient can receive multiple organs, but each organ is associated with a single recipient.
- Blood banks store blood of multiple donors.
- Hospitals store organs and manage transportation And Manager oversee bloodbank.

IMPLEMENTATION OF RELATIONAL MODEL VIA MYSQL AND NOSQL

1. SIMPLE QUERY

```
SELECT Name, BloodGroup, Age
    FROM donor
    WHERE Gender = 'Female';
```

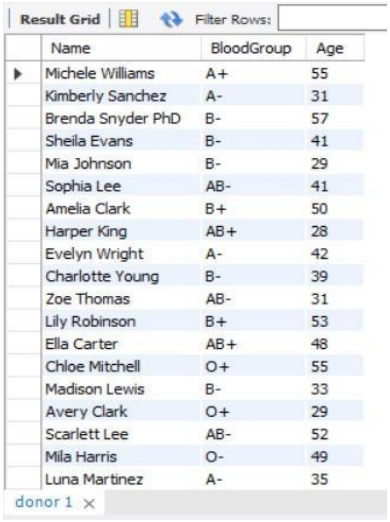


Result Grid | Filter Rows:

Name	BloodGroup	Age
Michele Williams	A+	55
Kimberly Sanchez	A-	31
Brenda Snyder PhD	B-	57
Sheila Evans	B-	41
Mia Johnson	B-	29
Sophia Lee	AB-	41
Amelia Clark	B+	50
Harper King	AB+	28
Evelyn Wright	A-	42
Charlotte Young	B-	39

2. AGGREGATE QUERY

```
SELECT H.Name AS HospitalName, SUM(OS.Quantity) AS TotalOrgansStored
FROM OrganStorage OS
JOIN Hospital H ON OS.HospitalID = H.HospitalID
GROUP BY H.HospitalID, H.Name;
```



Result Grid | Filter Rows:

Name	BloodGroup	Age
Michele Williams	A+	55
Kimberly Sanchez	A-	31
Brenda Snyder PhD	B-	57
Sheila Evans	B-	41
Mia Johnson	B-	29
Sophia Lee	AB-	41
Amelia Clark	B+	50
Harper King	AB+	28
Evelyn Wright	A-	42
Charlotte Young	B-	39
Zoe Thomas	AB-	31
Lily Robinson	B+	53
Ella Carter	AB+	48
Chloe Mitchell	O+	55
Madison Lewis	B-	33
Avery Clark	O+	29
Scarlett Lee	AB-	52
Mila Harris	O-	49
Luna Martinez	A-	35

donor 1

3. INNER JOIN

Select D.Name AS DonorName, O.OrganType AS DonatedOrgan, P.Name AS PatientName, DonationDate AS Date

FROM Donor D
JOIN organ O ON D.DonorID = O.DonorID
JOIN patient P ON O.PatientID = P.PatientID
JOIN donation DN ON D.DonorID = DN.DonorID AND O.OrganID = DN.OrganID;

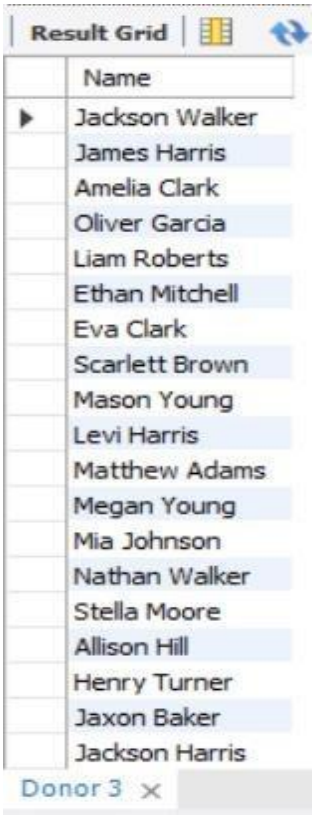
Result Grid | Filter Rows:

