BLOOD BANK MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

CH.S.K. GOWTHAM

[Reg No: RA2211027010149]

S.SAI CHARANI

[Reg No: RA2211027010186]

Under the Guidance of

DR. SUTHANTHIRA DEVI

Assistant Professor, Department of Data Science and Business Systems

In partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in

COMPUTER SCIENCE AND ENGINEERING

with a specialization in BIG DATA ANALYTICS

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM UNIVERSITY OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this B.Tech project report titled “**BLOOD BANK MANAGEMENT SYSTEM**” is the bonafide work of **Mr.C.H.S.K.Gowtham [Reg. No : RA2211027010149] and Ms. S.SaiCharani [Reg. No.RA2211027010186]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on thebasis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Date:

Dr. Suthanthira Devi
Assistant Professor
Department of Data Science and
Business Systems

Dr. Lakshmi M
HEAD OF THE DEPARTMENT
Department of Data
Science and Business
Systems

ABSTRACT

The "Blood Bank Management System " is a user-centric platform designed to optimize the blood donation process. The system begins with a secure recipient login, allowing users to efficiently search for specific blood types within their vicinity. By integrating real-time blood inventory tracking and location-based services, the system provides recipient with instant information on the availability of the desired blood type in nearby areas. This Blood Bank Management System addresses the need for a streamlined, accessible, and community-driven approach to blood donation. By leveraging technology, real-time data, and user-friendly interfaces, the system aims to bridge the gap between donors and recipients, making the blood donation process more efficient, transparent, and responsive to urgent needs.

PROBLEMSTATEMENT

In the realm of blood banking, antiquated manual processes persist, impeding operational efficiency and exacerbating challenges across donor registration, inventory management, and donor engagement. The absence of digital infrastructure within blood banks results in disparate data sources, leading to inaccuracies in donor records and cumbersome inventory tracking processes. Moreover, limited avenues for donor communication hinder effective engagement strategies, impacting donor retention and the overall blood supply chain. To navigate these pressing challenges and ensure the uninterrupted availability of life-saving blood products, there is an imperative for a modernized blood bank management system. Such a system would revolutionize donor registration processes, implement robust inventory management capabilities for real-time monitoring, and facilitate seamless communication channels to foster meaningful connections with donors. By leveraging automation and digitalization, this system aims to transform the blood donation landscape, optimizing resource utilization and ensuring a sustainable supply of blood products for those in need

TABLE OF CONTENTS

Chapter No	Chapter Name	Page No
1.	Problem understanding, Identification of Entity and Relationships, Construction of DB using ER Model for the project	**2-3**
2.	Design of Relational Schemas, Creation of Database Tables for the project.	**4-6**
3.	Complex queries based on the concepts of constraints, sets, joins, views, Triggers and Cursors.	**7-9**
4.	Analyzing the pitfalls, identifying the dependencies, and applying normalizations	**10-12**
5.	Implementation of concurrency control and recovery mechanisms	**13-20**
6.	Code for the project	**21-23**
7.	Result and Discussion (Screen shots of the implementation with front end.	**24-26**
8.	Attach the Real Time project certificate / Online course certificate	**27**

CHAPTER 1

1.1) Problem understanding:

"Blood banks play a crucial role in ensuring a stable supply of safe blood for transfusions in healthcare facilities. However, many blood banks face challenges in efficiently managing their operations, including donor registration, inventory tracking, blood testing, and transfusion management. Existing manual or outdated systems often result in inefficiencies, errors, and delays, compromising the timely availability and safety of blood products. Moreover, ensuring compliance with regulatory standards and maintaining data security pose additional challenges. Therefore, there is a pressing need for a comprehensive and automated Blood Bank Management System (BBMS) that can address these issues effectively, streamline processes, enhance blood safety, and ensure regulatory compliance."

1.2) Identification of entities and Relationships

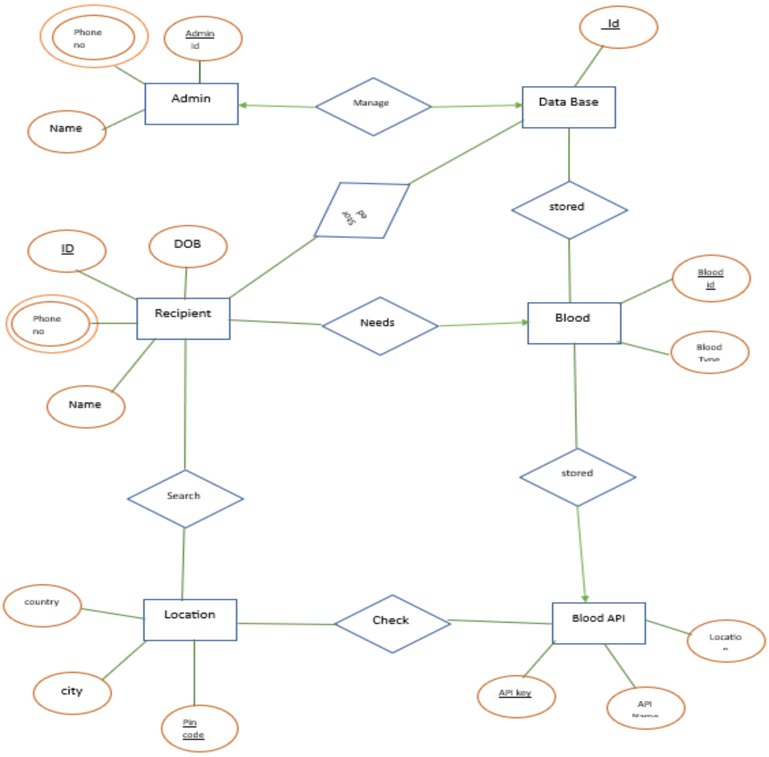
Entities:

- Blood Bank
- Admin
- Recipient
- Blood
- Location
- Blood API

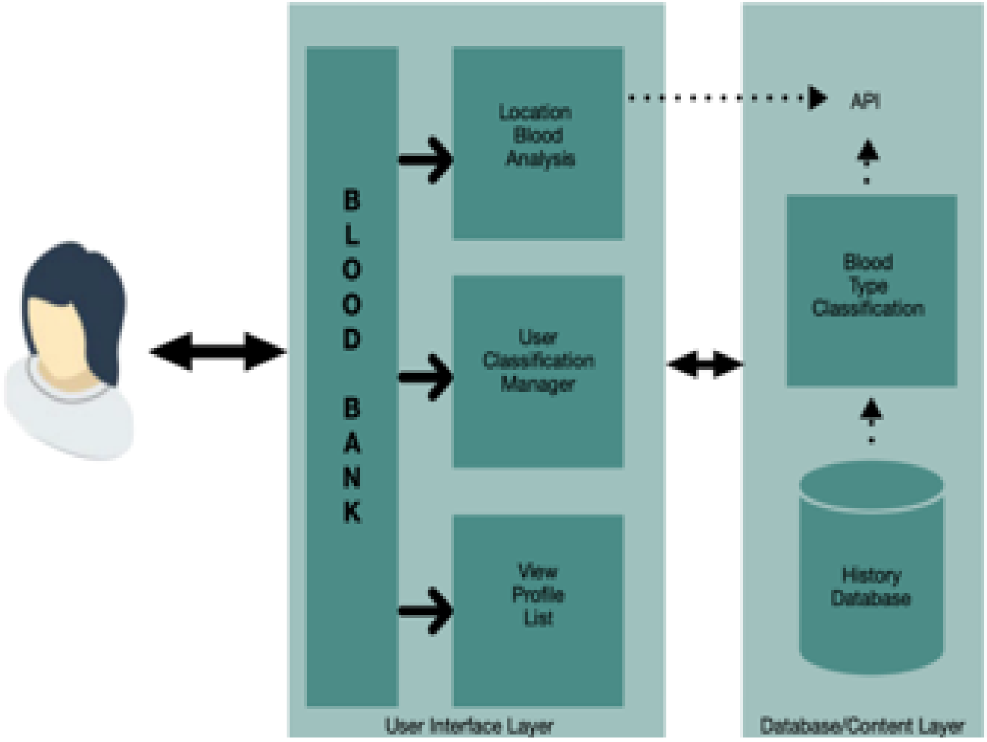
Relationships:

- Admin manages blood bank
- Admin stores data in database
- Blood bank stores blood
- Recipient needs blood from blood bank
- Blood Bank checks blood with Blood API
- Blood API uses location API

1.3) Construction of DB using ER Model for the project

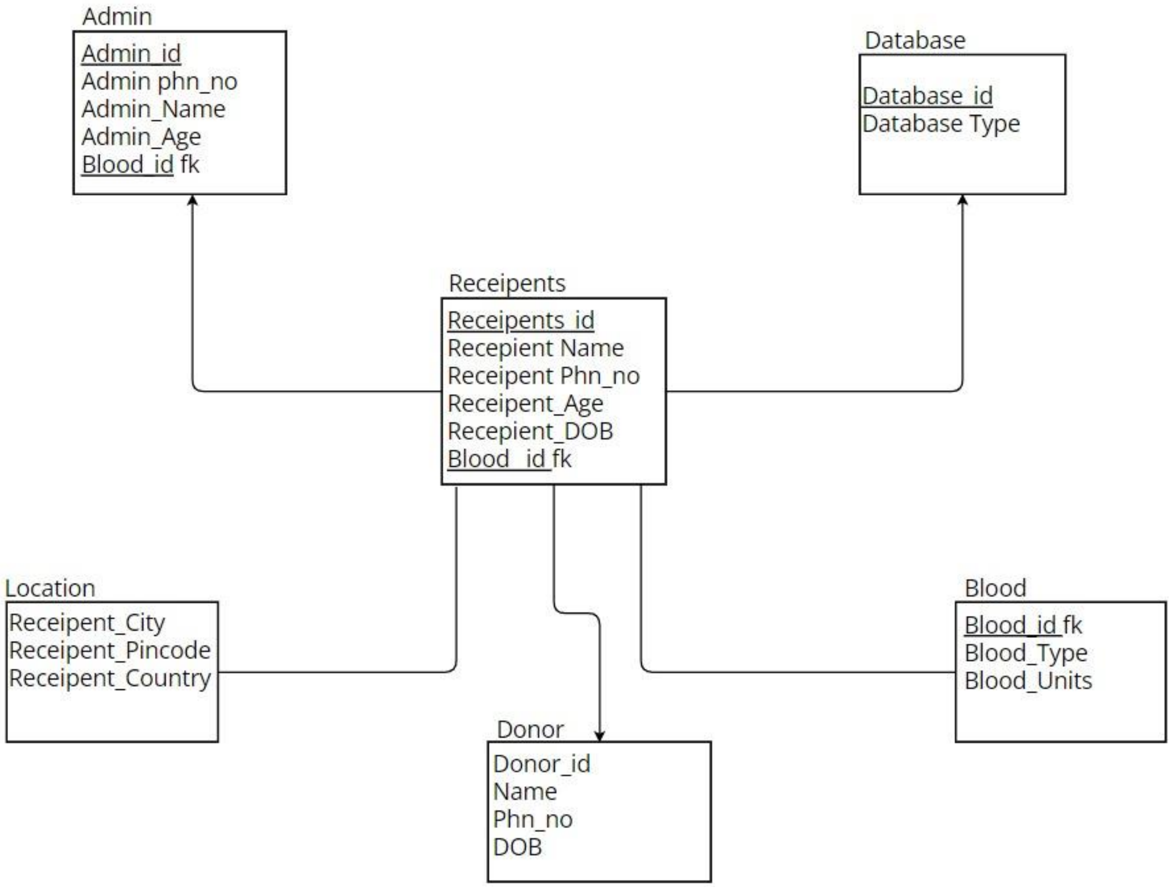


1.4) ARCHITECTURE DIAGRAM



CHAPTER 2

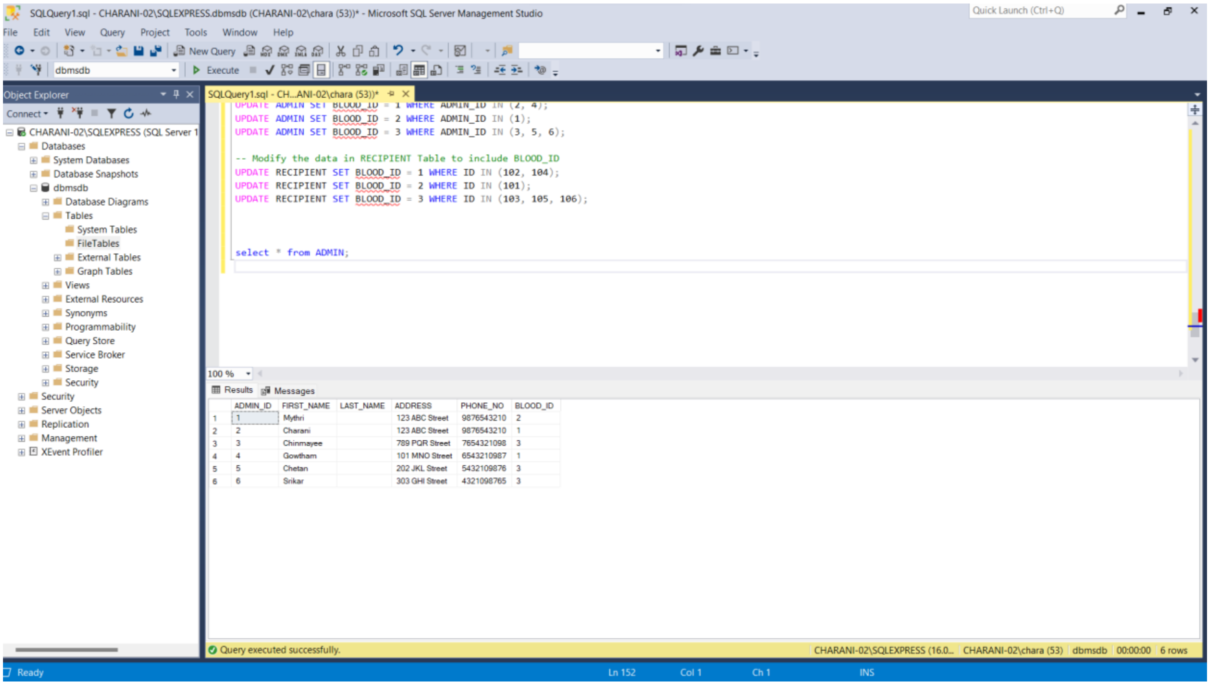
2.1) Design of Relational Schemas:



2.2) Schemas

- Admin(Admin_id,First Name,Last Name,Address,Phone_no,Blood_id)
- Database(Database_id,Database Type)
- Receipent(Id,First Name,Last Name,Phn_no,DOB,Age,Blood_id)
- Blood(Blood_id,Blood Type,Blood_units)
- Location(City,Country,Pincode)
- Donor(Donor_id,Name,Phn_no,DOB)

