Blood Donation Database System

Milestone: Project Report

Group 16
Hrithik Puri
Aryan Fernandes

617-935-8046(Hrithik Puri)
617-708-5129(Aryan Fernandes)

puri.hr@northeastern.edu
fernandes.ar@northeastern.edu

Percentage of Effort Contributed by Hrithik Puri: 50%
Percentage of Effort Contributed by Aryan Fernandes: 50%

Signature of Student 1: Hrithik Puri
Signature of Student 2: Aryan Fernandes

Submission Date: 12/9/22

USE CASE STUDY REPORT

Group No: Group 16

Topic: Blood Bank Database System

Student Names: Hrithik Puri and Aryan Fernandes

Executive Summary:

The main goal of this study was to develop and implement a relational database system that would allow users to obtain blood in the most practical way possible, taking into account variables like location, blood type, and period of storage. We'll be building a strong, centralized system that has all the information on the blood bank's stockpiles, donor information, blood bank information, and many more in order to provide the receiver with useful information.

Following the modeling of the EER and UML diagrams, the conceptual model was mapped to a relational model with the necessary main and foreign keys. This database was then implemented fully on MySQL and implemented on MongoDB NoSQL database to study the feasibility of this database in a NoSQL environment.

The built database is a huge success, and by integrating Python, it has tremendous analytical possibilities, some of which have been demonstrated in the study. The existing situation of blood banks around the world can be much improved thanks to these queries.

Introduction:

A blood donation happens when a person willingly consents to having blood drawn for transfusions or, in some cases, when their blood is fractionated and utilized to create pharmaceutical drugs. Blood in its entirety or a specific component may be directly donated. Blood banks frequently take part in both the operations that come before and after the collection process.

In hospitals or blood donation camps, one can give blood. Voluntary, unpaid blood donors from low-risk communities are the safest blood donors. According to 1975's World Health Assembly resolution 28.72, the World Health Organization wants all nations to receive all of their blood supplies from willing, unpaid donors.

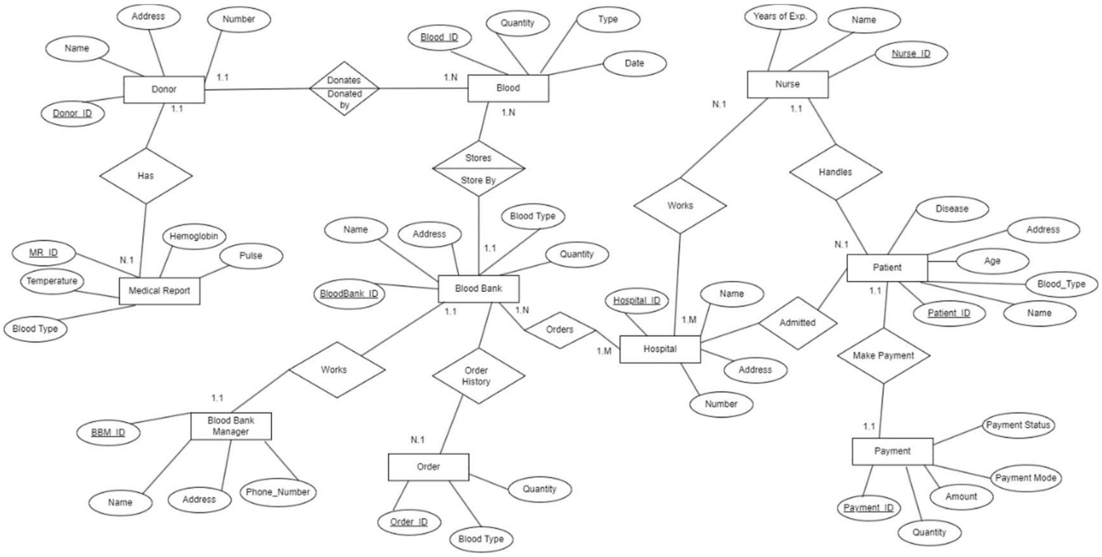
It has been observed that the number of blood donors is rising annually, in the US an estimated 6.8 million people donate blood each year. Surgery, severe injuries, childbirth, cancer therapy, blood abnormalities, chronic illnesses, anemia, and many more conditions all require blood. According to an American Red Cross report, the US needs about 29,000 units of red blood cells every day.

When it comes to the use of blood, there are a few crucial points such as, platelets must be utilized in just 5 days, while red blood cells must be used in 6 weeks (or fewer). However, a study from Johns Hopkins University found that red blood cells lose some of their capacity to transport oxygen-rich cells throughout the body after three weeks. Over three weeks, blood loses flexibility and loses its ability to fit in the body's smallest capillaries.