HospitalName	TotalOrgansStored
Columbus Health Network	2
Peoria General Hospital	1
Phoenix Valley Hospital	3
Orlando Health Institute	1
Glendale City Hospital	5
Boston Medical Group	4
Reno Health Institute	3
Tallahassee General	1
Durham Medical Center	2
San Antonio Health Clinic	5
Nashville Medical Research	6
Chicago Central Medical	3
Long Beach Health System	3
Detroit Metro Hospital	2
Cleveland Health Institute	8
Milwaukee General Hospital	5
Anaheim General Hospital	5
Overland Park Health Sy...	1
Grand Rapids City Hospital	2

Result 2

4. NESTED QUERY

```
SELECT Name
 FROM Donor
 WHERE DonorID IN (
 SELECT DonorID
 FROM Organ
 WHERE OrganType = 'Kidney' AND Status = 'Available' );
```



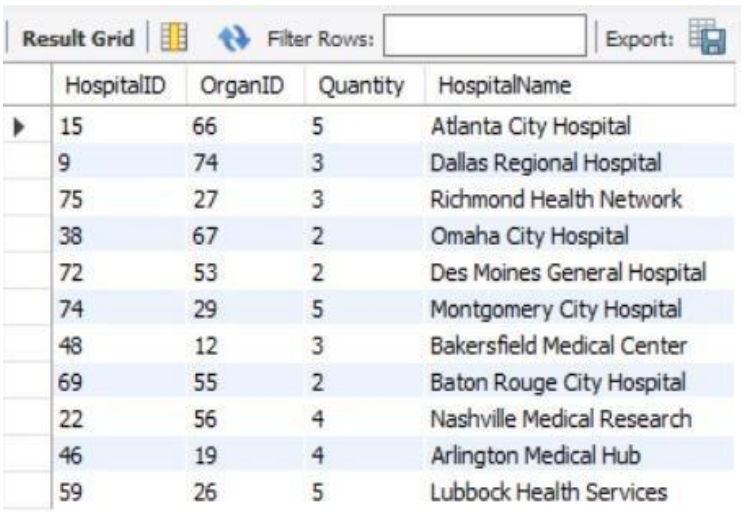
Result Grid

	Name
▶	Jackson Walker
	James Harris
	Amelia Clark
	Oliver Garcia
	Liam Roberts
	Ethan Mitchell
	Eva Clark
	Scarlett Brown
	Mason Young
	Levi Harris
	Matthew Adams
	Megan Young
	Mia Johnson
	Nathan Walker
	Stella Moore
	Allison Hill
	Henry Turner
	Jaxon Baker
	Jackson Harris

Donor 3 ×

5. CORRELATED QUERY

```
SELECT o.HospitalID, o.OrganID, o.Quantity, h.Name AS HospitalName
 FROM organstorage o
 JOIN hospital h ON o.HospitalID = h.HospitalID
WHERE o.Quantity > (
 SELECT AVG(os2.Quantity)
 FROM organstorage os2
 WHERE os2.OrganID = o.OrganID
);
```



Result Grid | Filter Rows: | Export:

HospitalID	OrganID	Quantity	HospitalName
15	66	5	Atlanta City Hospital
9	74	3	Dallas Regional Hospital
75	27	3	Richmond Health Network
38	67	2	Omaha City Hospital
72	53	2	Des Moines General Hospital
74	29	5	Montgomery City Hospital
48	12	3	Bakersfield Medical Center
69	55	2	Baton Rouge City Hospital
22	56	4	Nashville Medical Research
46	19	4	Arlington Medical Hub
59	26	5	Lubbock Health Services