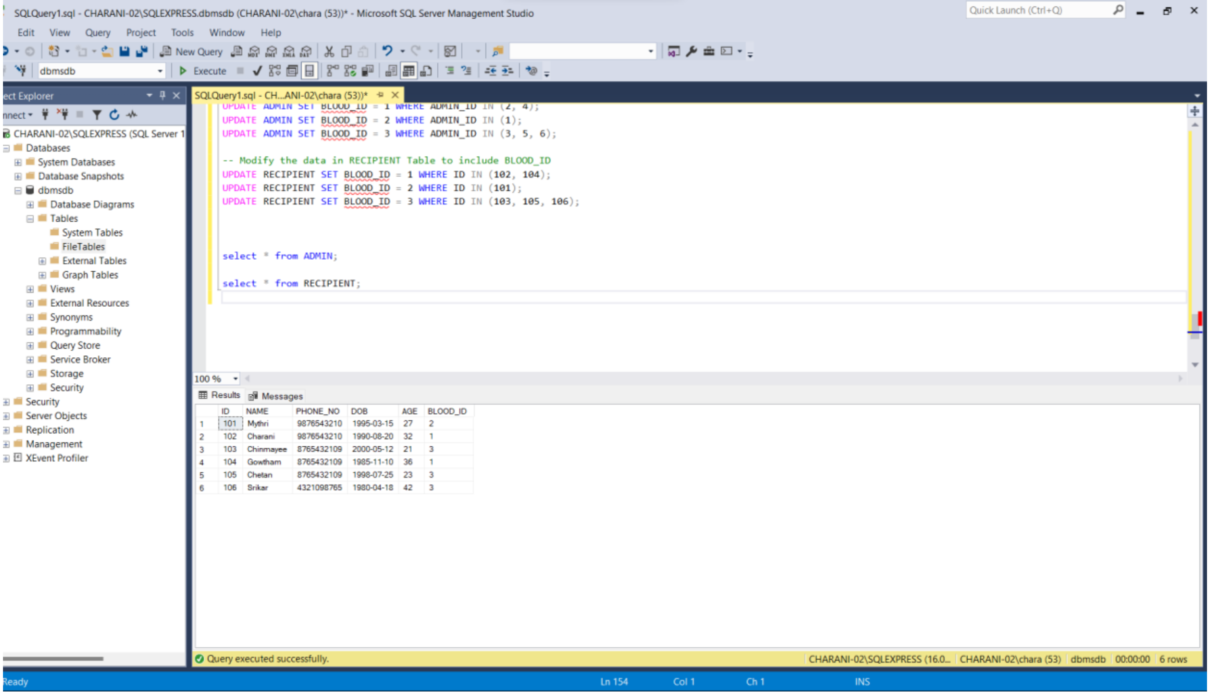
2.3) Creation of Database Tables for the project.



```
UPDATE ADMIN SET BLOOD_ID = 1 WHERE ADMIN_ID IN (2, 4);
UPDATE ADMIN SET BLOOD_ID = 2 WHERE ADMIN_ID IN (1);
UPDATE ADMIN SET BLOOD_ID = 3 WHERE ADMIN_ID IN (3, 5, 6);

-- Modify the data in RECIPIENT Table to include BLOOD_ID
UPDATE RECIPIENT SET BLOOD_ID = 1 WHERE ID IN (102, 104);
UPDATE RECIPIENT SET BLOOD_ID = 2 WHERE ID IN (101);
UPDATE RECIPIENT SET BLOOD_ID = 3 WHERE ID IN (103, 105, 106);



select * from ADMIN;
```

	ADMIN_ID	FIRST_NAME	LAST_NAME	ADDRESS	PHONE_NO	BLOOD_ID
1	1	Mythri		123 ABC Street	9876543210	2
2	2	Charani		123 ABC Street	9876543210	1
3	3	Chinmayee		789 PQR Street	7654321098	3
4	4	Gowtham		101 MNO Street	6543210987	1
5	5	Chetan		202 JKL Street	5432109876	3
6	6	Srikar		303 GHI Street	4321098765	3

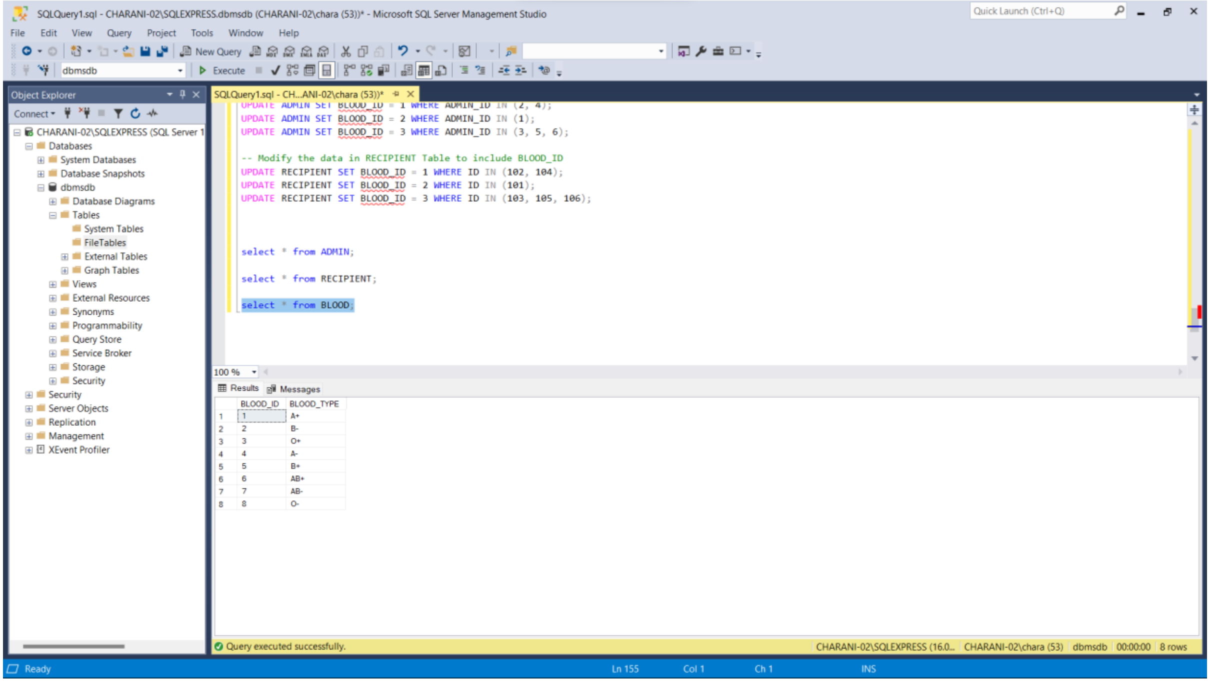
Query executed successfully. CHARANI-02\SQLEXPRESS (16.0... CHARANI-02\chara (53) dbmsdb 00:00:00 6 rows



```
UPDATE ADMIN SET BLOOD_ID = 1 WHERE ADMIN_ID IN (2, 4);
UPDATE ADMIN SET BLOOD_ID = 2 WHERE ADMIN_ID IN (1);
UPDATE ADMIN SET BLOOD_ID = 3 WHERE ADMIN_ID IN (3, 5, 6);

-- Modify the data in RECIPIENT Table to include BLOOD_ID
UPDATE RECIPIENT SET BLOOD_ID = 1 WHERE ID IN (102, 104);
UPDATE RECIPIENT SET BLOOD_ID = 2 WHERE ID IN (101);
UPDATE RECIPIENT SET BLOOD_ID = 3 WHERE ID IN (103, 105, 106);


select * from ADMIN;

select * from RECIPIENT;
```

	ID	NAME	PHONE_NO	DOB	AGE	BLOOD_ID
1	101	Mythri	9876543210	1995-03-15	27	2
2	102	Charani	9876543210	1990-08-20	32	1
3	103	Chinmayee	8765432109	2000-05-12	21	3
4	104	Gowtham	8765432109	1985-11-10	36	1
5	105	Chetan	8765432109	1998-07-25	23	3
6	106	Srikar	4321098765	1980-04-18	42	3

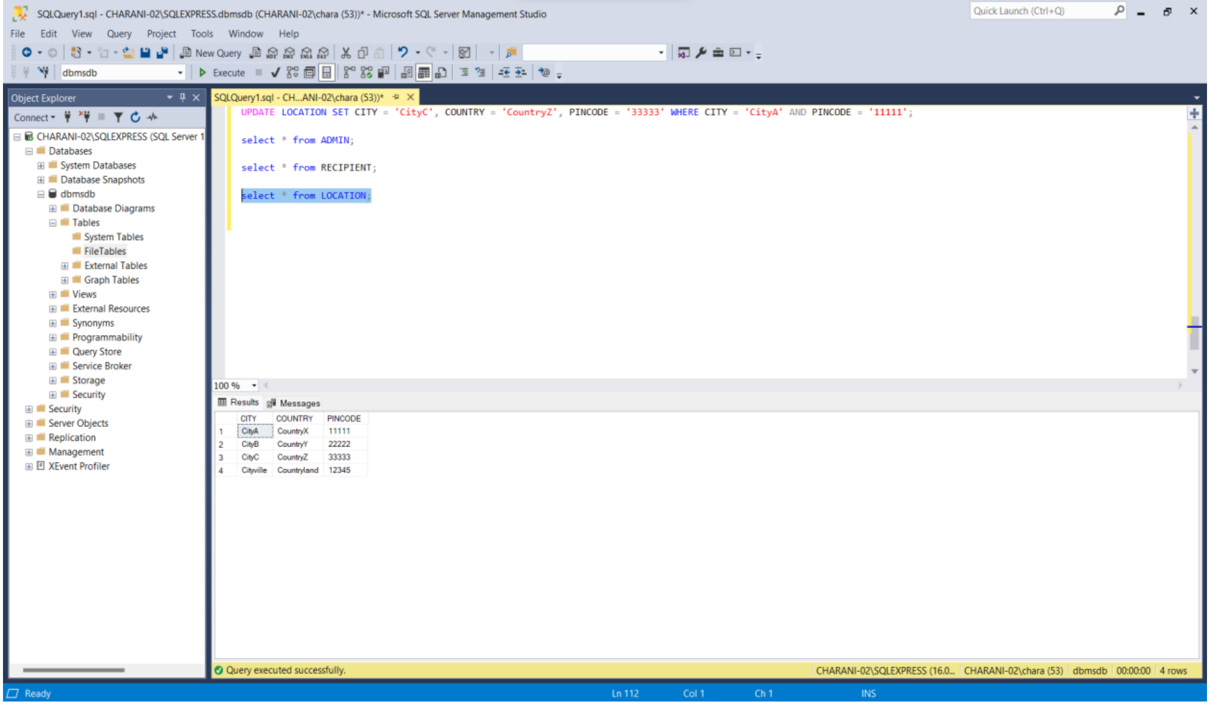
Query executed successfully. CHARANI-02\SQLEXPRESS (16.0... CHARANI-02\chara (53) dbmsdb 00:00:00 6 rows



```
UPDATE ADMIN SET BLOOD_ID = 1 WHERE ADMIN_ID IN (2, 4);
UPDATE ADMIN SET BLOOD_ID = 2 WHERE ADMIN_ID IN (1);
UPDATE ADMIN SET BLOOD_ID = 3 WHERE ADMIN_ID IN (3, 5, 6);

-- Modify the data in RECIPIENT Table to include BLOOD_ID
UPDATE RECIPIENT SET BLOOD_ID = 1 WHERE ID IN (102, 104);
UPDATE RECIPIENT SET BLOOD_ID = 2 WHERE ID IN (101);
UPDATE RECIPIENT SET BLOOD_ID = 3 WHERE ID IN (103, 105, 106);



select * from ADMIN;

select * from RECIPIENT;

select * from BLOOD;
```

	BLOOD_ID	BLOOD_TYPE
1	1	A+
2	2	B-
3	3	O+
4	4	A-
5	5	B+
6	6	AB+
7	7	AB-
8	8	O-

Query executed successfully. CHARANI-02\SQLEXPRESS (16.0... CHARANI-02\chara (53) dbmsdb 00:00:00 8 rows



```
UPDATE LOCATION SET CITY = 'CityC', COUNTRY = 'CountryZ', PINCODE = '33333' WHERE CITY = 'CityA' AND PINCODE = '11111';

select * from ADMIN;

select * from RECIPIENT;

select * from LOCATION;
```

	CITY	COUNTRY	PINCODE
1	CityA	CountryX	11111
2	CityB	CountryY	22222
3	CityC	CountryZ	33333
4	Cityville	Countryland	12345

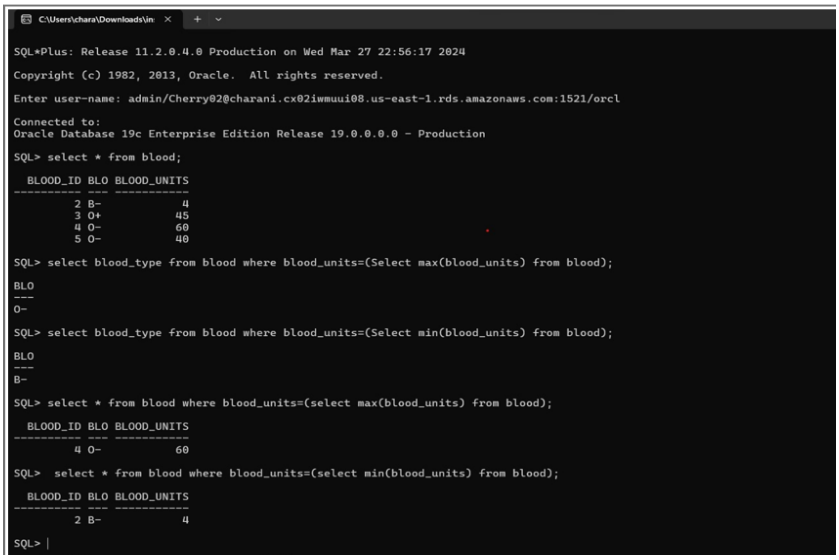
Query executed successfully. CHARANI-02\SQLEXPRESS (16.0... CHARANI-02\chara (53) dbmsdb 00:00:00 4 rows

CHAPTER 3

3.1) Complex queries based on the concepts of constraints:

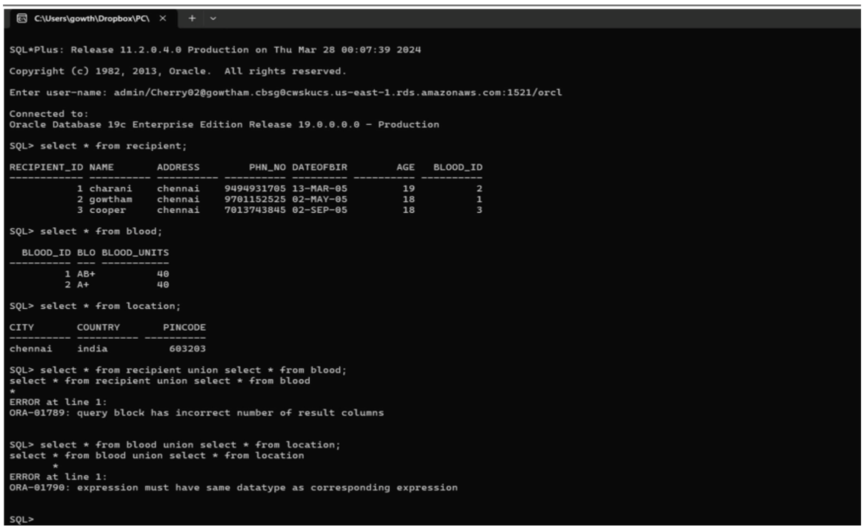
Select blood_type from blood where blood_units=(Select max(blood_units) from blood);

Select *from blood where blood_units=(select max(blood_units) from blood);



```
SQL*Plus: Release 11.2.0.4.0 Production on Wed Mar 27 22:56:17 2024

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter user-name: admin/Cherry02@charani.cx02iwmuui08.us-east-1.rds.amazonaws.com:1521/orcl

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

SQL> select * from blood;

  BLOOD_ID BLO BLOOD_UNITS
---------- --- -----------
         2 B-            4
         3 O+           45
         4 O-           60
         5 O-           40

SQL> select blood_type from blood where blood_units=(Select max(blood_units) from blood);

BLO
---
O-

SQL> select blood_type from blood where blood_units=(Select min(blood_units) from blood);

BLO
---
B-

SQL> select * from blood where blood_units=(select max(blood_units) from blood);

  BLOOD_ID BLO BLOOD_UNITS
---------- --- -----------
         4 O-           60

SQL>  select * from blood where blood_units=(select min(blood_units) from blood);

  BLOOD_ID BLO BLOOD_UNITS
---------- --- -----------
         2 B-            4

SQL> |
```