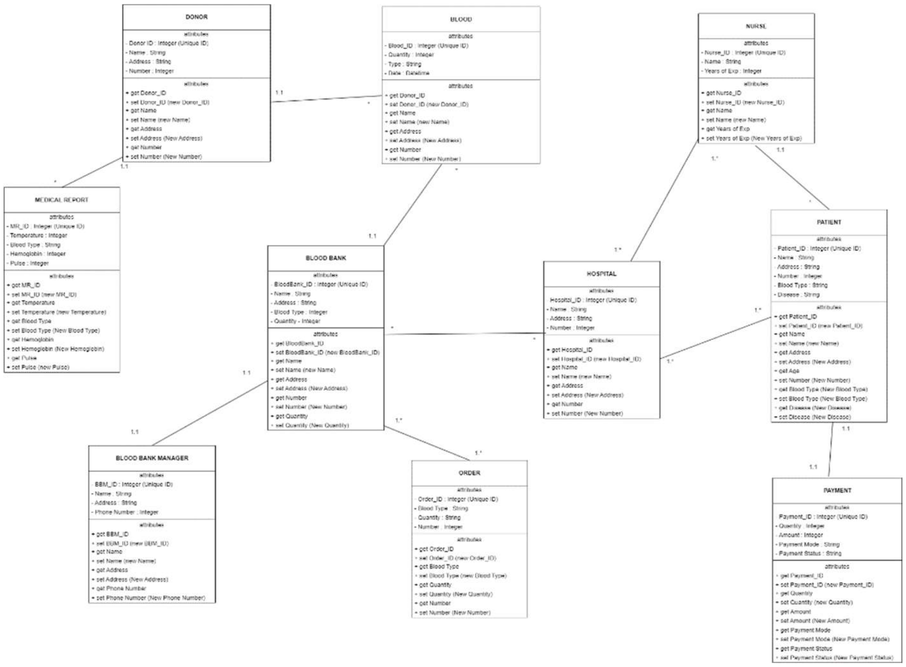
Our final aim is to maintain a database system where the user can get blood in the most efficient manner considering the factors such as location, blood type, and duration of blood stored. To give the receiver valuable information, we'll be constructing a robust, centralized system that contains all the data on the blood bank's stockpiles, donor information, blood bank information, and many more.

Conceptual Data Modeling:

1. EER Model



2. UML Diagram



Mapping Conceptual Model to Relational Model:

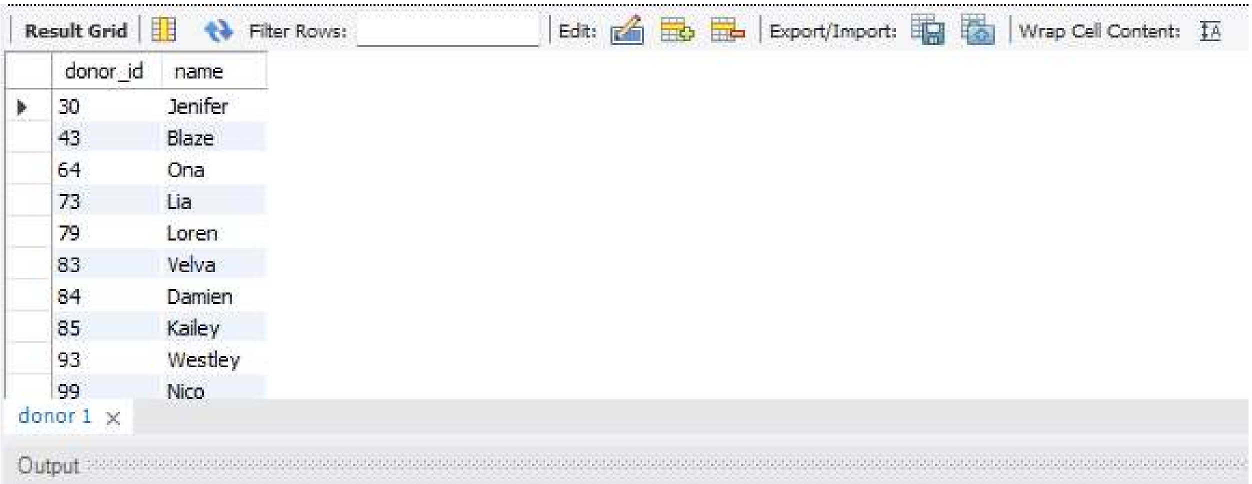
1. Donor (Donor_ID, Name, Address, Number)
2. Medical Report (MR_ID, Temperature, Hemoglobin, Pulse, *Donor_ID*)
3. Blood (Blood_ID, Type, Quantity, Date, *Donor_ID*)
4. Patient (Patient_ID, Name, Age, Disease, Address, Number, Blood Type, *Nurse_ID*, *Hospital_ID*)
5. Payment (Payment_ID, Payment_Status, Amount, Quantity, Payment mode, *Patient_ID*)
6. Nurse (Nurse_ID, Name, Age, Year_Experience, *Hospital_ID*)
7. Hospital (Hospital_ID, Name, Address, Number)
8. Blood Bank (BloodBank_ID, Name, Address, Blood_Type, Quantity, *BloodBankManager_ID*)
9. Blood Bank Manager (BloodBankManager_ID, Name, Address, PhoneNumber)
10. Order History (Order_ID, Blood_Type, Quantity, *BloodBank_ID*)
11. Orders (*Order_ID*, *Hospital_ID*)

Implementation of Relation Model via MySQL and NoSQL:

1. MySQL Implementation:

Query1: Selecting Names of Donors who donated more than 1 pint of blood (using exists)