6. EXIST QUERY

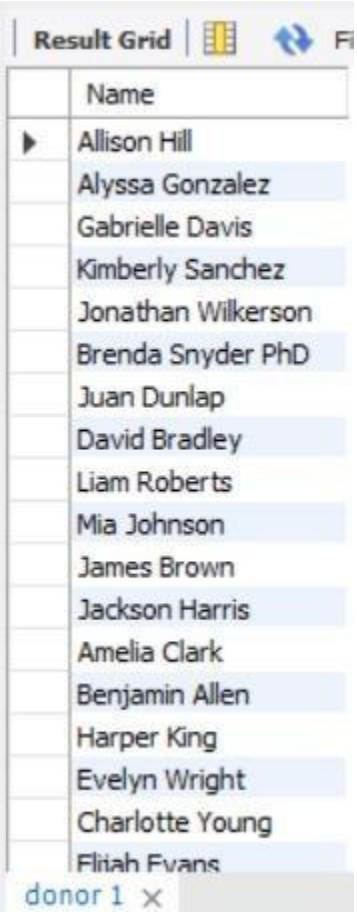
```
SELECT Name
     FROM donor d
    WHERE EXISTS (
    SELECT *
    FROM registrationteam r
    WHERE r.DonorID = d.DonorID
);
```

Result Grid | Filter Rows: | Export:

HospitalID	OrganID	Quantity	HospitalName
15	66	5	Atlanta City Hospital
9	74	3	Dallas Regional Hospital
75	27	3	Richmond Health Network
38	67	2	Omaha City Hospital
72	53	2	Des Moines General Hospital
74	29	5	Montgomery City Hospital
48	12	3	Bakersfield Medical Center
69	55	2	Baton Rouge City Hospital
22	56	4	Nashville Medical Research
46	19	4	Arlington Medical Hub
59	26	5	Lubbock Health Services

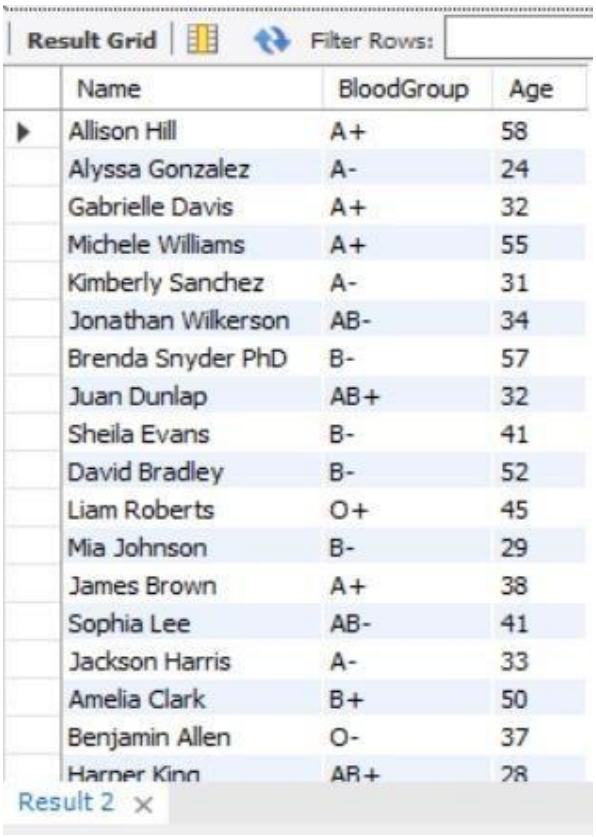
7. UNION QUERY

```
SELECT Name, BloodGroup, Age FROM donor
UNION
SELECT Name, BloodGroup, Age FROM patient;
```



8. SUBQUERIES IN SELECT AND FROM

```
SELECT BloodGroup, TotalDonors
  FROM (
  SELECT BloodGroup, COUNT(*) AS TotalDonors
  FROM donor
  GROUP BY BloodGroup
  ) AS BloodSummary#subquery in from ;
```



Result Grid | Filter Rows:

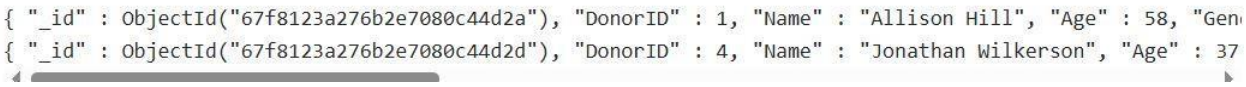
	Name	BloodGroup	Age
▶	Allison Hill	A+	58
	Alyssa Gonzalez	A-	24
	Gabrielle Davis	A+	32
	Michele Williams	A+	55
	Kimberly Sanchez	A-	31
	Jonathan Wilkerson	AB-	34
	Brenda Snyder PhD	B-	57
	Juan Dunlap	AB+	32
	Sheila Evans	B-	41
	David Bradley	B-	52
	Liam Roberts	O+	45
	Mia Johnson	B-	29
	James Brown	A+	38
	Sophia Lee	AB-	41
	Jackson Harris	A-	33
	Amelia Clark	B+	50
	Benjamin Allen	O-	37
	Harper King	AB+	28