3.2) Sets

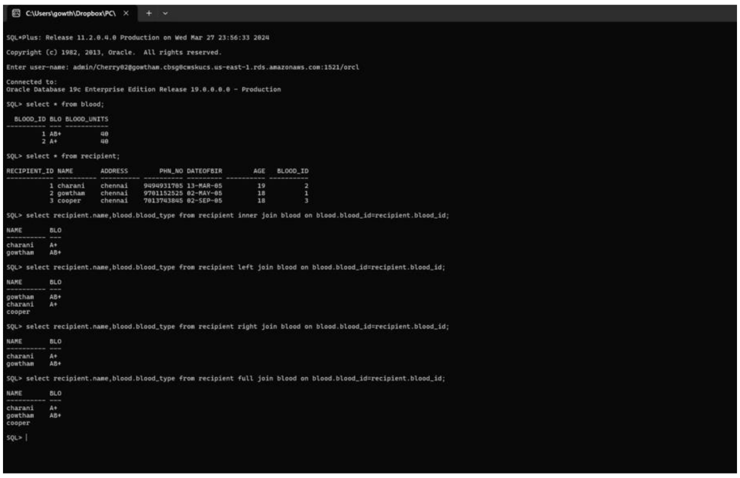


```
SQL*Plus: Release 11.2.0.4.0 Production on Thu Mar 28 00:07:39 2024

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter user-name: admin/Cherry02@gowtham.cbsg0cwskucs.us-east-1.rds.amazonaws.com:1521/orcl

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

SQL> select * from recipient;

RECIPIENT_ID NAME       ADDRESS        PHN_NO DATEOFBIR        AGE   BLOOD_ID
------------ ---------- ---------- ---------- --------- ---------- ----------
           1 charani    chennai    9494931705 13-MAR-05         19          2
           2 gowtham    chennai    9701152525 02-MAY-05         18          1
           3 cooper     chennai    7013743845 02-SEP-05         18          3

SQL> select * from blood;

  BLOOD_ID BLO BLOOD_UNITS
---------- --- -----------
         1 AB+          40
         2 A+           40

SQL> select * from location;

CITY       COUNTRY       PINCODE
---------- ---------- ----------
chennai    india          603203

SQL> select * from recipient union select * from blood;
select * from recipient union select * from blood
*
ERROR at line 1:
ORA-01789: query block has incorrect number of result columns


SQL> select * from blood union select * from location;
select * from blood union select * from location
       *
ERROR at line 1:
ORA-01790: expression must have same datatype as corresponding expression


SQL>
```

3.3) Joins:

select recipient.name,blood.blood_type from recipient inner join on blood.blood_id = recipient.blood_id;

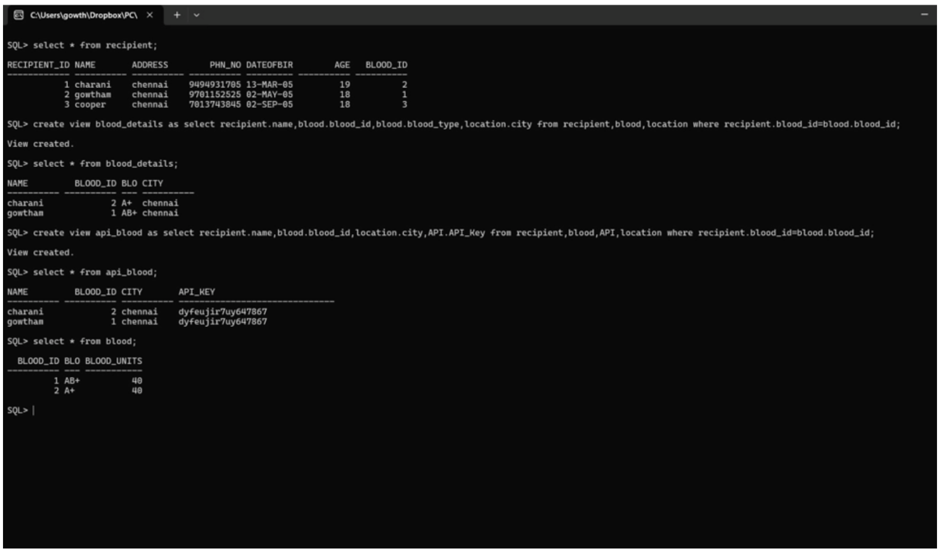
select recipient.name,blood.blood_type from recipient left join on blood.blood_id=recipient.blood_id;



```
SQL*Plus: Release 11.2.0.4.0 Production on Wed Mar 27 23:56:33 2024

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter user-name: admin/Cherry02@gowtham.cbsg0cwskucs.us-east-1.rds.amazonaws.com:1521/orcl

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

SQL> select * from blood;

  BLOOD_ID BLO BLOOD_UNITS
---------- --- -----------
         1 AB+          40
         2 A+           40

SQL> select * from recipient;

RECIPIENT_ID NAME       ADDRESS        PHN_NO DATEOFBIR        AGE   BLOOD_ID
------------ ---------- ---------- ---------- --------- ---------- ----------
           1 charani    chennai    9494931705 13-MAR-05         19          2
           2 gowtham    chennai    9701152525 02-MAY-05         18          1
           3 cooper     chennai    7013743845 02-SEP-05         18          3

SQL> select recipient.name,blood.blood_type from recipient inner join blood on blood.blood_id=recipient.blood_id;

NAME       BLO
---------- ---
charani    A+
gowtham    AB+

SQL> select recipient.name,blood.blood_type from recipient left join blood on blood.blood_id=recipient.blood_id;

NAME       BLO
---------- ---
gowtham    AB+
charani    A+
cooper

SQL> select recipient.name,blood.blood_type from recipient right join blood on blood.blood_id=recipient.blood_id;

NAME       BLO
---------- ---
charani    A+
gowtham    AB+

SQL> select recipient.name,blood.blood_type from recipient full join blood on blood.blood_id=recipient.blood_id;

NAME       BLO
---------- ---
charani    A+
gowtham    AB+
cooper

SQL> |
```

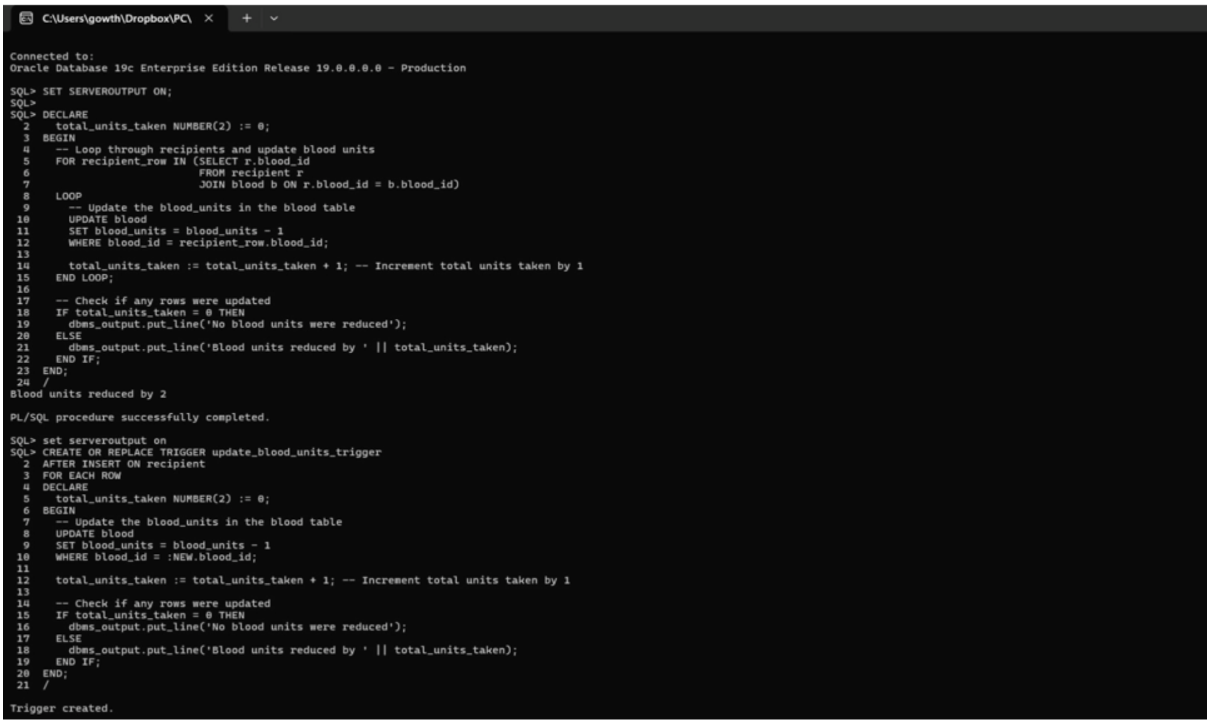
3.4) Views

Create view blood_details as select recipient.name,blood_id,blood_type,location.city from recipient,blood,location where recipient.blood_id=blood.blood_id;

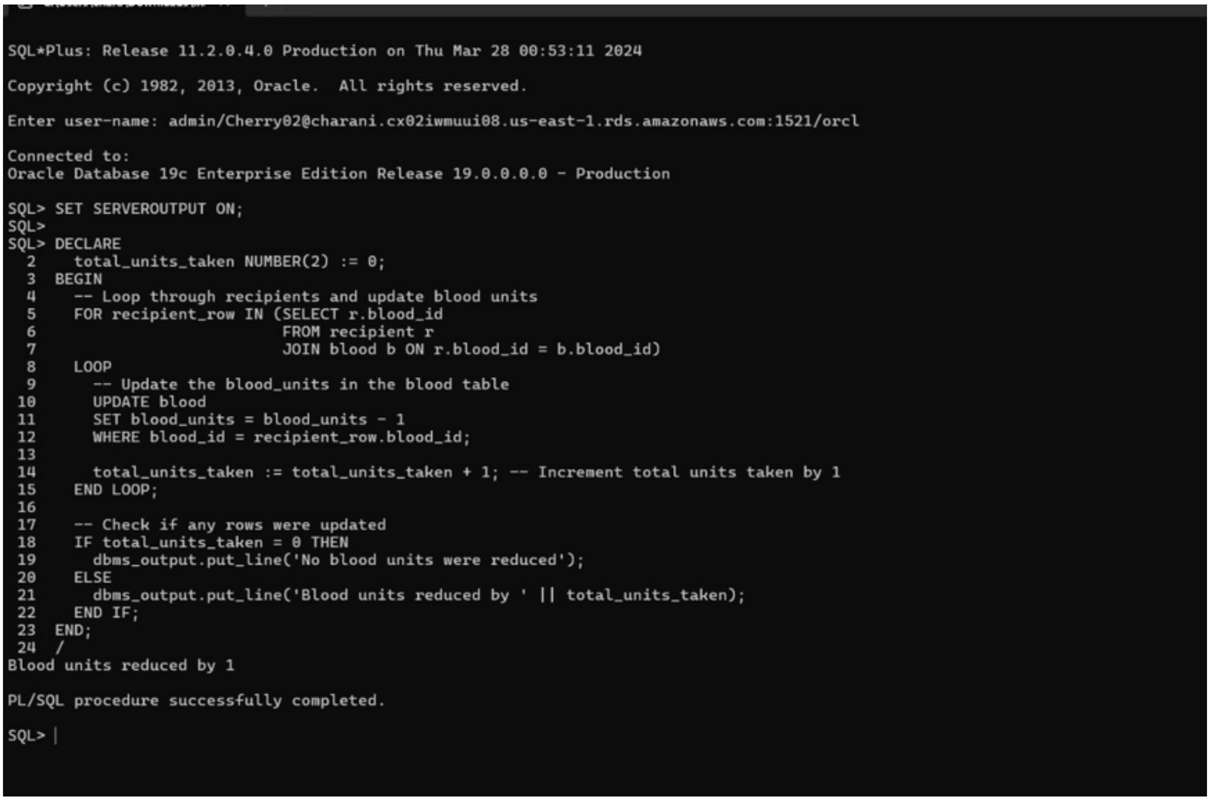


```
SQL> select * from recipient;

RECIPIENT_ID NAME       ADDRESS        PHN_NO DATEOFBIR        AGE   BLOOD_ID
------------ ---------- ---------- ---------- --------- ---------- ----------
           1 charani    chennai    9494931705 13-MAR-05         19          2
           2 gowtham    chennai    9701152525 02-MAY-05         18          1
           3 cooper     chennai    7013743845 02-SEP-05         18          3

SQL> create view blood_details as select recipient.name,blood.blood_id,blood.blood_type,location.city from recipient,blood,location where recipient.blood_id=blood.blood_id;

View created.

SQL> select * from blood_details;

NAME         BLOOD_ID BLO CITY
---------- ---------- --- ----------
charani             2 A+  chennai
gowtham             1 AB+ chennai

SQL> create view api_blood as select recipient.name,blood.blood_id,location.city,API.API_Key from recipient,blood,API,location where recipient.blood_id=blood.blood_id;

View created.

SQL> select * from api_blood;

NAME         BLOOD_ID CITY       API_KEY
---------- ---------- ---------- ------------------------------
charani             2 chennai    dyfeujir7uy647867
gowtham             1 chennai    dyfeujir7uy647867

SQL> select * from blood;

  BLOOD_ID BLO BLOOD_UNITS
---------- --- -----------
         1 AB+          40
         2 A+           40

SQL> |
```

3.5) Triggers and Cursors



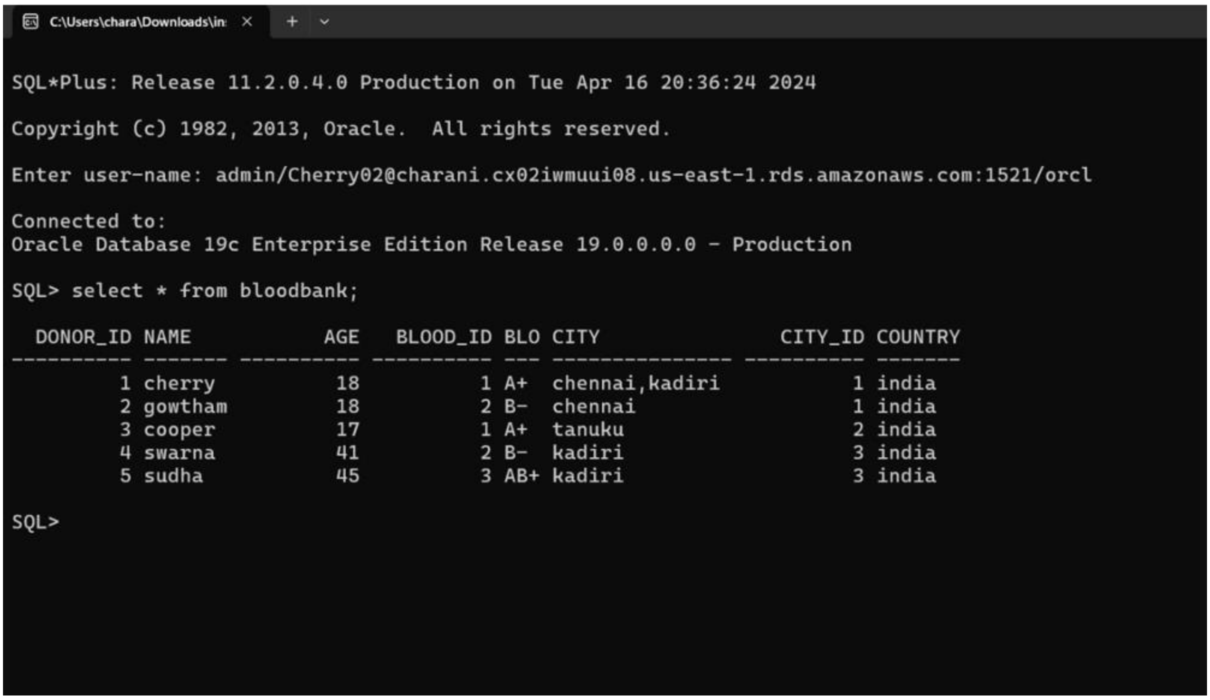
```
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

SQL> SET SERVEROUTPUT ON;
SQL>
SQL> DECLARE
       total_units_taken NUMBER(2) := 0;
     BEGIN
       -- Loop through recipients and update blood units
       FOR recipient_row IN (SELECT r.blood_id
                             FROM recipient r
                             JOIN blood b ON r.blood_id = b.blood_id)
       LOOP
         -- Update the blood_units in the blood table
         UPDATE blood
         SET blood_units = blood_units - 1
         WHERE blood_id = recipient_row.blood_id;

         total_units_taken := total_units_taken + 1; -- Increment total units taken by 1
       END LOOP;

       -- Check if any rows were updated
       IF total_units_taken = 0 THEN
         dbms_output.put_line('No blood units were reduced');
       ELSE
         dbms_output.put_line('Blood units reduced by ' || total_units_taken);
       END IF;
     END;
     /
Blood units reduced by 2

PL/SQL procedure successfully completed.

SQL> set serveroutput on
SQL> CREATE OR REPLACE TRIGGER update_blood_units_trigger
     AFTER INSERT ON recipient
     FOR EACH ROW
     DECLARE
       total_units_taken NUMBER(2) := 0;
     BEGIN
       -- Update the blood_units in the blood table
       UPDATE blood
       SET blood_units = blood_units - 1
       WHERE blood_id = :NEW.blood_id;

       total_units_taken := total_units_taken + 1; -- Increment total units taken by 1

       -- Check if any rows were updated
       IF total_units_taken = 0 THEN
         dbms_output.put_line('No blood units were reduced');
       ELSE
         dbms_output.put_line('Blood units reduced by ' || total_units_taken);
       END IF;
     END;
     /

Trigger created.
```



```
SQL*Plus: Release 11.2.0.4.0 Production on Thu Mar 28 00:53:11 2024

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter user-name: admin/Cherry02@charani.cx02iwmuui08.us-east-1.rds.amazonaws.com:1521/orcl

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

SQL> SET SERVEROUTPUT ON;
SQL>
SQL> DECLARE
       total_units_taken NUMBER(2) := 0;
     BEGIN
       -- Loop through recipients and update blood units
       FOR recipient_row IN (SELECT r.blood_id
                             FROM recipient r
                             JOIN blood b ON r.blood_id = b.blood_id)
       LOOP
         -- Update the blood_units in the blood table
         UPDATE blood
         SET blood_units = blood_units - 1
         WHERE blood_id = recipient_row.blood_id;

         total_units_taken := total_units_taken + 1; -- Increment total units taken by 1
       END LOOP;

       -- Check if any rows were updated
       IF total_units_taken = 0 THEN
         dbms_output.put_line('No blood units were reduced');
       ELSE
         dbms_output.put_line('Blood units reduced by ' || total_units_taken);
       END IF;
     END;
     /
Blood units reduced by 1

PL/SQL procedure successfully completed.

SQL> |
```