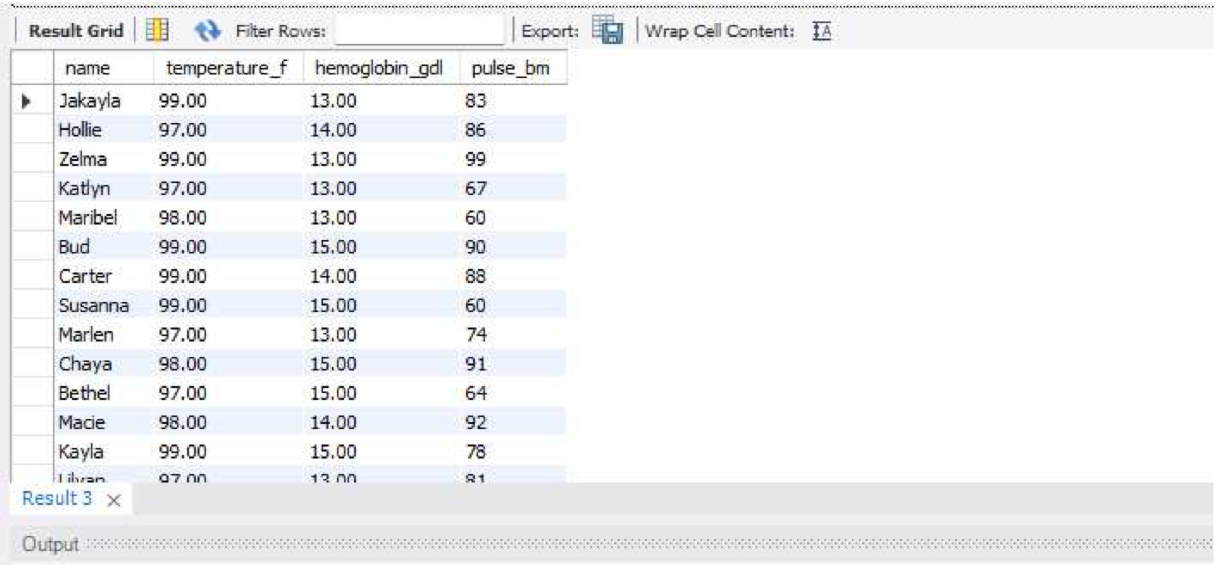
select donor_id,name from donor where exists (select donor_id from blood where donor.donor_id=blood.donor_id and quantity>500);



donor_id	name
30	Jenifer
43	Blaze
64	Ona
73	Lia
79	Loren
83	Velva
84	Damien
85	Kailey
93	Westley
99	Nico

Query2: Checking medical reports of all existing donors (inner join)

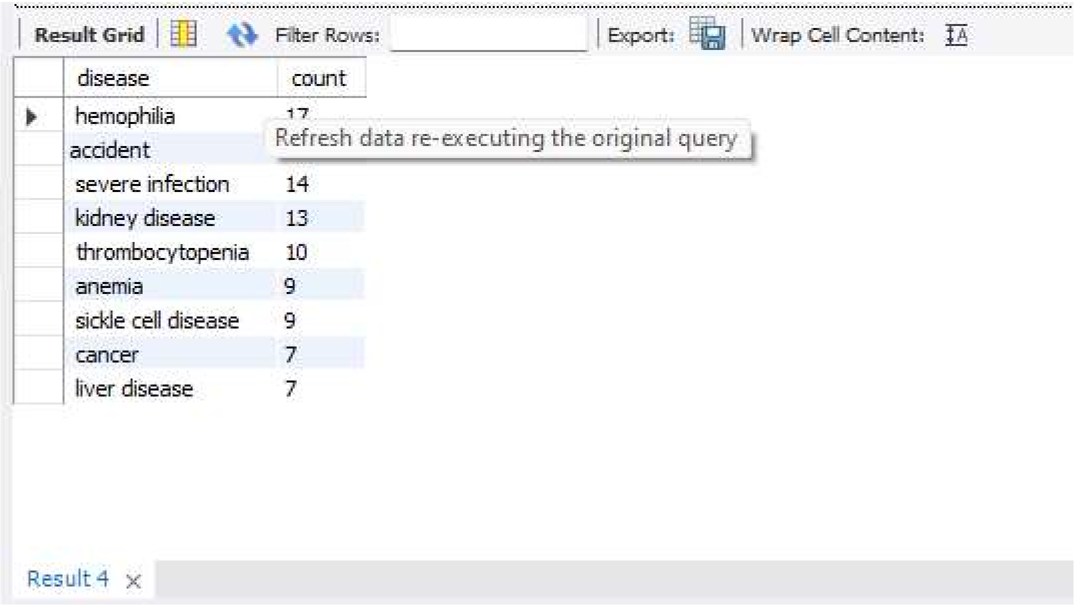
select name, temperature_f, hemoglobin_gdl, pulse_bm from medical_report inner join donor on medical_report.donor_id=donor.donor_id;



name	temperature_f	hemoglobin_gdl	pulse_bm
Jakayla	99.00	13.00	83
Hollie	97.00	14.00	86
Zelma	99.00	13.00	99
Katlyn	97.00	13.00	67
Maribel	98.00	13.00	60
Bud	99.00	15.00	90
Carter	99.00	14.00	88
Susanna	99.00	15.00	60
Marlen	97.00	13.00	74
Chaya	98.00	15.00	91
Bethel	97.00	15.00	64
Macie	98.00	14.00	92
Kayla	99.00	15.00	78
Lilyan	97.00	13.00	81

Query3: Most common disease that required blood (order by)

select disease, count(*) as count from patient group by disease order by count desc;