Result 2 ×

NOSQL IMPLEMENTATION

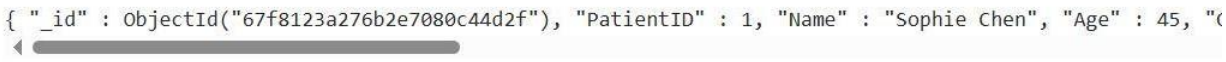
1. SIMPLE QUERY
   db.donor.find({ BloodGroup: "O+" })

Result



2. COMPLEX QUERY
   db.patient.find({ OrganRequested: "Kidney", Age: { $gt: 40 } })

Result



3. AGGREGATE QUERY
   db.donor.aggregate([
   { $group: { _id: "$BloodGroup", totalQuantity: { $sum: "$QuantityDonated" } } }
   ])

Result

```
{ "_id" : "B-", "totalQuantity" : 1 }
{ "_id" : "A+", "totalQuantity" : 4 }
{ "_id" : "O+", "totalQuantity" : 2 }
```

VISUALISATIONS IN PYTHON

1. CONNECTING THE DATABASE TO PYTHON

```python
import mysql.connector
import pandas as pd

# Connect to MySQL
conn = mysql.connector.connect(
    host="127.0.0.1",
    port=3306,
    user="root",
    password="pranathi123",
    database="organdonationdb"
)

# Create a dictionary to store DataFrames
tables_data = {}

# Fetch the list of all tables in the schema
cursor = conn.cursor()
cursor.execute("SHOW TABLES;")
tables = cursor.fetchall()

# Iterate over each table and fetch its data
for table in tables:
    table_name = table[0]
    try:
        query = f"SELECT * FROM {table_name}"
        df = pd.read_sql(query, conn)
        tables_data[table_name] = df
        print(f"Successfully fetched data from table: {table_name}")
    except Exception as e:
        print(f"Error fetching data from table {table_name}: {e}")

# Close the connection
conn.close()

# Print the data from each table
for table_name, df in tables_data.items():
    print(f"\nData from table '{table_name}':")
    print(df.head())
```

✓ 0.1s

```
Successfully fetched data from table: blood
Successfully fetched data from table: bloodbank
Successfully fetched data from table: clinicalanalyst
Successfully fetched data from table: donation
Successfully fetched data from table: donor
Successfully fetched data from table: hospital
Successfully fetched data from table: manager
Successfully fetched data from table: organ
Successfully fetched data from table: organstorage
Successfully fetched data from table: organtransplantcentre
Successfully fetched data from table: patient
Successfully fetched data from table: registrationteam

Data from table 'blood':
   BloodID BloodGroup  QuantityAvailable           StorageLocatio
0        1         A+                 10       New York Blood Ban
1        2         O-                 15  Los Angeles Blood Cente
2        3         B+                  8  Chicago Central Hospita
3        4        AB-                 12           Houston Genera
4        5         A-                  9     Phoenix Blood Storag

Data from table 'bloodbank':
   BloodBankID                 Name     Location       Contact
0            1  Red Cross Blood Bank     New York  123-555-6789
1            2   LifeSave Blood Bank  Los Angeles  987-555-1234
...
1               2       50         30       2025-01-10
2               3       30         70       2025-01-15
3               4       70         80       2025-01-20
4               5       80         40       2025-01-25
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell
C:\Users\RCP\AppData\Local\Temp\ipykernel_1576\3282918274.py:26: U
  df = pd.read_sql(query, conn)
```