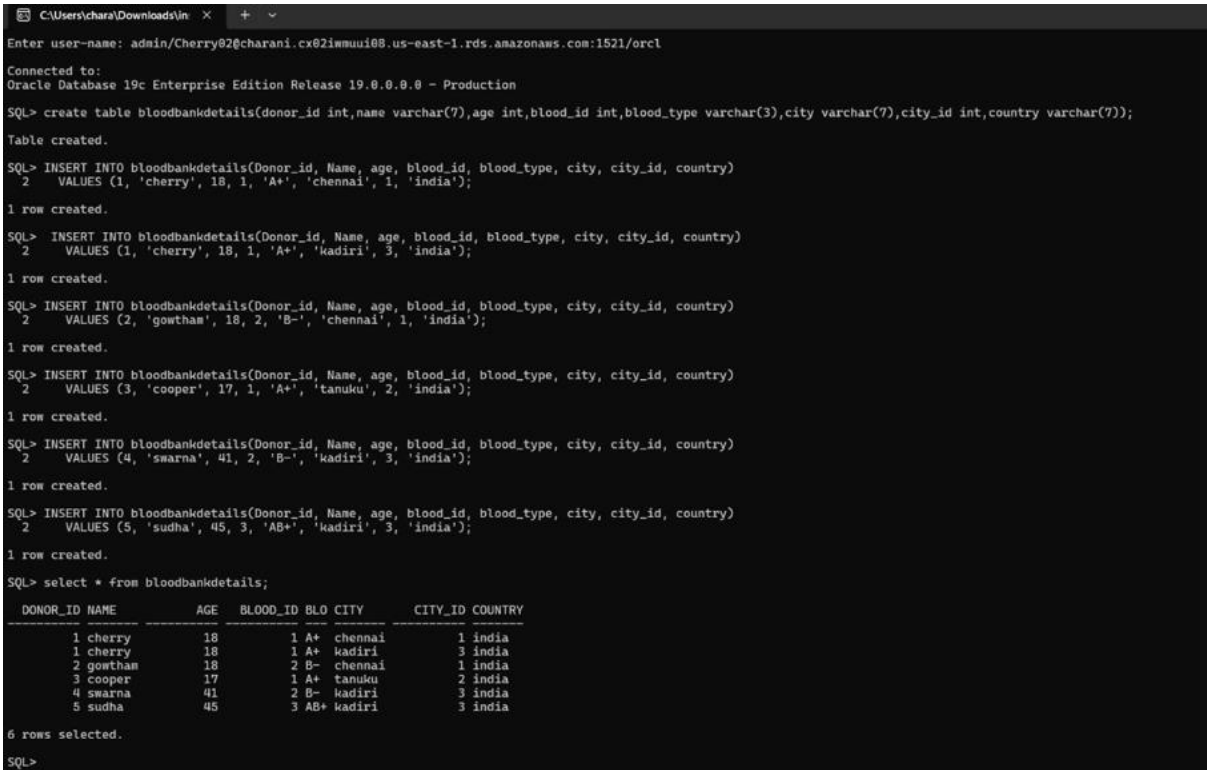
CHAPTER 4

4.1) Unnormalized Table



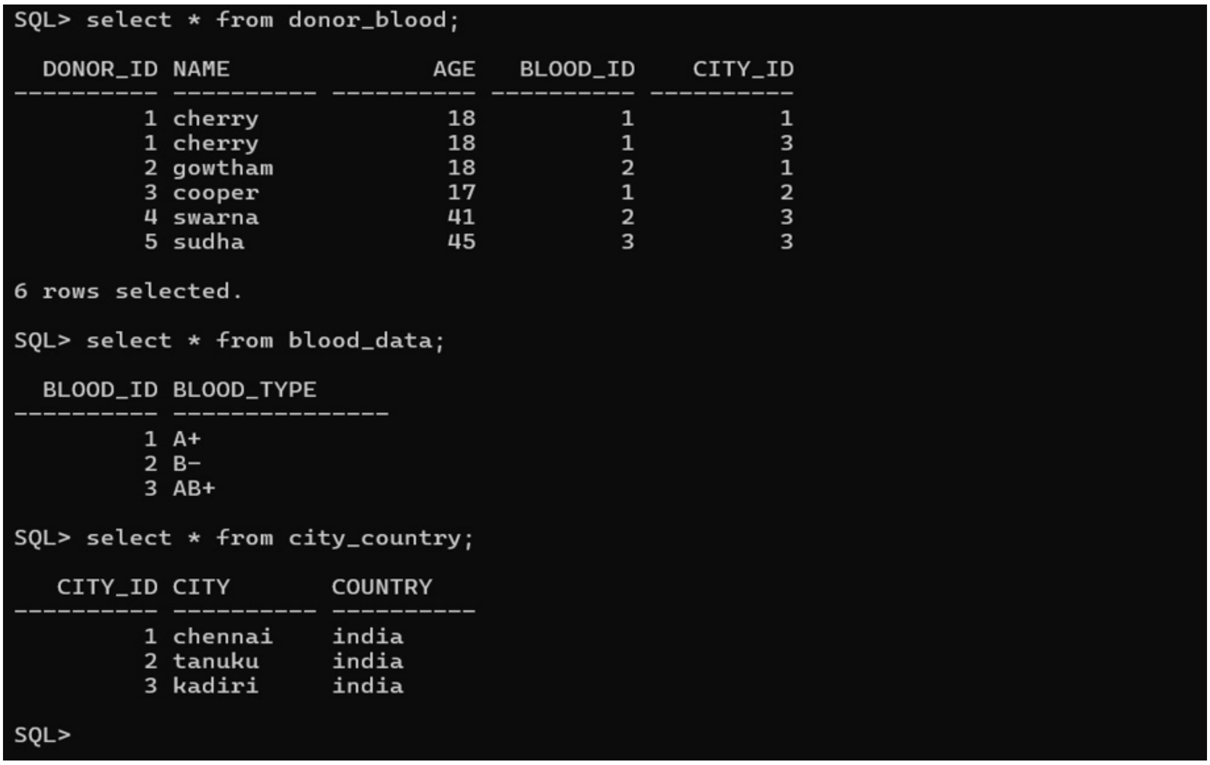
```
SQL*Plus: Release 11.2.0.4.0 Production on Tue Apr 16 20:36:24 2024

Copyright (c) 1982, 2013, Oracle.  All rights reserved.

Enter user-name: admin/Cherry02@charani.cx02iwmuui08.us-east-1.rds.amazonaws.com:1521/orcl

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

SQL> select * from bloodbank;

  DONOR_ID NAME           AGE   BLOOD_ID BLO CITY              CITY_ID COUNTRY
---------- ------- ---------- ---------- --- --------------- ---------- -------
         1 cherry          18          1 A+  chennai,kadiri           1 india
         2 gowtham         18          2 B-  chennai                  1 india
         3 cooper          17          1 A+  tanuku                   2 india
         4 swarna          41          2 B-  kadiri                   3 india
         5 sudha           45          3 AB+ kadiri                   3 india

SQL>
```

1NF



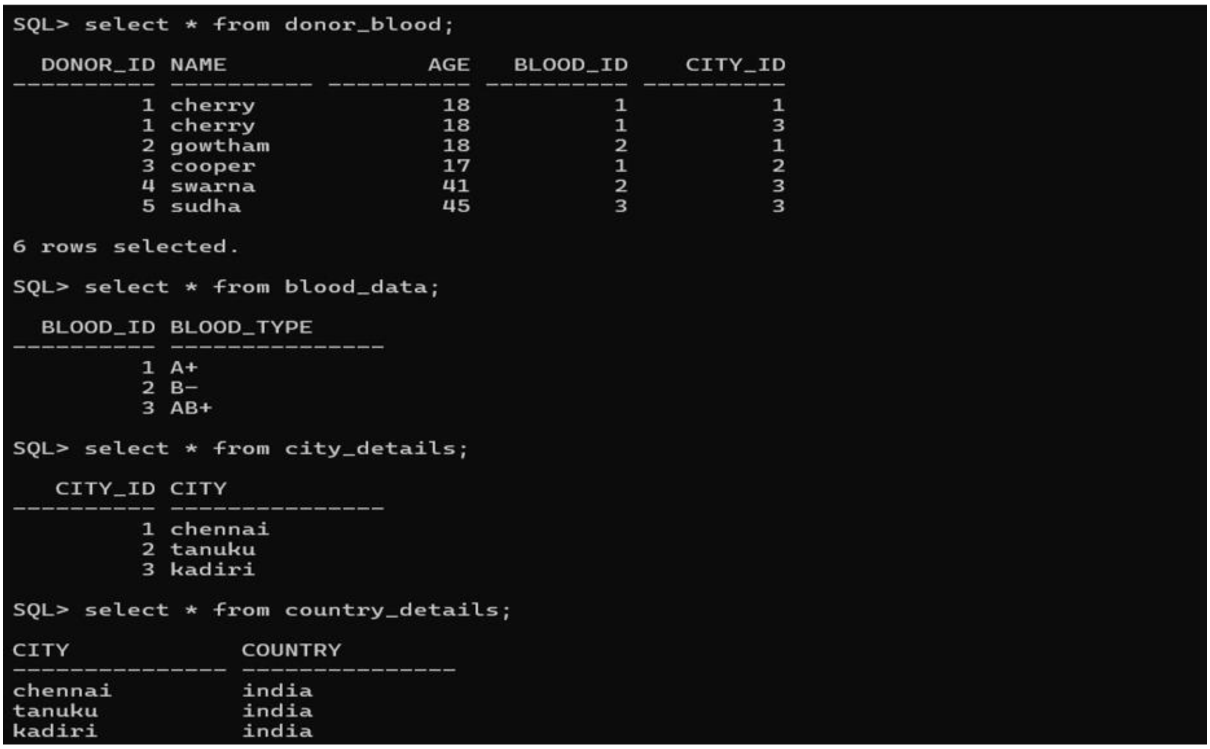
```
Enter user-name: admin/Cherry02@charani.cx02iwmuui08.us-east-1.rds.amazonaws.com:1521/orcl

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production

SQL> create table bloodbankdetails(donor_id int,name varchar(7),age int,blood_id int,blood_type varchar(3),city varchar(7),city_id int,country varchar(7));

Table created.

SQL> INSERT INTO bloodbankdetails(Donor_id, Name, age, blood_id, blood_type, city, city_id, country)
  2    VALUES (1, 'cherry', 18, 1, 'A+', 'chennai', 1, 'india');

1 row created.

SQL>  INSERT INTO bloodbankdetails(Donor_id, Name, age, blood_id, blood_type, city, city_id, country)
  2     VALUES (1, 'cherry', 18, 1, 'A+', 'kadiri', 3, 'india');

1 row created.

SQL> INSERT INTO bloodbankdetails(Donor_id, Name, age, blood_id, blood_type, city, city_id, country)
  2     VALUES (2, 'gowtham', 18, 2, 'B-', 'chennai', 1, 'india');

1 row created.

SQL> INSERT INTO bloodbankdetails(Donor_id, Name, age, blood_id, blood_type, city, city_id, country)
  2     VALUES (3, 'cooper', 17, 1, 'A+', 'tanuku', 2, 'india');

1 row created.

SQL> INSERT INTO bloodbankdetails(Donor_id, Name, age, blood_id, blood_type, city, city_id, country)
  2     VALUES (4, 'swarna', 41, 2, 'B-', 'kadiri', 3, 'india');

1 row created.

SQL> INSERT INTO bloodbankdetails(Donor_id, Name, age, blood_id, blood_type, city, city_id, country)
  2     VALUES (5, 'sudha', 45, 3, 'AB+', 'kadiri', 3, 'india');

1 row created.

SQL> select * from bloodbankdetails;

  DONOR_ID NAME           AGE   BLOOD_ID BLO CITY       CITY_ID COUNTRY
---------- ------- ---------- ---------- --- ------- ---------- -------
         1 cherry          18          1 A+  chennai          1 india
         1 cherry          18          1 A+  kadiri           3 india
         2 gowtham         18          2 B-  chennai          1 india
         3 cooper          17          1 A+  tanuku           2 india
         4 swarna          41          2 B-  kadiri           3 india
         5 sudha           45          3 AB+ kadiri           3 india

6 rows selected.

SQL>
```