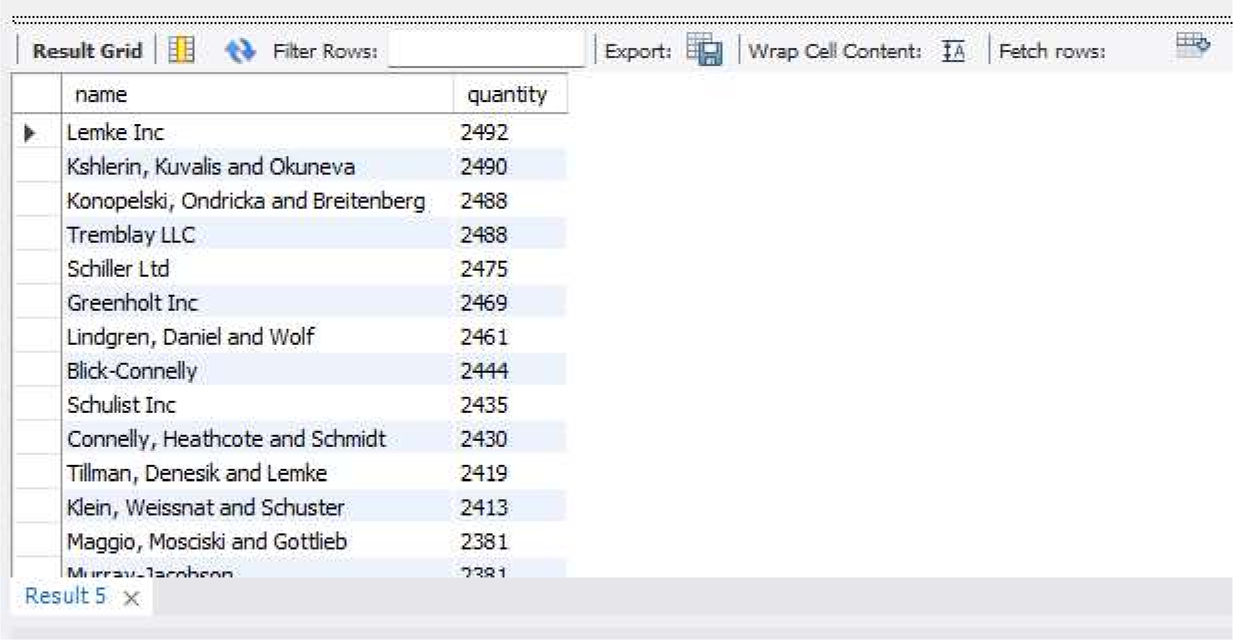


disease	count
hemophilia	17
accident	
severe infection	14
kidney disease	13
thrombocytopenia	10
anemia	9
sickle cell disease	9
cancer	7
liver disease	7

Query4: Analysis of top 20 hospitals that ordered blood in high quantities (correlated)

select * from
(select h.name, q1.quantity
from hospital as h,
(SELECT o.hospital_id,od.quantity from orders as o, order_history as od

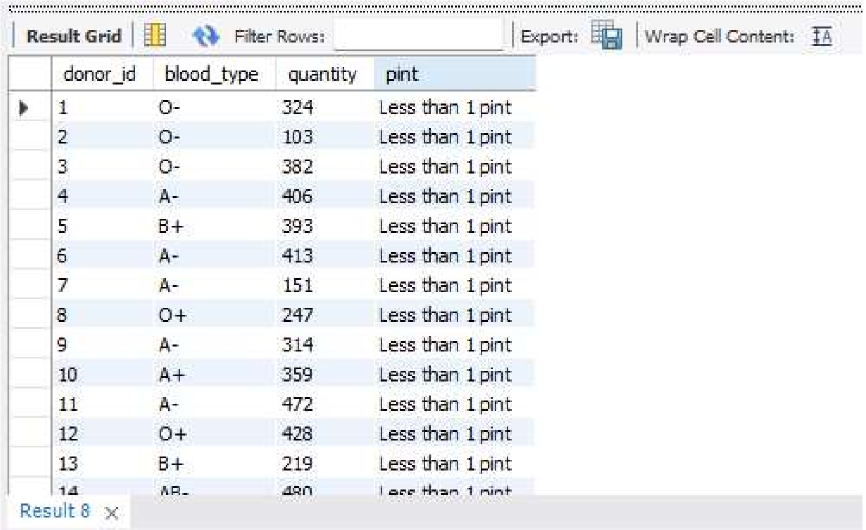
```
where od.order_id=o.order_id) as q1
where h.hospital_id=q1.hospital_id) as q2
order by q2.quantity desc limit 20;
```



name	quantity
Lemke Inc	2492
Kshlerin, Kuvalis and Okuneva	2490
Konopelski, Ondricka and Breitenberg	2488
Tremblay LLC	2488
Schiller Ltd	2475
Greenholt Inc	2469
Lindgren, Daniel and Wolf	2461
Blick-Connelly	2444
Schulist Inc	2435
Connelly, Heathcote and Schmidt	2430
Tillman, Denesik and Lemke	2419
Klein, Weissnat and Schuster	2413
Maggio, Mosciski and Gottlieb	2381
Murray-Jacobson	2381

Query5: Converting donated blood into pint (using case statement)

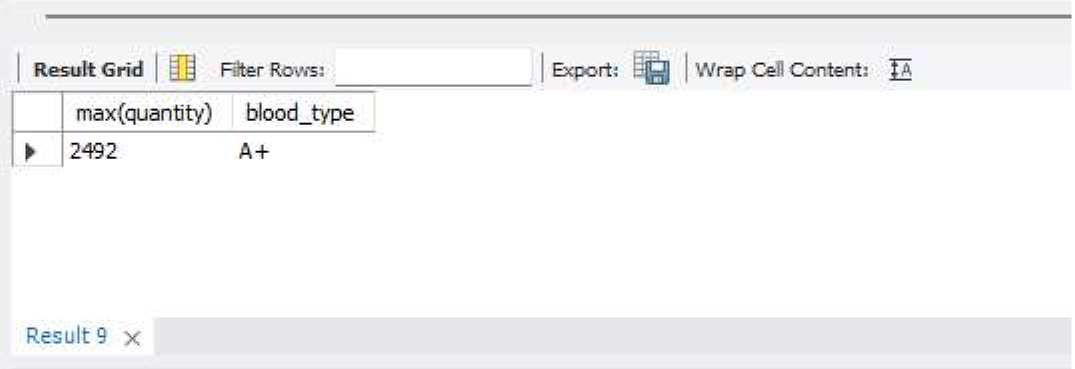
```
select donor_id, blood_type, quantity,
CASE
          WHEN quantity>500 THEN 'More than 1 pint'
   ELSE 'Less than 1 pint'
END AS pint
from blood;
```