2. QUERY TO RETRIVE DATA FROM TABLES

```python
import mysql.connector
import pandas as pd

# Connect to MySQL
conn = mysql.connector.connect(
    host="127.0.0.1",
    port=3306,
    user="root",
    password="pranathi123",
    database="organdonationdb"
)

# Tables to fetch data from
tables = ["blood", "organ", "patient"]

# Dictionary to store DataFrames
tables_data = {}

try:
    for table in tables:
        query = f"SELECT * FROM {table};"
        print(f"Fetching data from table: {table}")

        # Execute the query and load data into a DataFrame
        df = pd.read_sql(query, conn)
        tables_data[table] = df

        # Display the first few rows
        print(f"\nData from table '{table}':")
        print(df.head())

finally:
    # Close the connection
    conn.close()

# Saving data to CSV files (optional)
for table, df in tables_data.items():
    df.to_csv(f"{table}.csv", index=False)
    print(f"Data from '{table}' table saved to {table}.csv")
```

✓ 0.0s

```
Fetching data from table: blood

Data from table 'blood':
   BloodID BloodGroup  QuantityAvailable           StorageLocation
0        1         A+                 10       New York Blood Bank
1        2         O-                 15  Los Angeles Blood Center
2        3         B+                  8  Chicago Central Hospital
3        4        AB-                 12          Houston General
4        5         A-                  9    Phoenix Blood Storage
Fetching data from table: organ

Data from table 'organ':
   OrganID OrganType  DonorID  PatientID           StorageLocation     Status
0        1    Kidney       57         78       New York Blood Bank  Available
1        2     Heart       19         36  Los Angeles Blood Center  Available
2        3     Liver       95         43  Chicago Central Hospital  Available
3        4      Lung       41         72          Houston General  Available
4        5    Kidney       76         98    Phoenix Blood Storage  Available

  TissueType
0          A
1          B
2         AB
3          O
4          A
```

3. FULL JOIN

- Using Left Join, Right Join and the Union function to Join both the tables.

```
import mysql.connector
import pandas as pd

# Connect to MySQL
conn = mysql.connector.connect(
    host="127.0.0.1",
    port=3306,
    user="root",
    password="pranathi123",
    database="organdonationdb"
)

try:
    # full Join query between blood and organ tables based on StorageLocation
    query = """
    SELECT b.BloodID, b.BloodGroup, b.QuantityAvailable, b.StorageLocation,
           o.OrganID, o.OrganType, o.Status, o.TissueType
    FROM blood AS b
    LEFT JOIN organ AS o ON b.StorageLocation = o.StorageLocation
    UNION
    SELECT b.BloodID, b.BloodGroup, b.QuantityAvailable, b.StorageLocation,
           o.OrganID, o.OrganType, o.Status, o.TissueType
    FROM blood AS b
    RIGHT JOIN organ AS o ON b.StorageLocation = o.StorageLocation;
    """

    # Execute the query and load data into a DataFrame
    df = pd.read_sql(query, conn)

    # Display the result
    print("Outer Join of blood and organ tables:")
    print(df.head())

finally:
    # Close the connection
    conn.close()

# Save the result to a CSV file (optional)
df.to_csv("blood_organ_outer_join.csv", index=False)
print("Joined data saved to 'blood_organ_outer_join.csv'")
```

```
Outer Join of blood and organ tables:
   BloodID BloodGroup  QuantityAvailable      StorageLocation  OrganID  \
0        1         A+                 10  New York Blood Bank     91.0
1        1         A+                 10  New York Blood Bank     76.0
2        1         A+                 10  New York Blood Bank     61.0
3        1         A+                 10  New York Blood Bank     46.0
4        1         A+                 10  New York Blood Bank     31.0

  OrganType     Status TissueType
0      Lung  Available          O
1    Kidney  Available          O
2      Lung  Available          A
3    Kidney  Available          O
4      Lung  Available          A
Joined data saved to 'blood_organ_outer_join.csv'
```