2NF



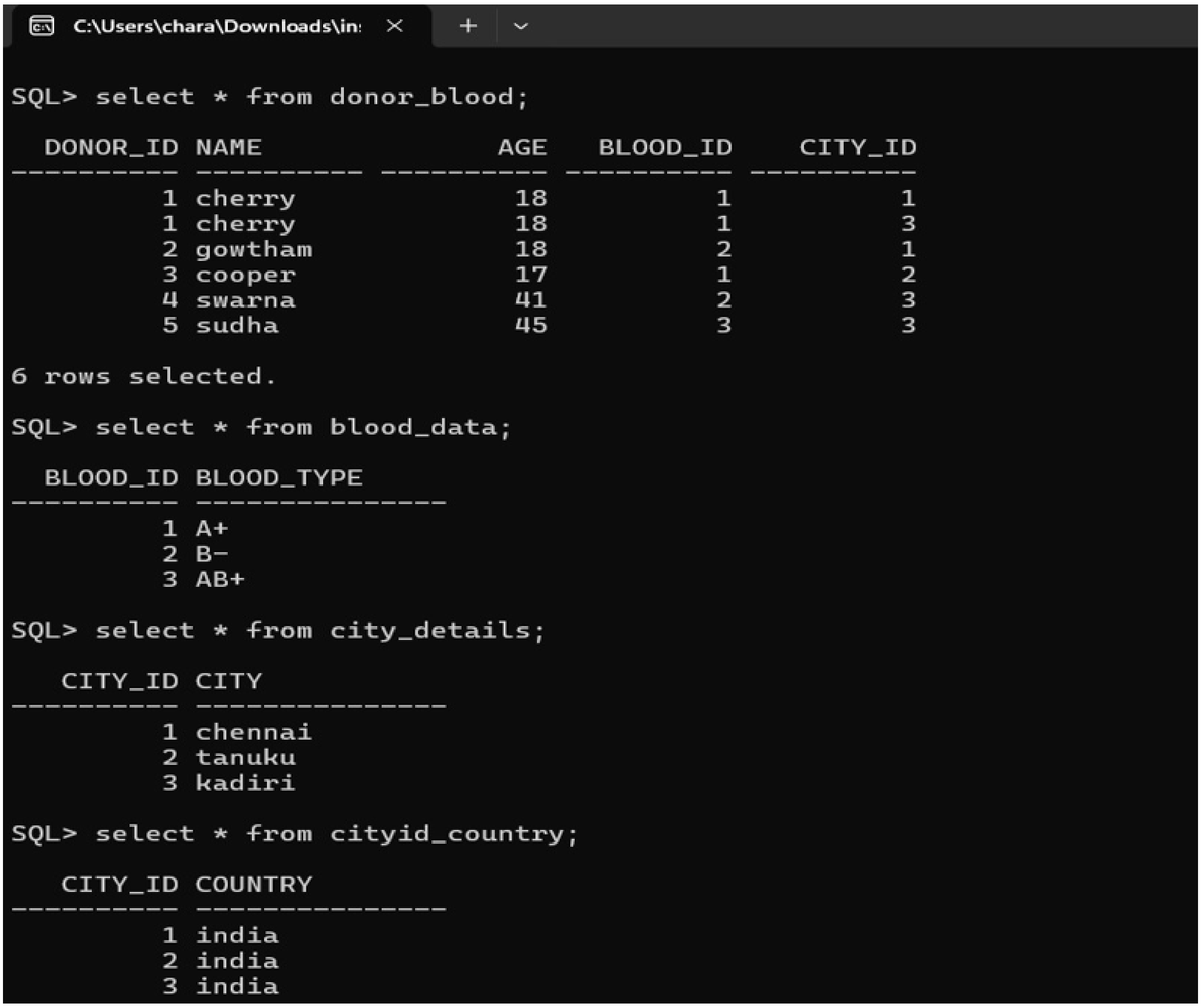
```
SQL> select * from donor_blood;

  DONOR_ID NAME              AGE   BLOOD_ID    CITY_ID
---------- ---------- ---------- ---------- ----------
         1 cherry             18          1          1
         1 cherry             18          1          3
         2 gowtham            18          2          1
         3 cooper             17          1          2
         4 swarna             41          2          3
         5 sudha              45          3          3

6 rows selected.

SQL> select * from blood_data;

  BLOOD_ID BLOOD_TYPE
---------- ---------------
         1 A+
         2 B-
         3 AB+

SQL> select * from city_country;

   CITY_ID CITY       COUNTRY
---------- ---------- ----------
         1 chennai    india
         2 tanuku     india
         3 kadiri     india

SQL>
```

3NF



```
SQL> select * from donor_blood;

  DONOR_ID NAME              AGE   BLOOD_ID    CITY_ID
---------- ---------- ---------- ---------- ----------
         1 cherry             18          1          1
         1 cherry             18          1          3
         2 gowtham            18          2          1
         3 cooper             17          1          2
         4 swarna             41          2          3
         5 sudha              45          3          3

6 rows selected.

SQL> select * from blood_data;

  BLOOD_ID BLOOD_TYPE
---------- ---------------
         1 A+
         2 B-
         3 AB+

SQL> select * from city_details;

   CITY_ID CITY
---------- ---------------
         1 chennai
         2 tanuku
         3 kadiri

SQL> select * from country_details;

CITY            COUNTRY
--------------- ---------------
chennai         india
tanuku          india
kadiri          india
```

BCNF



```
SQL> select * from donor_blood;

  DONOR_ID NAME                   AGE   BLOOD_ID    CITY_ID
---------- ---------- ---------- ---------- ----------
         1 cherry                  18          1          1
         1 cherry                  18          1          3
         2 gowtham                 18          2          1
         3 cooper                  17          1          2
         4 swarna                  41          2          3
         5 sudha                   45          3          3

6 rows selected.

SQL> select * from blood_data;

  BLOOD_ID BLOOD_TYPE
---------- ---------------
         1 A+
         2 B-
         3 AB+

SQL> select * from city_details;

   CITY_ID CITY
---------- ---------------
         1 chennai
         2 tanuku
         3 kadiri

SQL> select * from cityid_country;

   CITY_ID COUNTRY
---------- ---------------
         1 india
         2 india
         3 india
```

CHAPTER 5

TRANSACTION CONTROL & CONCURRENCY CONTROL

Satisfying ACID Properties:

ATOMICITY:

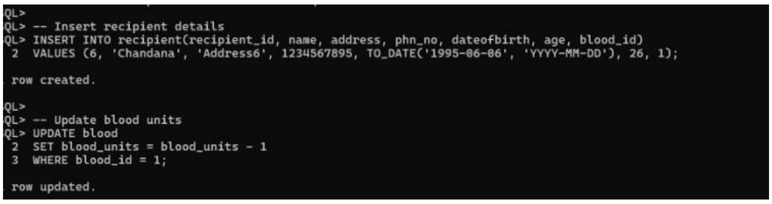
Scenario:

A new recipient is being added to the recipient table, but while inserting the recipient's details, the system fails after the recipient's details are inserted, but before updating the blood bank's available blood units.

Solution:

Use a transaction to ensure atomicity. If any part of the transaction fails, all changes made in that transaction should be rolled back

MySQL Query:



```
QL>
QL> -- Insert recipient details
QL> INSERT INTO recipient(recipient_id, name, address, phn_no, dateofbirth, age, blood_id)
 2  VALUES (6, 'Chandana', 'Address6', 1234567895, TO_DATE('1995-06-06', 'YYYY-MM-DD'), 26, 1);

 row created.

QL>
QL> -- Update blood units
QL> UPDATE blood
 2  SET blood_units = blood_units - 1
 3  WHERE blood_id = 1;

 row updated.
```

Explanation:

- START TRANSACTION; begins a new transaction.
- INSERT INTO recipient... inserts a new recipient's details.
- UPDATE blood... decreases the available blood units.
- COMMIT; commits the transaction. If no errors occur, changes made within the transaction are saved to the database. If any part of the transaction fails, the ROLLBACK; command would undo any changes made within the transaction.

CONSISTENCY:

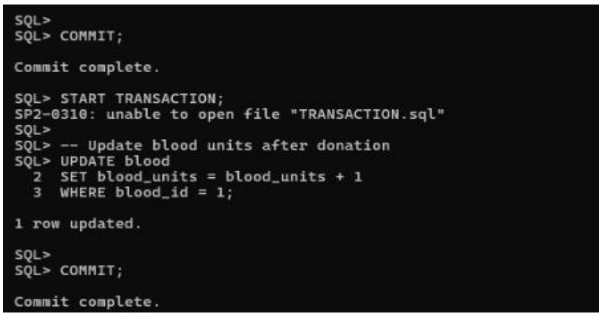
Scenario:

A donor donates blood, and the system needs to ensure that after the donation, the total number of blood units is updated and consistent.

Solution:

Use transactions to ensure that the total number of blood units is consistent after a donation.

MySQL Query:



```
SQL>
SQL> COMMIT;

Commit complete.

SQL> START TRANSACTION;
SP2-0310: unable to open file "TRANSACTION.sql"
SQL>
SQL> -- Update blood units after donation
SQL> UPDATE blood
  2  SET blood_units = blood_units + 1
  3  WHERE blood_id = 1;

1 row updated.

SQL>
SQL> COMMIT;

Commit complete.
```

Explanation:

- START TRANSACTION; begins a new transaction.
- UPDATE blood... increases the available blood units.
- COMMIT; commits the transaction. If no errors occur, changes made within the transaction are saved to the database. If any part of the transaction fails, the ROLLBACK; command would undo any changes made within the transaction.

ISOLATION:

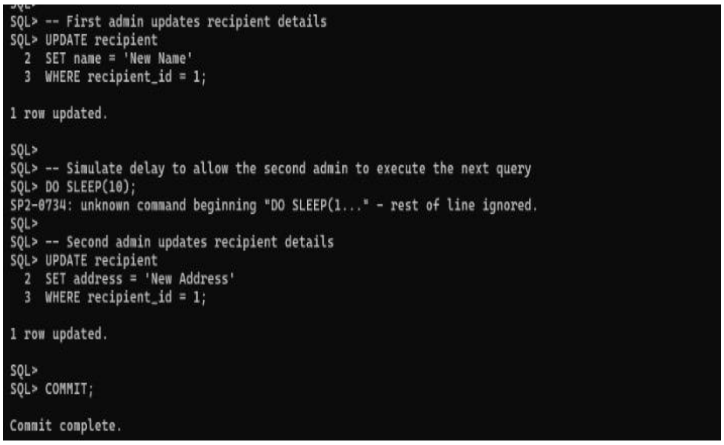
Scenario:

Two administrators simultaneously try to update the same recipient's details

Solution:

Use locking mechanisms to prevent concurrent access to the same data.

MySQL Query:



```
SQL> -- First admin updates recipient details
SQL> UPDATE recipient
  2  SET name = 'New Name'
  3  WHERE recipient_id = 1;

1 row updated.

SQL>
SQL> -- Simulate delay to allow the second admin to execute the next query
SQL> DO SLEEP(10);
SP2-0734: unknown command beginning "DO SLEEP(1..." - rest of line ignored.
SQL>
SQL> -- Second admin updates recipient details
SQL> UPDATE recipient
  2  SET address = 'New Address'
  3  WHERE recipient_id = 1;

1 row updated.

SQL>
SQL> COMMIT;

Commit complete.
```

Explanation:

- START TRANSACTION; begins a new transaction.
- The first UPDATE recipient... query is executed by the first admin.
- DO SLEEP(10); is a MySQL function that pauses execution for 10 seconds.
- The second UPDATE recipient... query is executed by the second admin.
- COMMIT; commits the transaction. MySQL automatically handles locking and ensures that the changes made by one transaction are isolated from the changes made by other transactions.

DURABILITY:

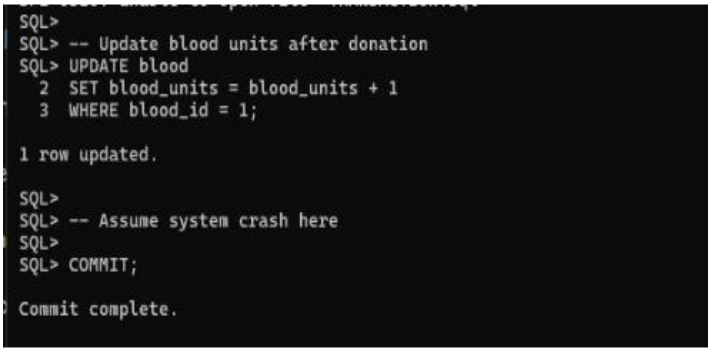
Scenario:

After a successful blood donation, the system crashes.

Solution:

Ensure that the changes made to the database are permanently saved, even if the system crashes.

Query:



```
SQL>
SQL> -- Update blood units after donation
SQL> UPDATE blood
  2  SET blood_units = blood_units + 1
  3  WHERE blood_id = 1;

1 row updated.

SQL>
SQL> -- Assume system crash here
SQL>
SQL> COMMIT;

Commit complete.
```

Explanation:

- START TRANSACTION; begins a new transaction.
- UPDATE blood... increases the available blood units.
- Even if the system crashes after the UPDATE statement, the changes made within the transaction will be durable. MySQL ensures durability by saving transaction logs and ensuring that committed transactions are permanently stored in the database.

For database tables:

‘Admin’ Table:

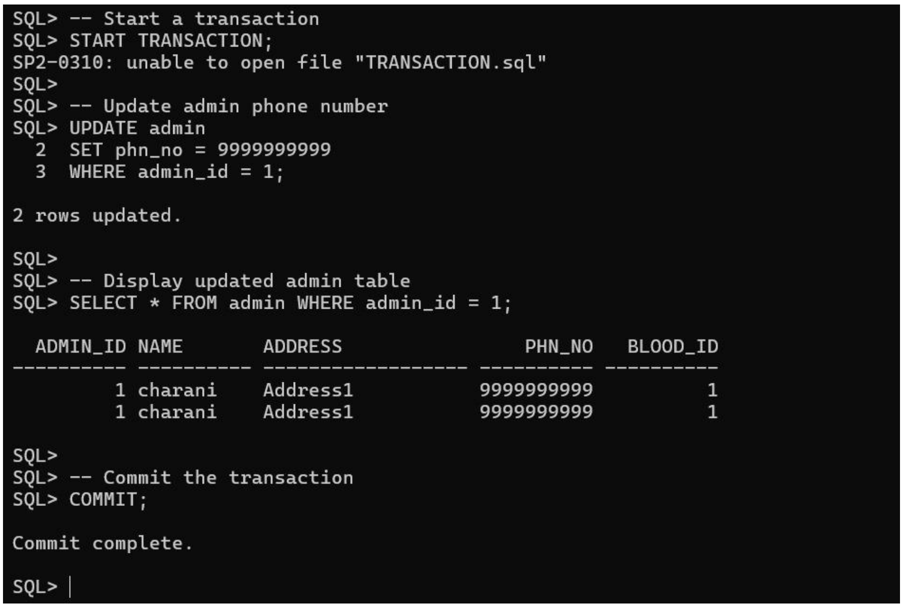
Scenario:

Concurrency Control and Transaction Control for the admin table

Solution:

To demonstrate concurrency control and transaction control for the admin table, we will perform a simple update operation within a transaction. This operation will update the phone number of an admin.

Query:



```
SQL> -- Start a transaction
SQL> START TRANSACTION;
SP2-0310: unable to open file "TRANSACTION.sql"
SQL>
SQL> -- Update admin phone number
SQL> UPDATE admin
  2  SET phn_no = 9999999999
  3  WHERE admin_id = 1;

2 rows updated.

SQL>
SQL> -- Display updated admin table
SQL> SELECT * FROM admin WHERE admin_id = 1;

  ADMIN_ID NAME       ADDRESS                PHN_NO   BLOOD_ID
---------- ---------- ------------------ ---------- ----------
         1 charani    Address1           9999999999          1
         1 charani    Address1           9999999999          1

SQL>
SQL> -- Commit the transaction
SQL> COMMIT;

Commit complete.

SQL> |
```

Explanation:

- We begin a transaction using START TRANSACTION.
- We then update the phone number of the admin with admin_id 1.
- The SELECT statement displays the updated admin table showing the changes.
- Finally, we commit the transaction using COMMIT

‘Blood’ Table:

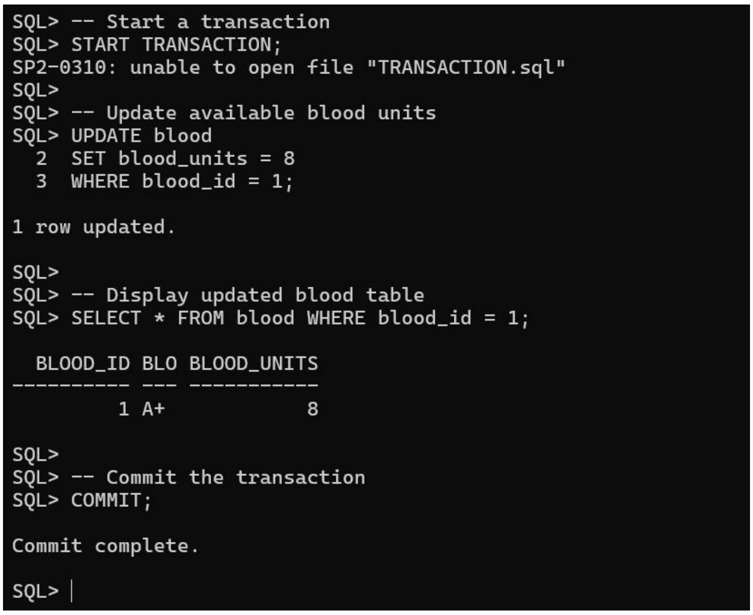
Scenario:

Concurrency Control and Transaction Control for the blood table

Solution:

To demonstrate concurrency control and transaction control for the blood table, we will perform a simple update operation within a transaction. This operation will update the available units of blood.