donor_id	blood_type	quantity	pint
1	O-	324	Less than 1 pint
2	O-	103	Less than 1 pint
3	O-	382	Less than 1 pint
4	A-	406	Less than 1 pint
5	B+	393	Less than 1 pint
6	A-	413	Less than 1 pint
7	A-	151	Less than 1 pint
8	O+	247	Less than 1 pint
9	A-	314	Less than 1 pint
10	A+	359	Less than 1 pint
11	A-	472	Less than 1 pint
12	O+	428	Less than 1 pint
13	B+	219	Less than 1 pint
14	AB-	480	Less than 1 pint

Query6: Procedure to find highest quantity order anytime (storage procedure)

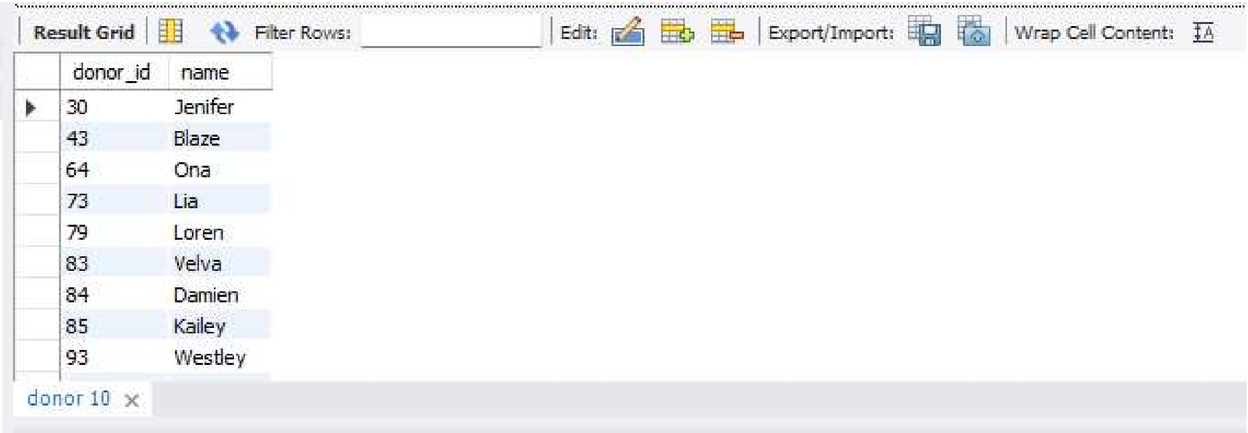
```
DELIMITER //

CREATE PROCEDURE GetMaxOrderDetails()
BEGIN
        SELECT max(quantity),blood_type  FROM order_history;
END //

CALL GetMaxOrderDetails()
```



max(quantity)	blood_type
2492	A+

Query7 - Finding all the Names of donor who has donated more than 1 pint (using ANY/ALL)

```
select donor_id, name
from donor
where donor_id = ANY
        (select donor_id
        from blood
  where quantity>500);
```

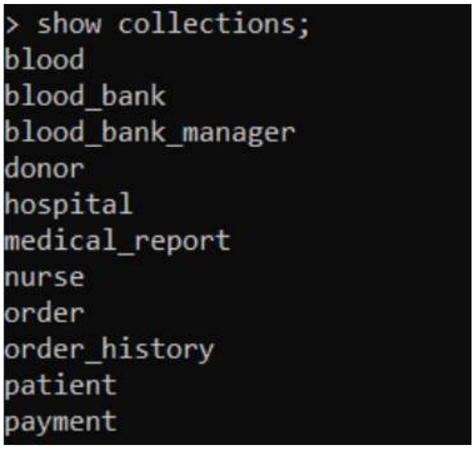


donor_id	name
30	Jenifer
43	Blaze
64	Ona
73	Lia
79	Loren
83	Velva
84	Damien
85	Kailey
93	Westley