4. NESTED QUERY

```
# Connect to MySQL
conn = mysql.connector.connect(
    host="127.0.0.1",
    port=3306,
    user="root",
    password="pranathi123",
    database="organdonationdb"
)

try:
    # Nested query to find blood groups with quantity greater than the average
    query = """
    SELECT BloodID, BloodGroup, QuantityAvailable, StorageLocation
    FROM blood
    WHERE QuantityAvailable > (
        SELECT AVG(QuantityAvailable)
        FROM blood
    );
    """

    # Execute the query and load data into a DataFrame
    df = pd.read_sql(query, conn)

    # Display the result
    print("Blood groups with quantity greater than the average:")
    print(df)

finally:
    # Close the connection
    conn.close()

# Save the result to a CSV file (optional)
df.to_csv("blood_above_average.csv", index=False)
print("Results saved to 'blood_above_average.csv'")
```

✓ 0.0s

```
Blood groups with quantity greater than the average:
    BloodID BloodGroup  QuantityAvailable                StorageLocation
0         2         O-                 15         Los Angeles Blood Center
1         4        AB-                 12                  Houston General
2         6         O+                 20       San Francisco Donor Center
3         8        AB+                 14          Seattle Health Services
4         9         A+                 13                 Dallas Blood Hub
5        11         B+                 16         Miami Transfusion Center
6        13         A-                 18        Boston Medical Blood Bank
7        15         B-                 20          Atlanta General Storage
8        17         A+                 17                  Detroit Medical
9        18         O-                 14          Portland Blood Services
10       20        AB-                 19         Indianapolis Blood Vault
11       22         O+                 12         Nashville Donor Services
12       26         O-                 13            Sacramento Blood Bank
13       27         B+                 15            Salt Lake City Health
14       28        AB-                 14          Cleveland Blood Reserve
15       29         A-                 16                    Tampa Medical
16       30         O+                 18       New Orleans Blood Services
17       33         A+                 12          Columbus Health Storage
18       35         B+                 13               Raleigh Donor Bank
19       37         A-                 14              Albuquerque Medical
20       40        AB+                 17  Colorado Springs Blood Storage
21       41         A+                 19           Tulsa General Hospital
22       43         B+                 16                  Mesa Blood Bank
...
47       95         B-                 17        Tallahassee Blood Storage
48       99         B+                 13          Brownsville Blood Vault
49      100        AB-                 15        Newport News Blood Center
Results saved to 'blood_above_average.csv'
```