Query:



```
SQL> -- Start a transaction
SQL> START TRANSACTION;
SP2-0310: unable to open file "TRANSACTION.sql"
SQL>
SQL> -- Update available blood units
SQL> UPDATE blood
  2  SET blood_units = 8
  3  WHERE blood_id = 1;

1 row updated.

SQL>
SQL> -- Display updated blood table
SQL> SELECT * FROM blood WHERE blood_id = 1;

  BLOOD_ID BLO BLOOD_UNITS
---------- --- -----------
         1 A+            8

SQL>
SQL> -- Commit the transaction
SQL> COMMIT;

Commit complete.

SQL>
```

Explanation:

- We begin a transaction using START TRANSACTION.
- We then update the available units of blood for type 'A+'.
- The SELECT statement displays the updated blood table showing the changes.
- Finally, we commit the transaction using COMMIT.

‘Recipient’ Table:

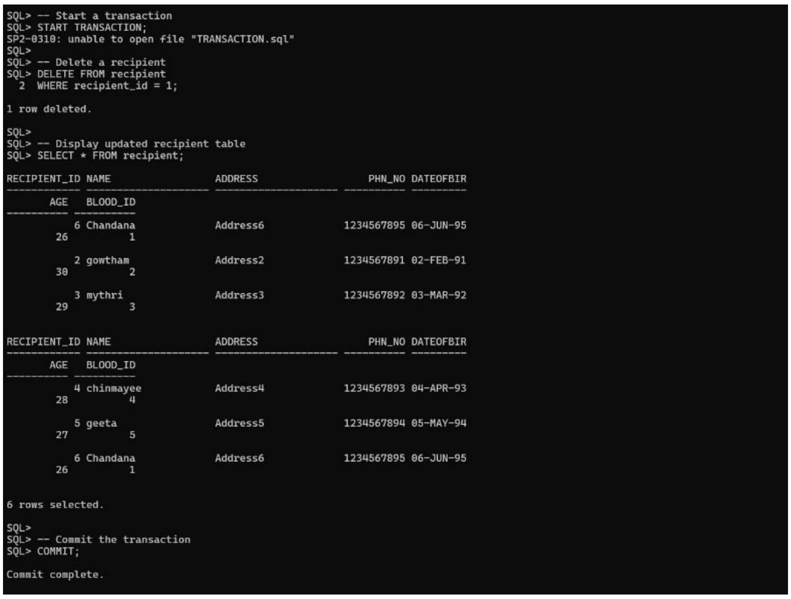
Scenario:

Concurrency Control and Transaction Control for the recipient table

Solution:

To demonstrate concurrency control and transaction control for the recipient table, we will perform a simple delete operation within a transaction. This operation will delete a recipient from the table.

Query:



```
SQL> -- Start a transaction
SQL> START TRANSACTION;
SP2-0310: unable to open file "TRANSACTION.sql"
SQL>
SQL> -- Delete a recipient
SQL> DELETE FROM recipient
  2  WHERE recipient_id = 1;

1 row deleted.

SQL>
SQL> -- Display updated recipient table
SQL> SELECT * FROM recipient;

RECIPIENT_ID NAME                 ADDRESS                  PHN_NO DATEOFBIR
------------ -------------------- -------------------- ---------- ---------
       AGE   BLOOD_ID
---------- ----------
           6 Chandana             Address6             1234567895 06-JUN-95
        26          1

           2 gowtham              Address2             1234567891 02-FEB-91
        30          2

           3 mythri               Address3             1234567892 03-MAR-92
        29          3


RECIPIENT_ID NAME                 ADDRESS                  PHN_NO DATEOFBIR
------------ -------------------- -------------------- ---------- ---------
       AGE   BLOOD_ID
---------- ----------
           4 chinmayee            Address4             1234567893 04-APR-93
        28          4

           5 geeta                Address5             1234567894 05-MAY-94
        27          5

           6 Chandana             Address6             1234567895 06-JUN-95
        26          1


6 rows selected.

SQL>
SQL> -- Commit the transaction
SQL> COMMIT;

Commit complete.
```

Explanation:

- We begin a transaction using START TRANSACTION.
- We then delete a recipient with recipient_id 1.
- The SELECT statement displays the updated recipient table showing the changes.
- Finally, we commit the transaction using COMMIT

‘Location’ Table:

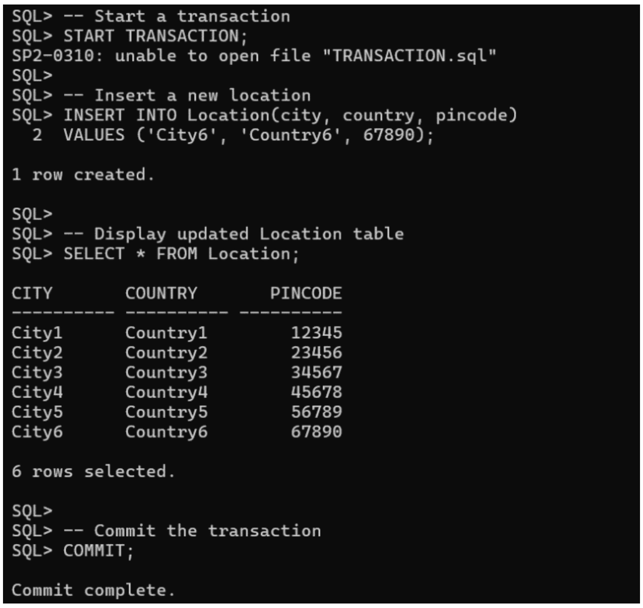
Scenario:

Concurrency Control and Transaction Control for the Location table

Solution:

To demonstrate concurrency control and transaction control for the Location table, we will perform a simple insert operation within a transaction. This operation will insert a new location into the table.

Query:



```
SQL> -- Start a transaction
SQL> START TRANSACTION;
SP2-0310: unable to open file "TRANSACTION.sql"
SQL>
SQL> -- Insert a new location
SQL> INSERT INTO Location(city, country, pincode)
  2  VALUES ('City6', 'Country6', 67890);

1 row created.

SQL>
SQL> -- Display updated Location table
SQL> SELECT * FROM Location;

CITY       COUNTRY       PINCODE
---------- ---------- ----------
City1      Country1        12345
City2      Country2        23456
City3      Country3        34567
City4      Country4        45678
City5      Country5        56789
City6      Country6        67890

6 rows selected.

SQL>
SQL> -- Commit the transaction
SQL> COMMIT;

Commit complete.
```

Explanation:

- We begin a transaction using START TRANSACTION.
- We then insert a new location into the Location table.
- The SELECT statement displays the updated Location table showing the changes.
- Finally, we commit the transaction using COMMIT.

‘Donor’ Table:

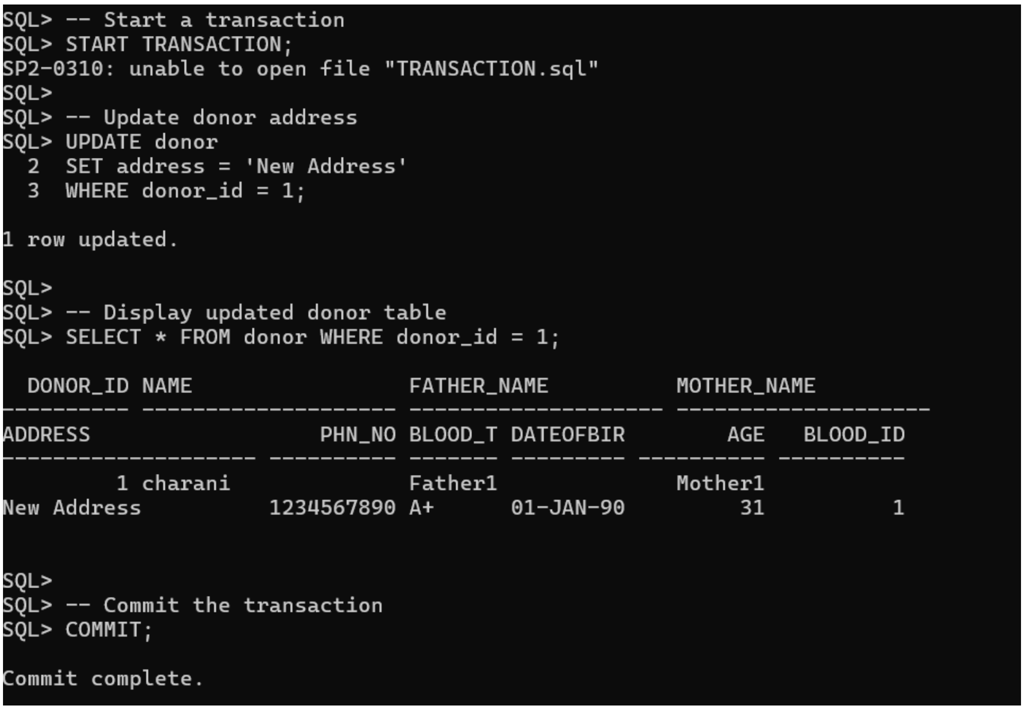
Scenario:

Concurrency Control and Transaction Control for the donor table

Solution:

To demonstrate concurrency control and transaction control for the donor table, we will perform a simple update operation within a transaction. This operation will update the address of a donor**.**

Query:



```
SQL> -- Start a transaction
SQL> START TRANSACTION;
SP2-0310: unable to open file "TRANSACTION.sql"
SQL>
SQL> -- Update donor address
SQL> UPDATE donor
  2  SET address = 'New Address'
  3  WHERE donor_id = 1;

1 row updated.

SQL>
SQL> -- Display updated donor table
SQL> SELECT * FROM donor WHERE donor_id = 1;

  DONOR_ID NAME                 FATHER_NAME          MOTHER_NAME
---------- -------------------- -------------------- --------------------
ADDRESS                  PHN_NO BLOOD_T DATEOFBIR        AGE   BLOOD_ID
-------------------- ---------- ------- --------- ---------- ----------
         1 charani              Father1              Mother1
New Address          1234567890 A+      01-JAN-90         31          1


SQL>
SQL> -- Commit the transaction
SQL> COMMIT;

Commit complete.
```

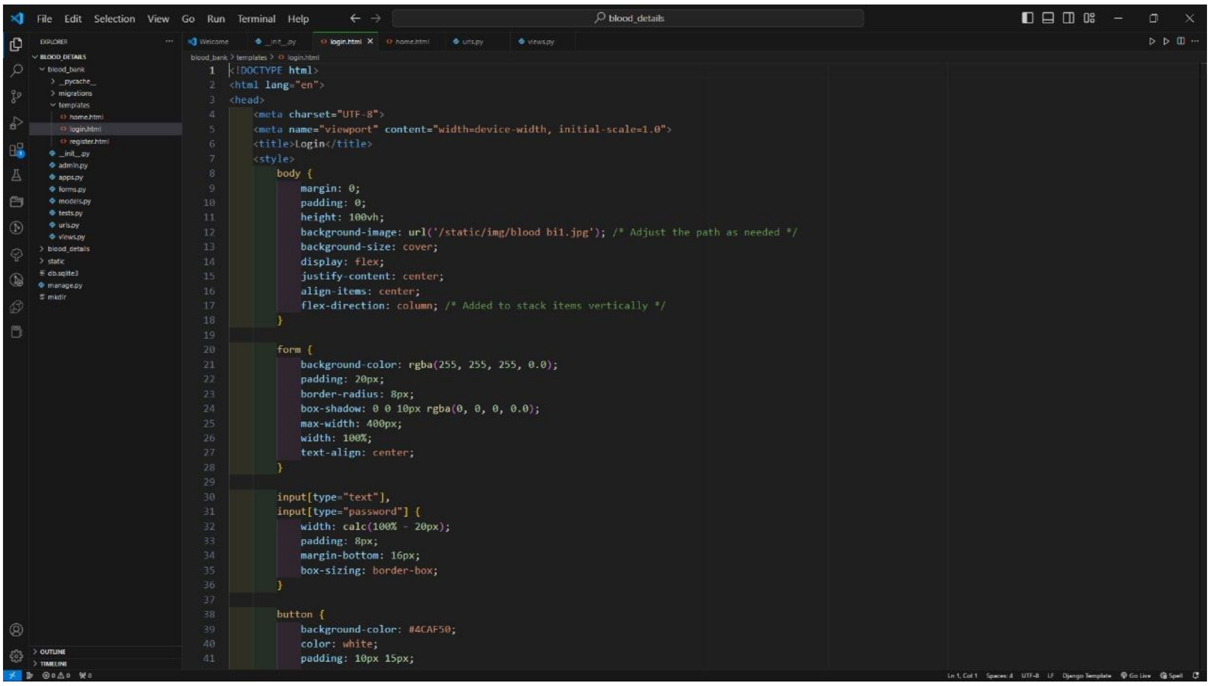
Explanation:

- We begin a transaction using START TRANSACTION.
- We then update the address of the donor with donor_id 1.
- The SELECT statement displays the updated donor table showing the changes.
- Finally, we commit the transaction using COMMIT.