2. NoSQL Implementation:

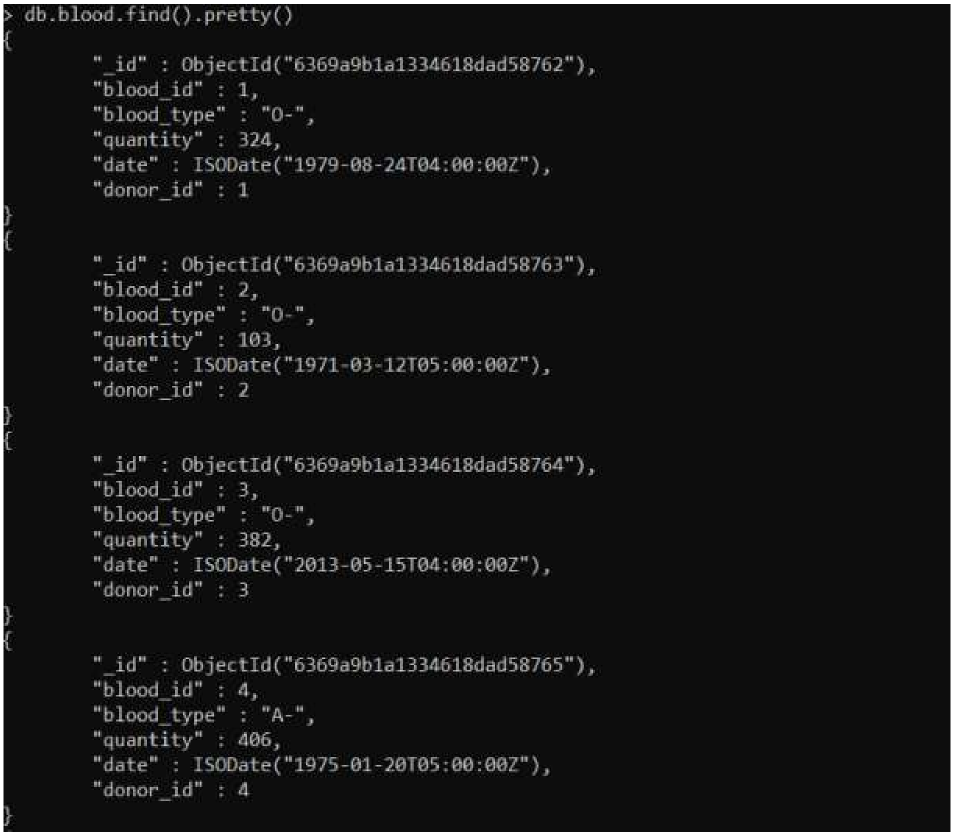
We have created a NoSQL database in mongodb and have successfully migrated our data from SQL tables to NoSQL collections in mongodb.

1) **Collections present in our mongodb blood_bank database.**



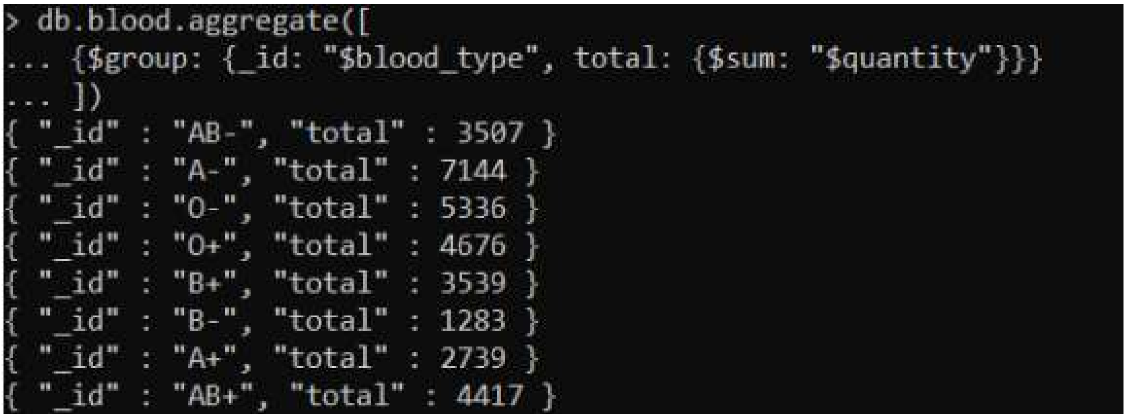
```
> show collections;
blood
blood_bank
blood_bank_manager
donor
hospital
medical_report
nurse
order
order_history
patient
payment
```