VISUALIZATIONS IN THE FORM OF GRAPHS

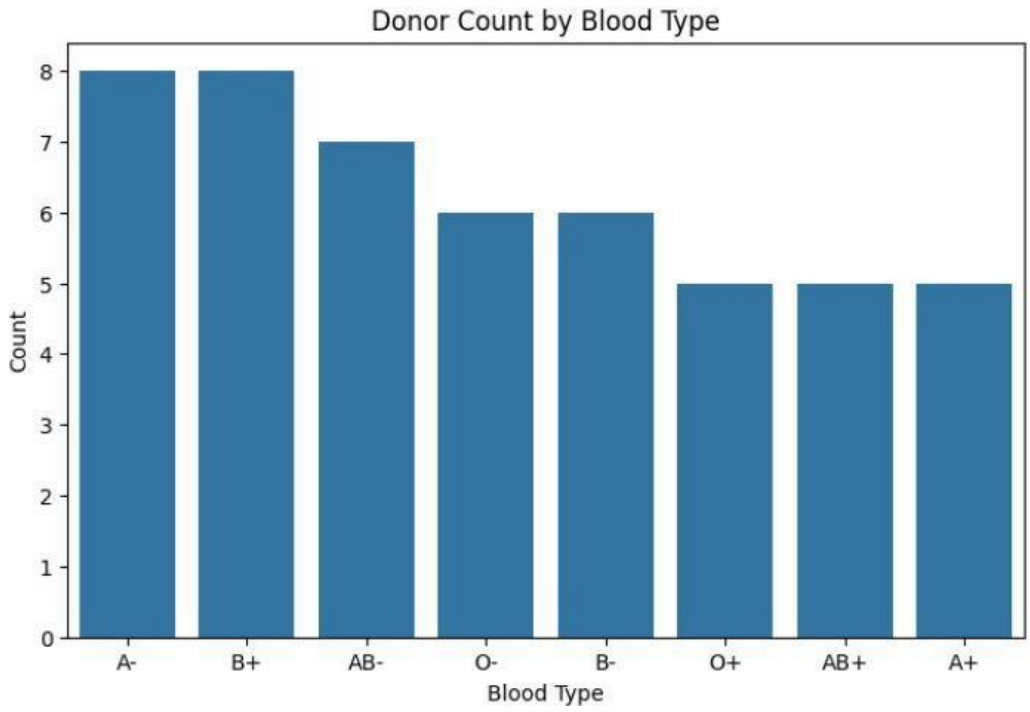
- BAR GRAPH

  Represents the Donor count by the Blood Type i.e., count the total number of Donor for each Blood Group.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Example: Count of donors by blood type
plt.figure(figsize=(8, 5))
sns.countplot(data=df, x='BloodGroup', order=df['BloodGroup'].value_counts().index)
plt.title('Donor Count by Blood Type')
plt.xlabel('Blood Type')
plt.ylabel('Count')
plt.show()
```

✓ 0.2s



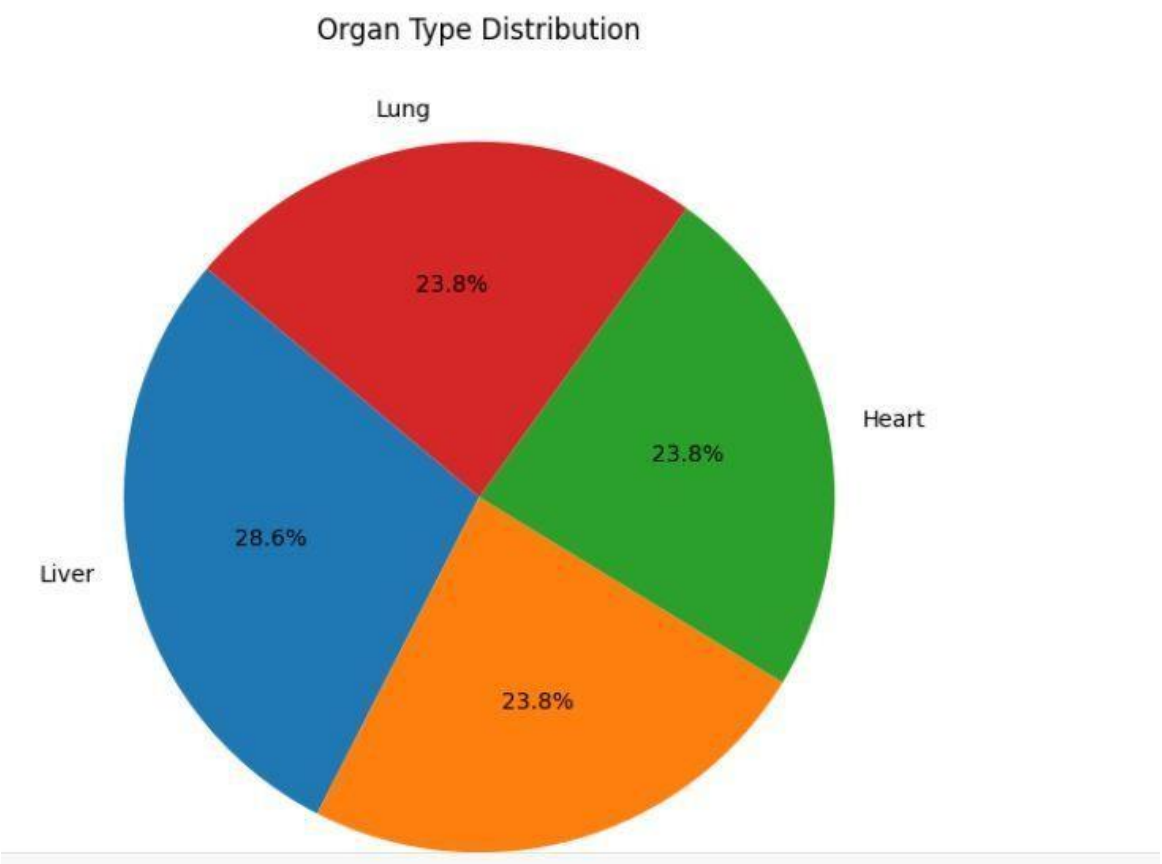
- PIE CHART

  Represents the Organ Type Distribution i.e., shows the total count for each Organ present in the Storage Centre.