CHAPTER 6

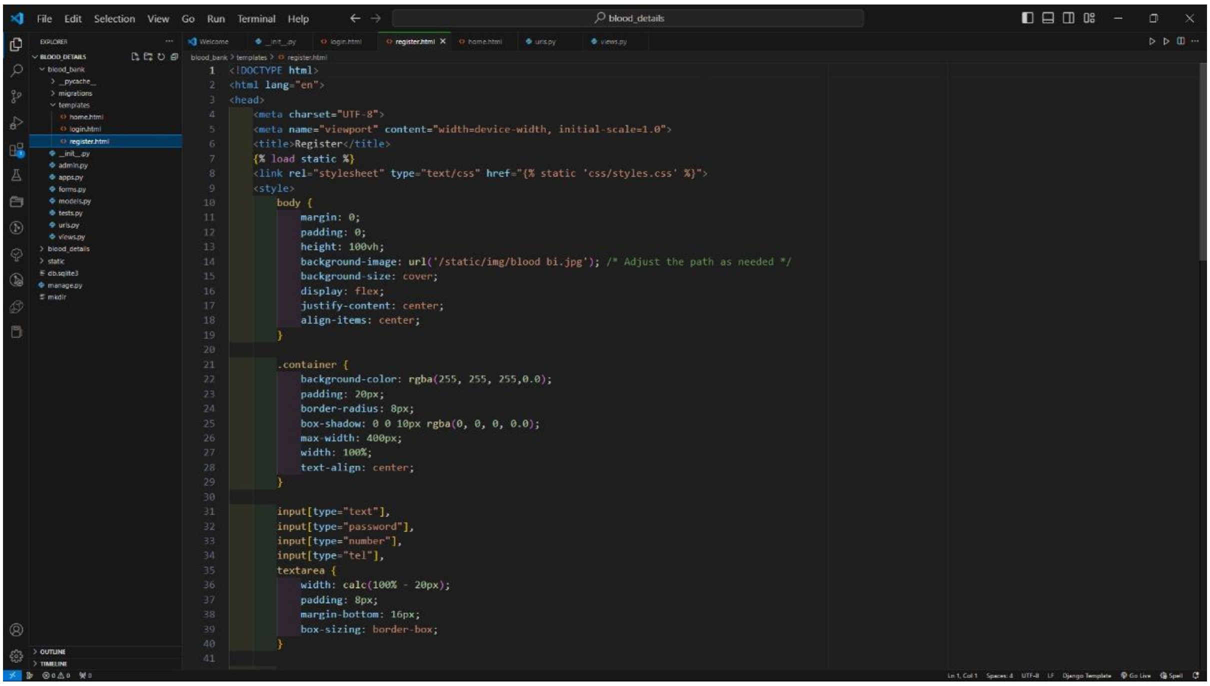
CODE SNIPPETS

Login.Html



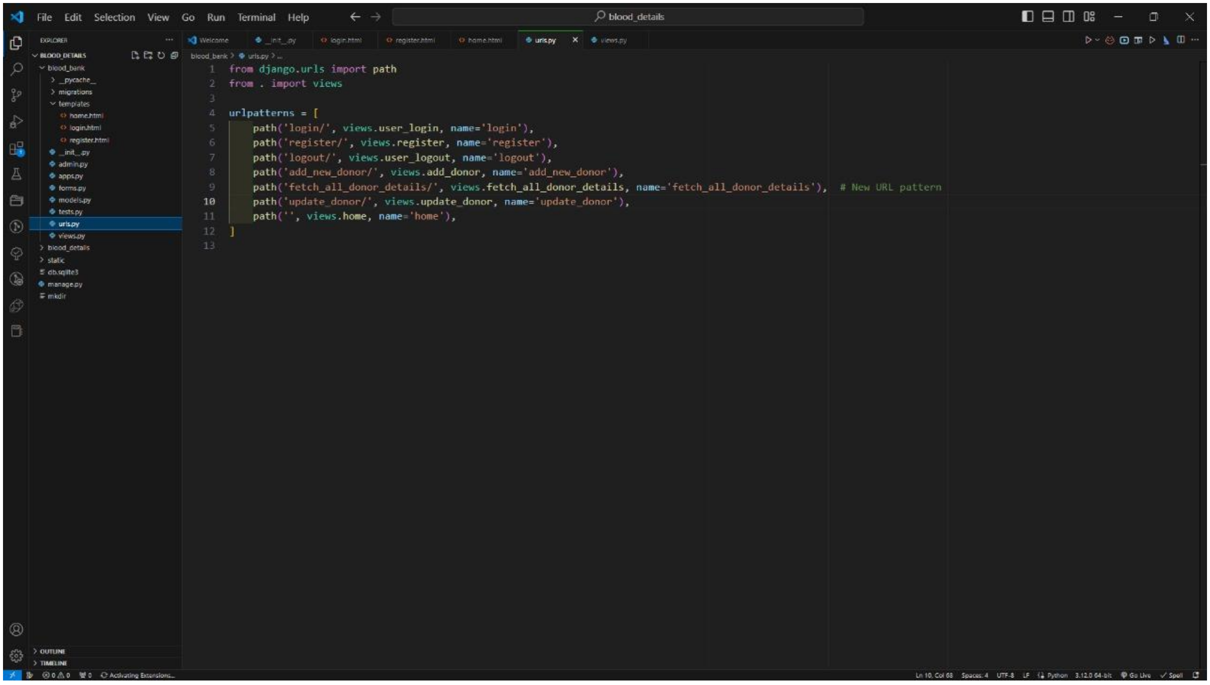
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <style>
        body {
            margin: 0;
            padding: 0;
            height: 100vh;
            background-image: url('/static/img/blood bi1.jpg'); /* Adjust the path as needed */
            background-size: cover;
            display: flex;
            justify-content: center;
            align-items: center;
            flex-direction: column; /* Added to stack items vertically */
        }

        form {
            background-color: rgba(255, 255, 255, 0.0);
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.0);
            max-width: 400px;
            width: 100%;
            text-align: center;
        }

        input[type="text"],
        input[type="password"] {
            width: calc(100% - 20px);
            padding: 8px;
            margin-bottom: 16px;
            box-sizing: border-box;
        }

        button {
            background-color: #4CAF50;
            color: white;
            padding: 10px 15px;
```

Register.Html



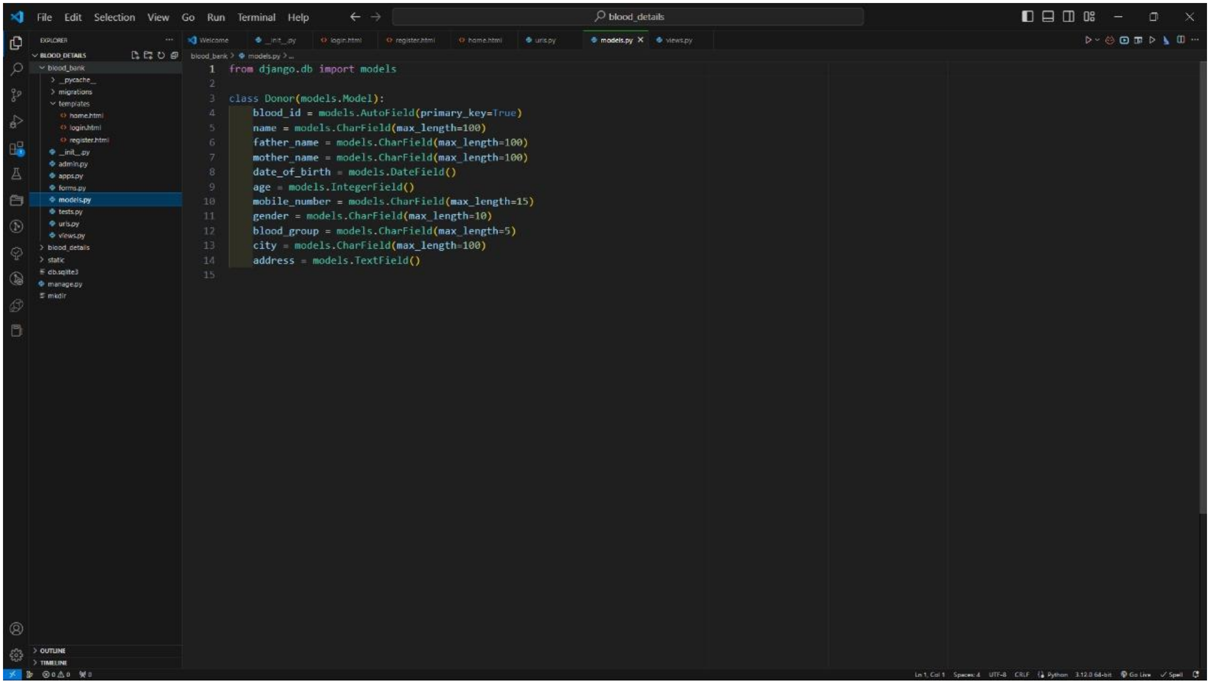
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
    {% load static %}
    <link rel="stylesheet" type="text/css" href="{% static 'css/styles.css' %}">
    <style>
        body {
            margin: 0;
            padding: 0;
            height: 100vh;
            background-image: url('/static/img/blood bi.jpg'); /* Adjust the path as needed */
            background-size: cover;
            display: flex;
            justify-content: center;
            align-items: center;
        }

        .container {
            background-color: rgba(255, 255, 255,0.0);
            padding: 20px;
            border-radius: 8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.0);
            max-width: 400px;
            width: 100%;
            text-align: center;
        }

        input[type="text"],
        input[type="password"],
        input[type="number"],
        input[type="tel"],
        textarea {
            width: calc(100% - 20px);
            padding: 8px;
            margin-bottom: 16px;
            box-sizing: border-box;
        }

```

Urls.py



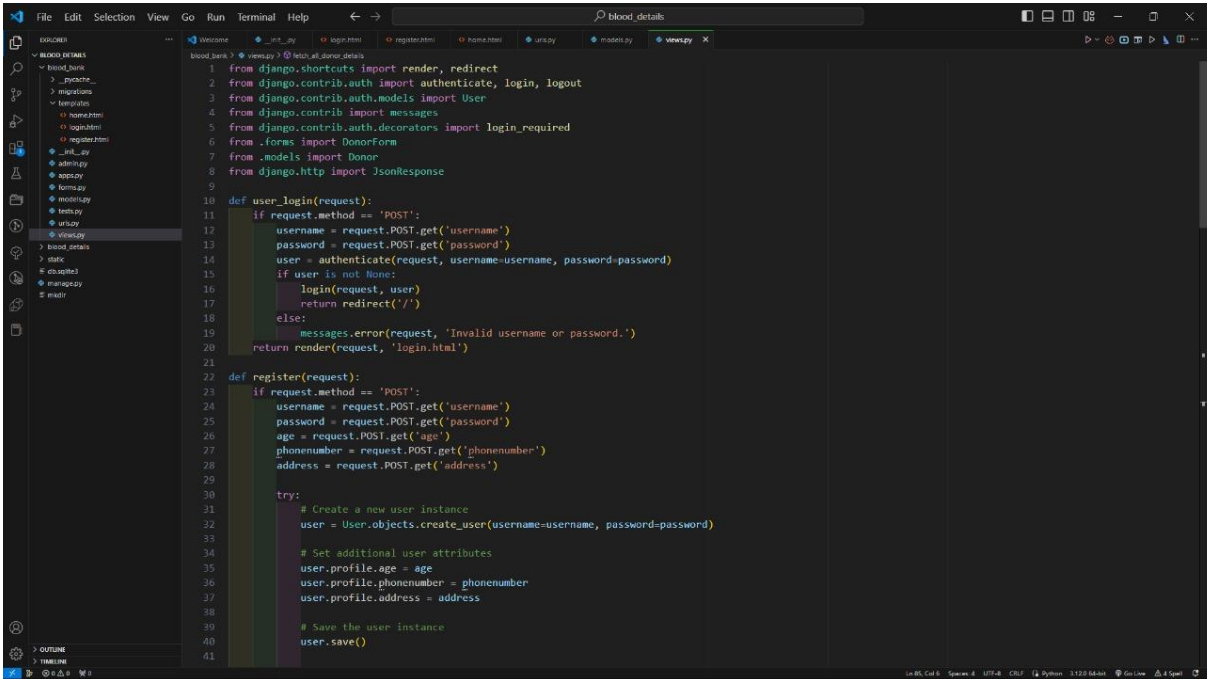
```python
from django.urls import path
from . import views

urlpatterns = [
    path('login/', views.user_login, name='login'),
    path('register/', views.register, name='register'),
    path('logout/', views.user_logout, name='logout'),
    path('add_new_donor/', views.add_donor, name='add_new_donor'),
    path('fetch_all_donor_details/', views.fetch_all_donor_details, name='fetch_all_donor_details'),  # New URL pattern
    path('update_donor/', views.update_donor, name='update_donor'),
    path('', views.home, name='home'),
]
```

Models.py



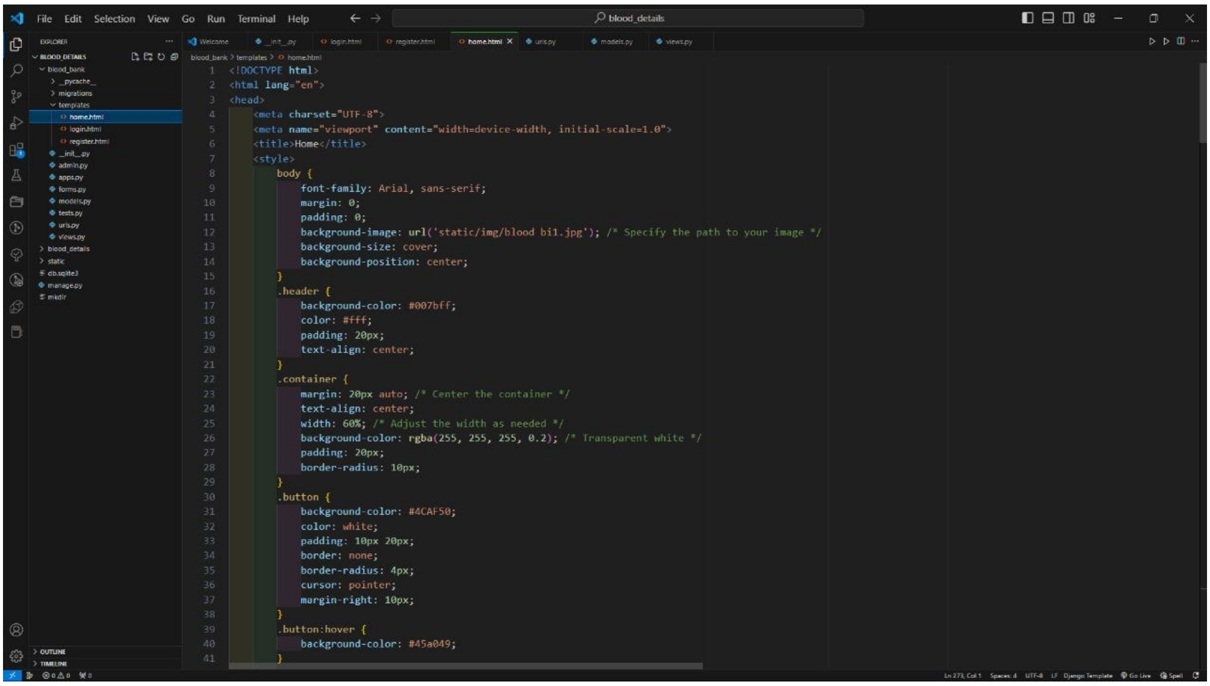
```python
from django.db import models

class Donor(models.Model):
    blood_id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=100)
    father_name = models.CharField(max_length=100)
    mother_name = models.CharField(max_length=100)
    date_of_birth = models.DateField()
    age = models.IntegerField()
    mobile_number = models.CharField(max_length=15)
    gender = models.CharField(max_length=10)
    blood_group = models.CharField(max_length=5)
    city = models.CharField(max_length=100)
    address = models.TextField()
```

Views.py



```python
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.models import User
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from .forms import DonorForm
from .models import Donor
from django.http import JsonResponse

def user_login(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('/')
        else:
            messages.error(request, 'Invalid username or password.')
    return render(request, 'login.html')

def register(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        age = request.POST.get('age')
        phonenumber = request.POST.get('phonenumber')
        address = request.POST.get('address')

        try:
            # Create a new user instance
            user = User.objects.create_user(username=username, password=password)

            # Set additional user attributes
            user.profile.age = age
            user.profile.phonenumber = phonenumber
            user.profile.address = address

            # Save the user instance
            user.save()