2) **Select query on blood collection.**



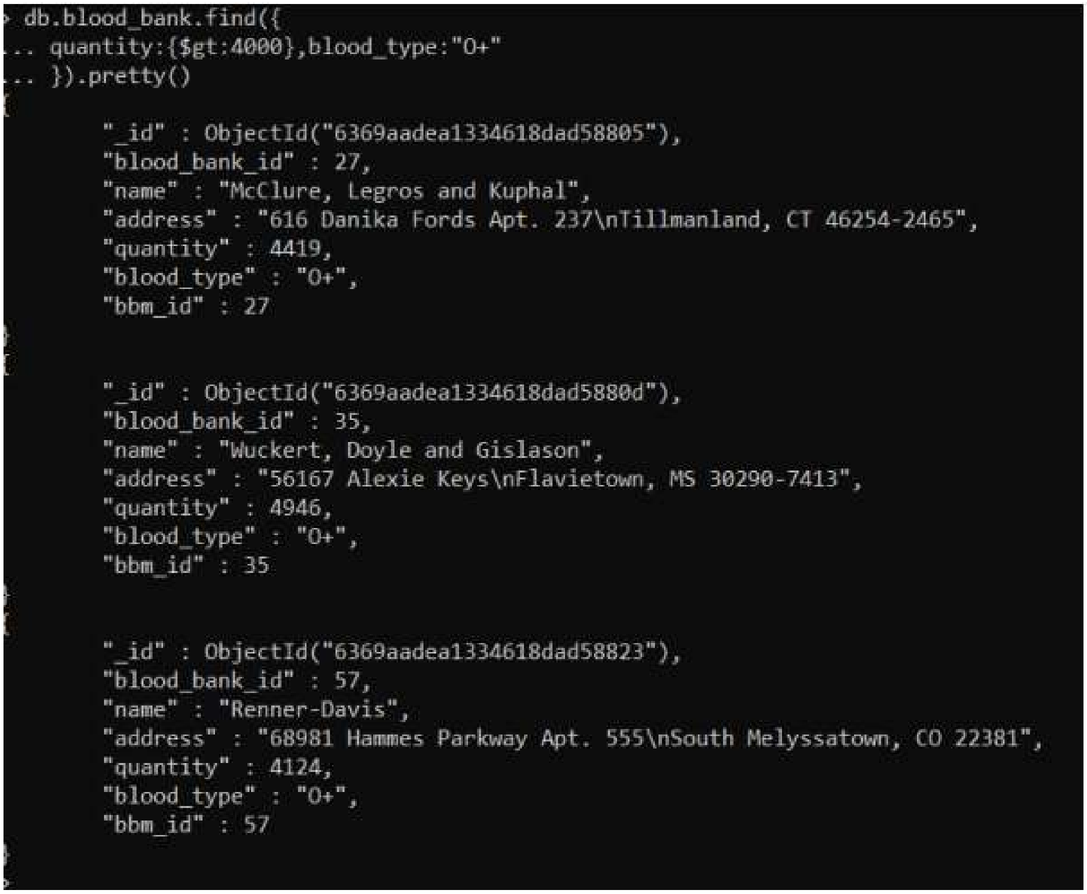
```
> db.blood.find().pretty()
{
        "_id" : ObjectId("6369a9b1a1334618dad58762"),
        "blood_id" : 1,
        "blood_type" : "O-",
        "quantity" : 324,
        "date" : ISODate("1979-08-24T04:00:00Z"),
        "donor_id" : 1
}
{
        "_id" : ObjectId("6369a9b1a1334618dad58763"),
        "blood_id" : 2,
        "blood_type" : "O-",
        "quantity" : 103,
        "date" : ISODate("1971-03-12T05:00:00Z"),
        "donor_id" : 2
}
{
        "_id" : ObjectId("6369a9b1a1334618dad58764"),
        "blood_id" : 3,
        "blood_type" : "O-",
        "quantity" : 382,
        "date" : ISODate("2013-05-15T04:00:00Z"),
        "donor_id" : 3
}
{
        "_id" : ObjectId("6369a9b1a1334618dad58765"),
        "blood_id" : 4,
        "blood_type" : "A-",
        "quantity" : 406,
        "date" : ISODate("1975-01-20T05:00:00Z"),
        "donor_id" : 4
}
```

3) **Finding total quantity of each blood type available.**



```
> db.blood.aggregate([
... {$group: {_id: "$blood_type", total: {$sum: "$quantity"}}}
... ])
{ "_id" : "AB-", "total" : 3507 }
{ "_id" : "A-", "total" : 7144 }
{ "_id" : "O-", "total" : 5336 }
{ "_id" : "O+", "total" : 4676 }
{ "_id" : "B+", "total" : 3539 }
{ "_id" : "B-", "total" : 1283 }
{ "_id" : "A+", "total" : 2739 }
{ "_id" : "AB+", "total" : 4417 }
```

4) **Finding Blood Banks which has more than 4 liters of O+ blood**

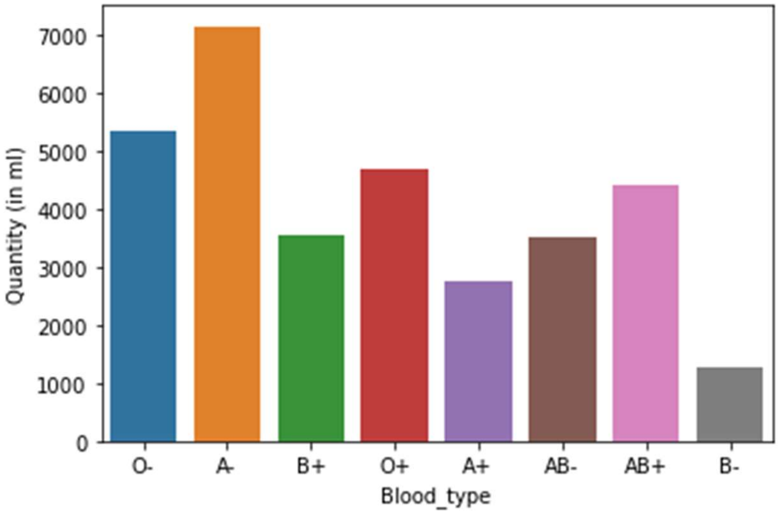


```
> db.blood_bank.find({
.. quantity:{$gt:4000},blood_type:"O+"
.. }).pretty()
{
        "_id" : ObjectId("6369aadea1334618dad58805"),
        "blood_bank_id" : 27,
        "name" : "McClure, Legros and Kuphal",
        "address" : "616 Danika Fords Apt. 237\nTillmanland, CT 46254-2465",
        "quantity" : 4419,
        "blood_type" : "O+",
        "bbm_id" : 27
}
{
        "_id" : ObjectId("6369aadea1334618dad5880d"),
        "blood_bank_id" : 35,
        "name" : "Wuckert, Doyle and Gislason",
        "address" : "56167 Alexie Keys\nFlavietown, MS 30290-7413",
        "quantity" : 4946,
        "blood_type" : "O+",
        "bbm_id" : 35
}
{
        "_id" : ObjectId("6369aadea1334618dad58823"),
        "blood_bank_id" : 57,
        "name" : "Renner-Davis",
        "address" : "68981 Hammes Parkway Apt. 555\nSouth Melyssatown, CO 22381",
        "quantity" : 4124,
        "blood_type" : "O+",
        "bbm_id" : 57
}
>
```