```
import matplotlib.pyplot as plt

# Count the number of each organ type
organ_counts = organ_df['OrganType'].value_counts()

# Plotting the pie chart
plt.figure(figsize=(7, 7))
plt.pie(organ_counts, labels=organ_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Organ Type Distribution')
plt.show()
```



- HISTOGRAM

  Represents the Donor count by Disease Type i.e., shows the total count of the Donors who may or may not have a disease present.

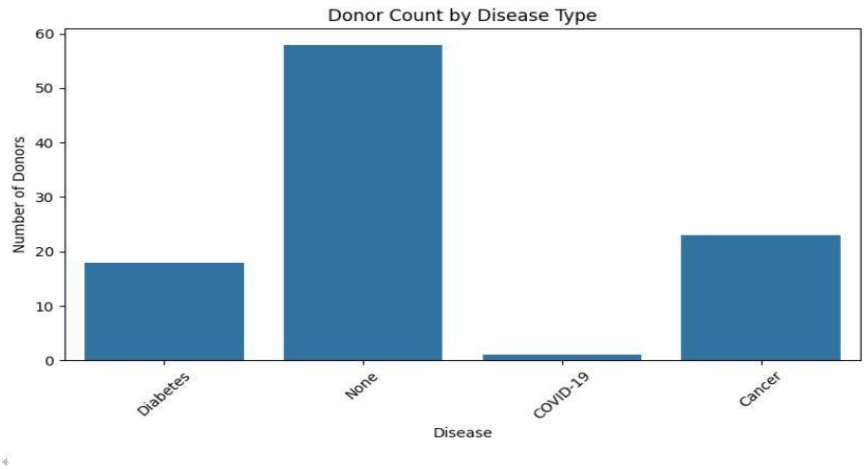
```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Connect to MySQL
conn = mysql.connector.connect(
    host="127.0.0.1",
    port=3306,
    user="root",
    password="pranathi123",
    database="organdonationdb"
)

# Load the data
query = "SELECT Disease, COUNT(*) AS DonorCount FROM donor GROUP BY Disease;"
disease_df = pd.read_sql(query, conn)

# Close connection
conn.close()

# Plot the bar chart
plt.figure(figsize=(8,5))
sns.barplot(data=disease_df, x='Disease', y='DonorCount')
plt.title('Donor Count by Disease Type')
plt.xlabel('Disease')
plt.ylabel('Number of Donors')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



SUMMARY

The Blood and Organ Donation Management System developed through this project serves as a robust solution for enhancing the coordination and efficiency of life-saving donations. By centralizing data related to donors, recipients, blood banks, and transplant centers, the system ensures seamless access, retrieval, and analysis of critical information.

Built on a secure and scalable MySQL database, the platform effectively manages donor records, tracks blood inventory across locations, monitors hospital requests, and facilitates the matching of organs with compatible recipients. Real-time availability updates, detailed monitoring of donation and transplant processes, and compatibility-based donor-recipient matching significantly improve transparency and responsiveness.

Furthermore, the system not only optimizes operational workflows but also supports social impact by identifying donation trends and promoting donor engagement through reminders and awareness initiatives. The integration of both SQL and NoSQL technologies, combined with data visualization and Python automation, makes the solution comprehensive and ready for real-world deployment in healthcare networks.

In conclusion, this project delivers a data-driven and purpose-focused system that strengthens the blood and organ donation ecosystem, contributing meaningfully to saving lives and supporting healthcare infrastructure.