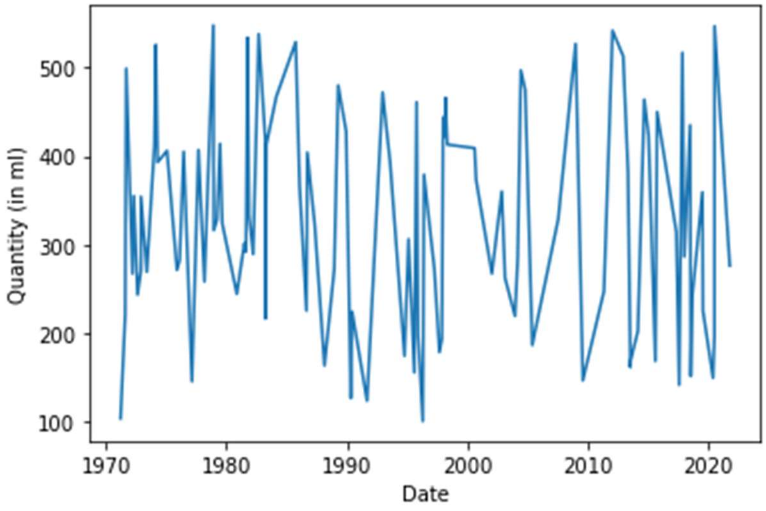
Database Access via Python:

Python is used to access the database. It uses the "mysql.connector" package, which has all functions for connections and SQL queries. To connect to the database “mysql.connector.connect” and to send a query to it a cursor is created, use “con.cursor()”, ”mycur.execute(query)”. The “mycur.fetchall()” function is used to retrieve the result set. The result set list is turned into a dataframe using the pandas package “pd.DataFrame()” function and using the seaborn package “sns.barplot”, “sns.lineplot”, etc.. for various graphs and additional analysis.

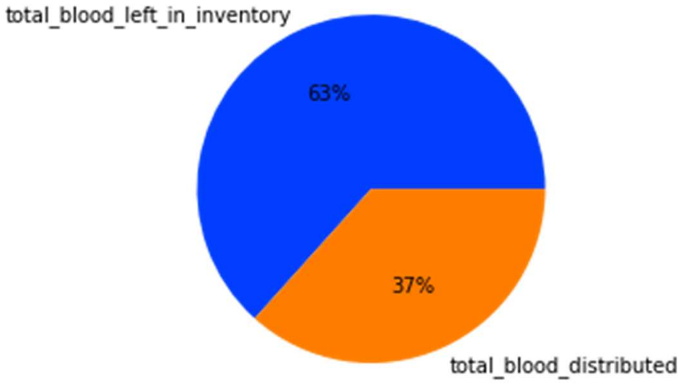
Available blood types and their quantities(ml)



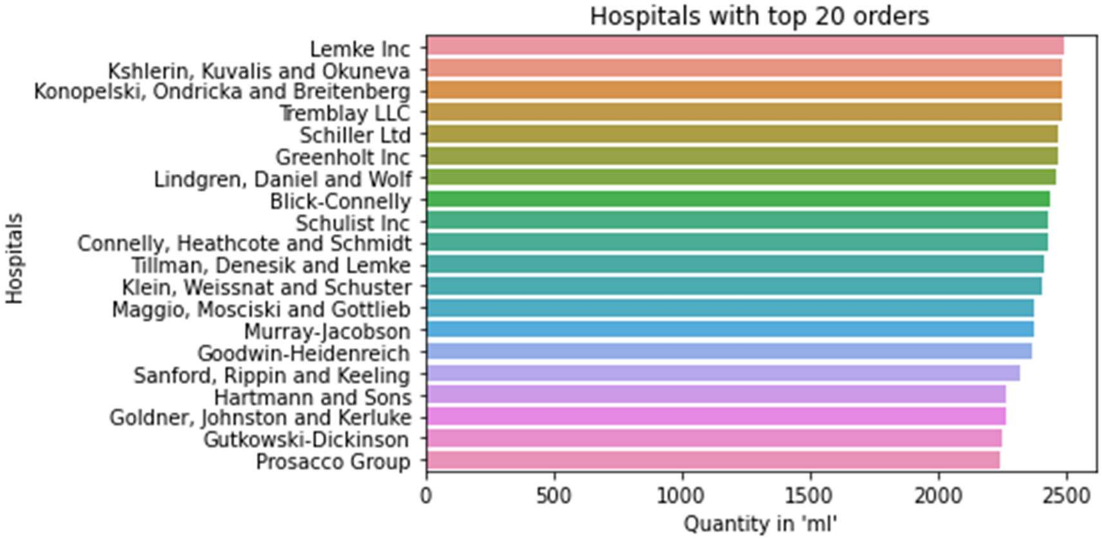
Total quantity of blood in the inventory year wise analysis



Total blood collected and distributed till date



Analysis of top 15 hospitals that ordered blood in high quantities



Summary and recommendation:

In conclusion, we have achieved our major goal, which was to maintain a dependable database that is accessible to both donors and receivers. Our project can address real world issues like locating the closest blood bank, getting access to the donors' most recent medical records, and many others. It also provides a proper end-to-end flow of blood donated and how it’s utilized.

We can improve this project, by adding additional information about the blood donation camps that take place and saving information for donors who are willing to provide blood whenever needed, especially in an emergency. Additionally, because the data we utilized was created at random using “filldb”, we were only able to conduct a small number of analyses, which we could have easily done with data from the real world. Additionally, making a user-friendly GUI for the entire database could simplify the process of operating on the database.