```

Home.html

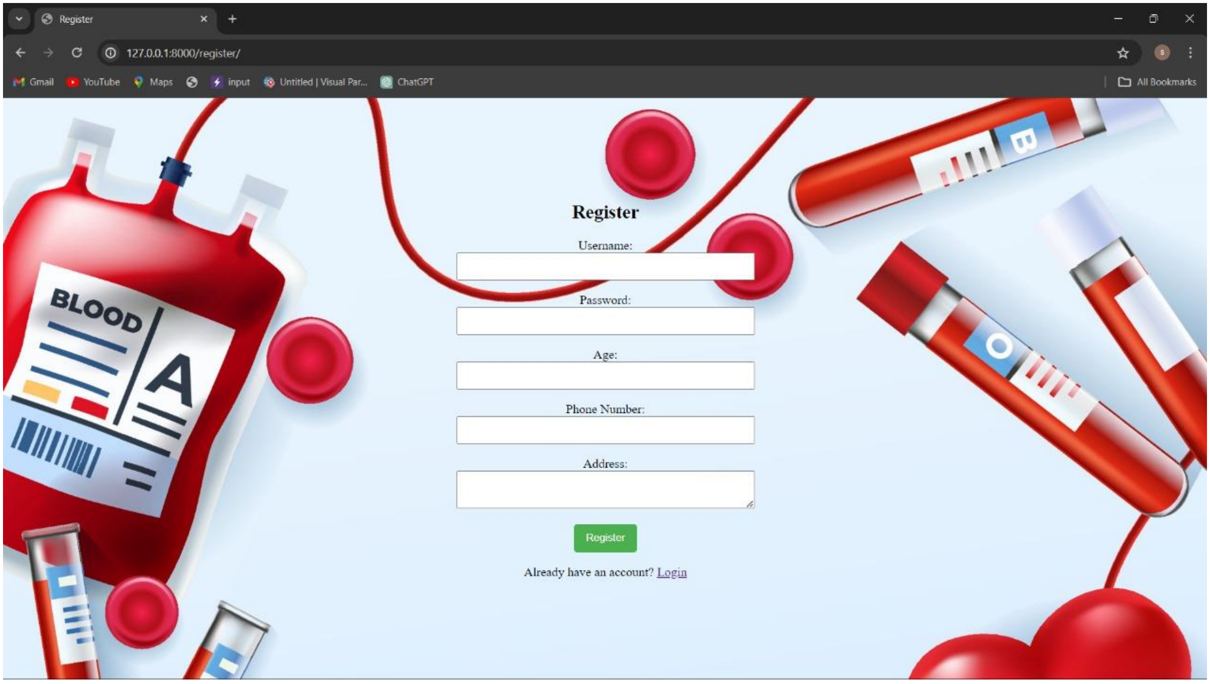


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-image: url('static/img/blood bi1.jpg'); /* Specify the path to your image */
            background-size: cover;
            background-position: center;
        }
        .header {
            background-color: #007bff;
            color: #fff;
            padding: 20px;
            text-align: center;
        }
        .container {
            margin: 20px auto; /* Center the container */
            text-align: center;
            width: 60%; /* Adjust the width as needed */
            background-color: rgba(255, 255, 255, 0.2); /* Transparent white */
            padding: 20px;
            border-radius: 10px;
        }
        .button {
            background-color: #4CAF50;
            color: white;
            padding: 10px 20px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            margin-right: 10px;
        }
        .button:hover {
            background-color: #45a049;
        }
```

CHAPTER 7

RESULTS AND DISCUSSION

Register Page



Register

Username:

Password:

Age:

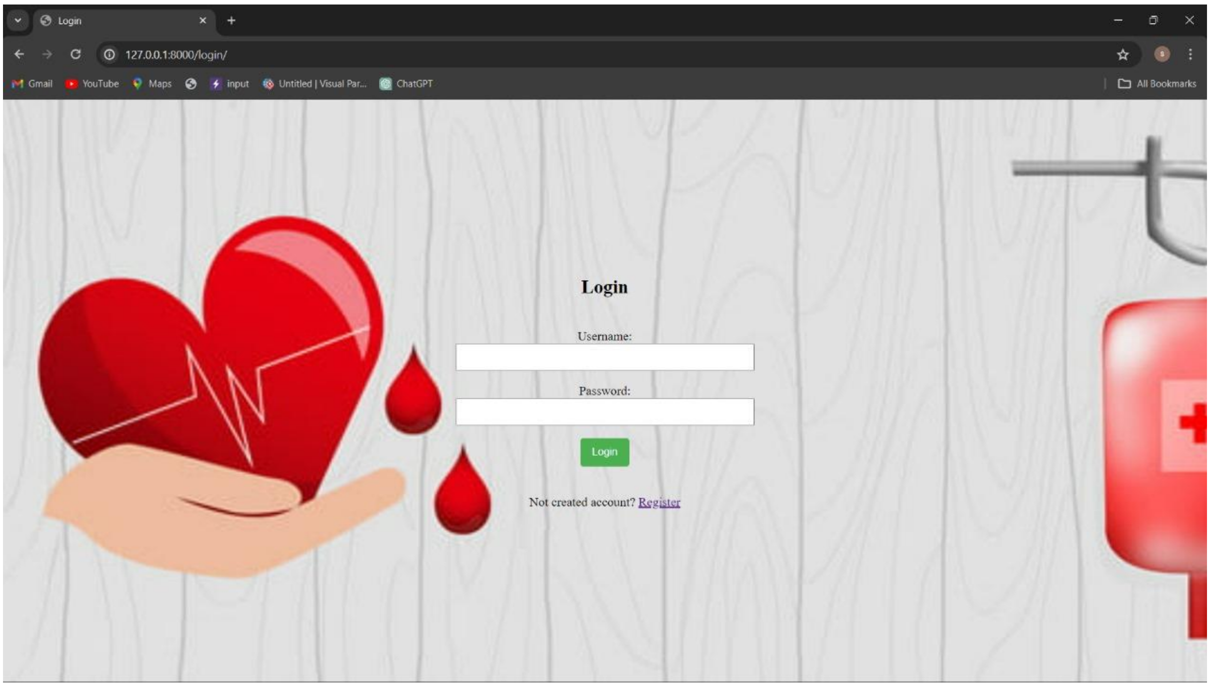
Phone Number:

Address:

Register

Already have an account? Login

Login Page



Login

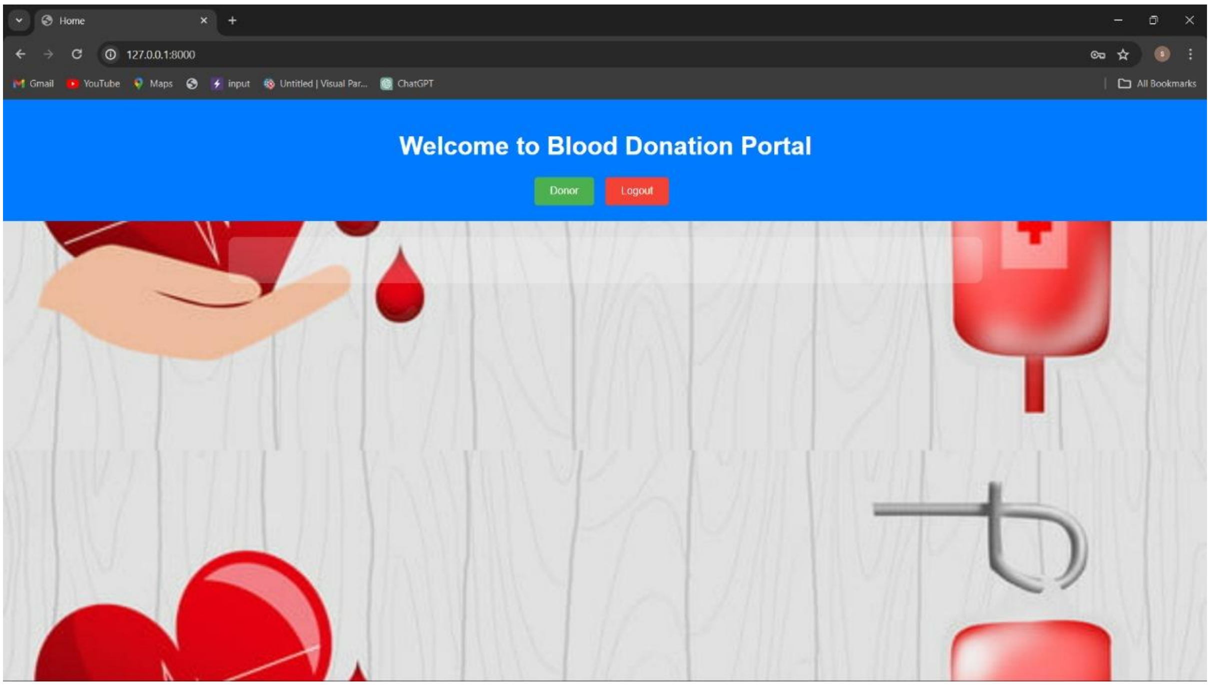
Username:

Password:

Login

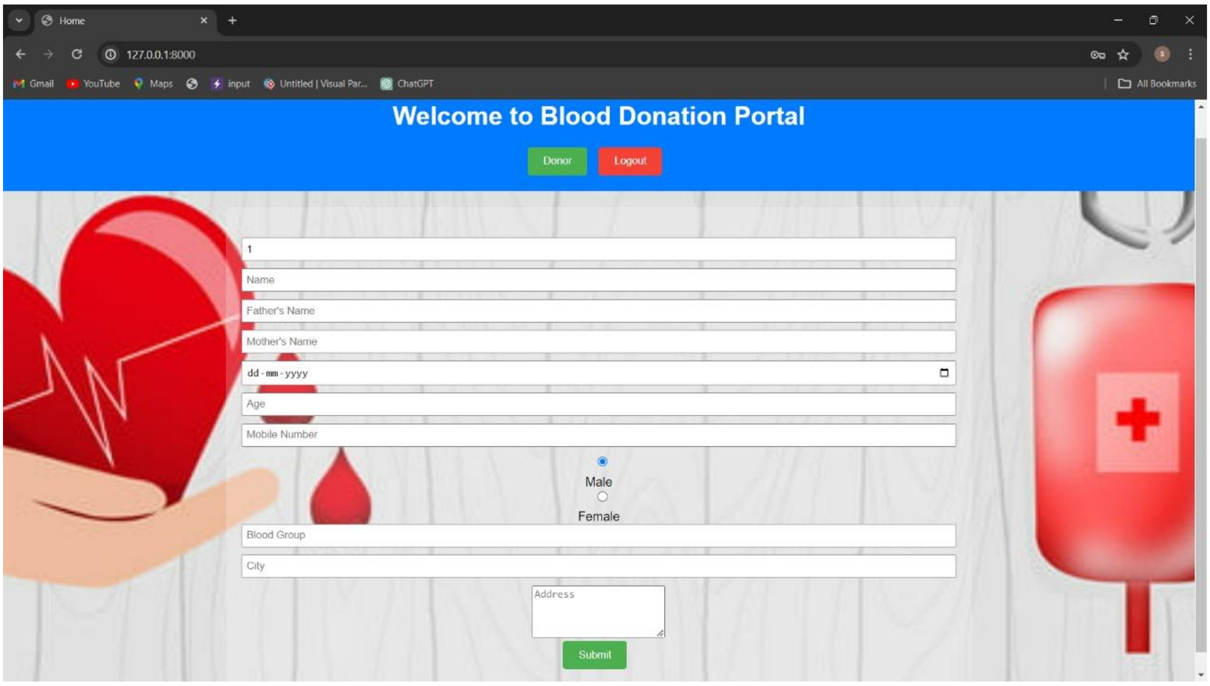
Not created account? Register

Home Page



Welcome to Blood Donation Portal

New Donor Details



Welcome to Blood Donation Portal

1

Name

Father's Name

Mother's Name

dd-mm-yyyy

Age

Mobile Number

● Male

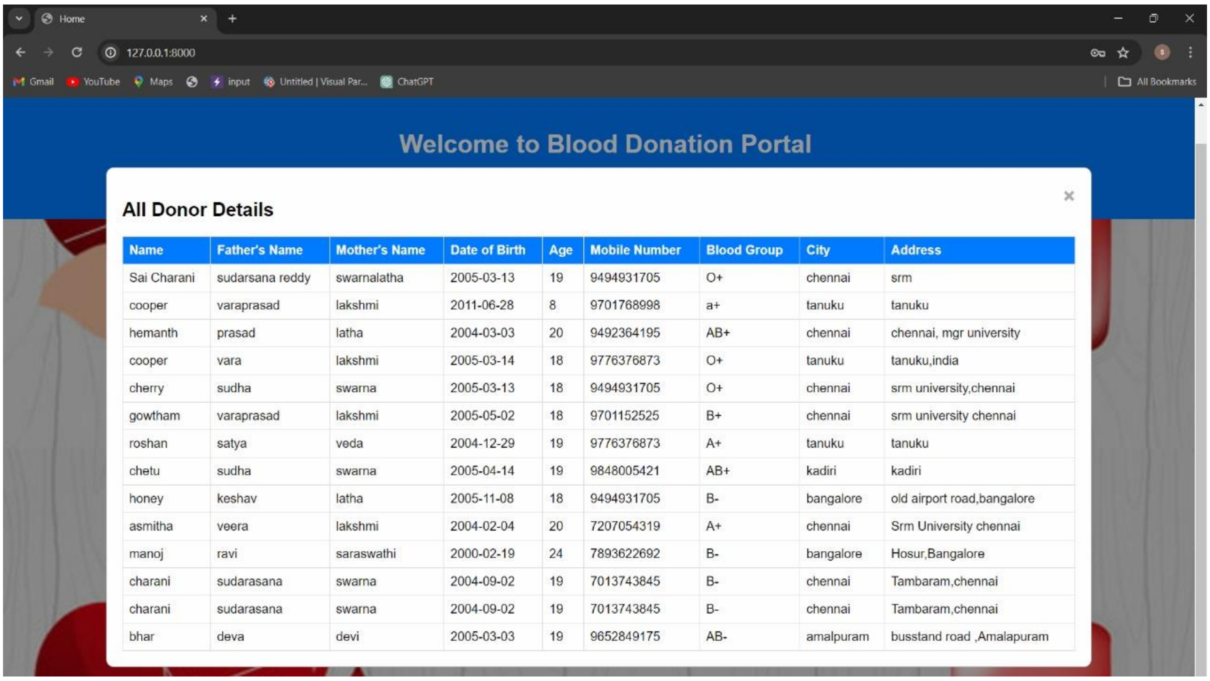
○ Female

Blood Group

City

Address

Add New Donor Details

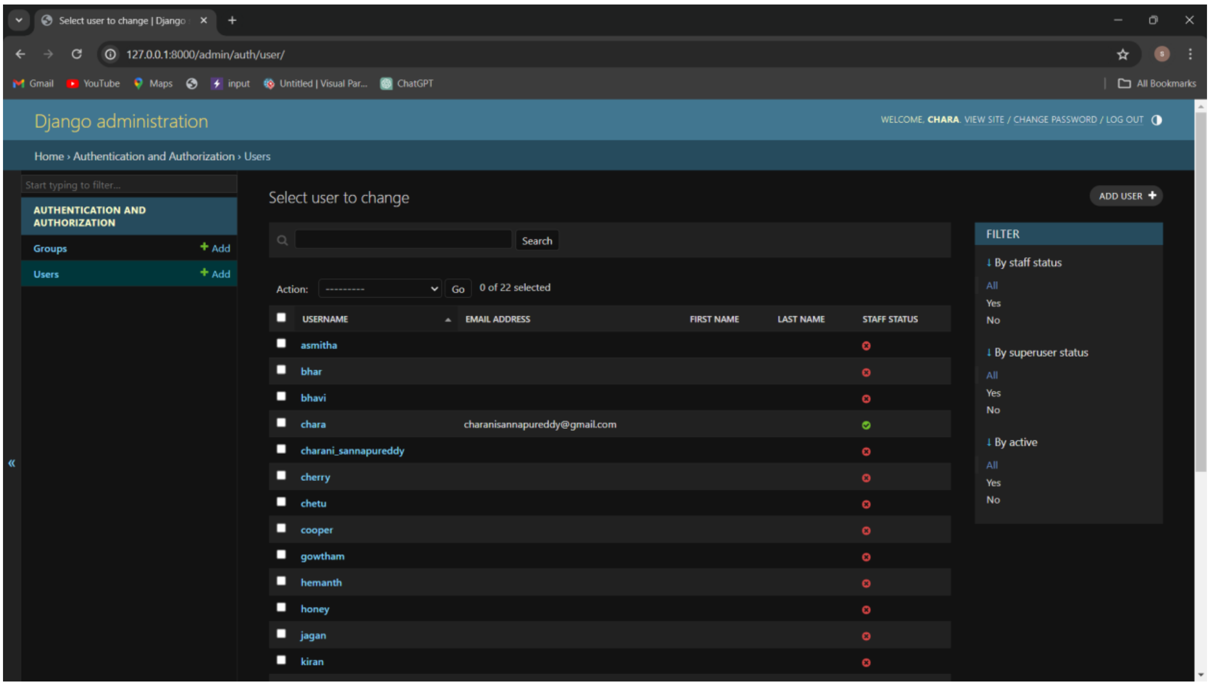


Welcome to Blood Donation Portal

All Donor Details

Name	Father's Name	Mother's Name	Date of Birth	Age	Mobile Number	Blood Group	City	Address
Sai Charani	sudarsana reddy	swarnalatha	2005-03-13	19	9494931705	O+	chennai	srm
cooper	varaprasad	lakshmi	2011-06-28	8	9701768998	a+	tanuku	tanuku
hemanth	prasad	latha	2004-03-03	20	9492364195	AB+	chennai	chennai, mgr university
cooper	vara	lakshmi	2005-03-14	18	9776376873	O+	tanuku	tanuku,india
cherry	sudha	swarna	2005-03-13	18	9494931705	O+	chennai	srm university,chennai
gowtham	varaprasad	lakshmi	2005-05-02	18	9701152525	B+	chennai	srm university chennai
roshan	satya	veda	2004-12-29	19	9776376873	A+	tanuku	tanuku
chetu	sudha	swarna	2005-04-14	19	9848005421	AB+	kadiri	kadiri
honey	keshav	latha	2005-11-08	18	9494931705	B-	bangalore	old airport road,bangalore
asmitha	veera	lakshmi	2004-02-04	20	7207054319	A+	chennai	Srm University chennai
manoj	ravi	saraswathi	2000-02-19	24	7893622692	B-	bangalore	Hosur,Bangalore
charani	sudarasana	swarna	2004-09-02	19	7013743845	B-	chennai	Tambaram,chennai
charani	sudarasana	swarna	2004-09-02	19	7013743845	B-	chennai	Tambaram,chennai
bhar	deva	devi	2005-03-03	19	9652849175	AB-	amalpuram	busstand road ,Amalapuram

Admin Page



Select user to change

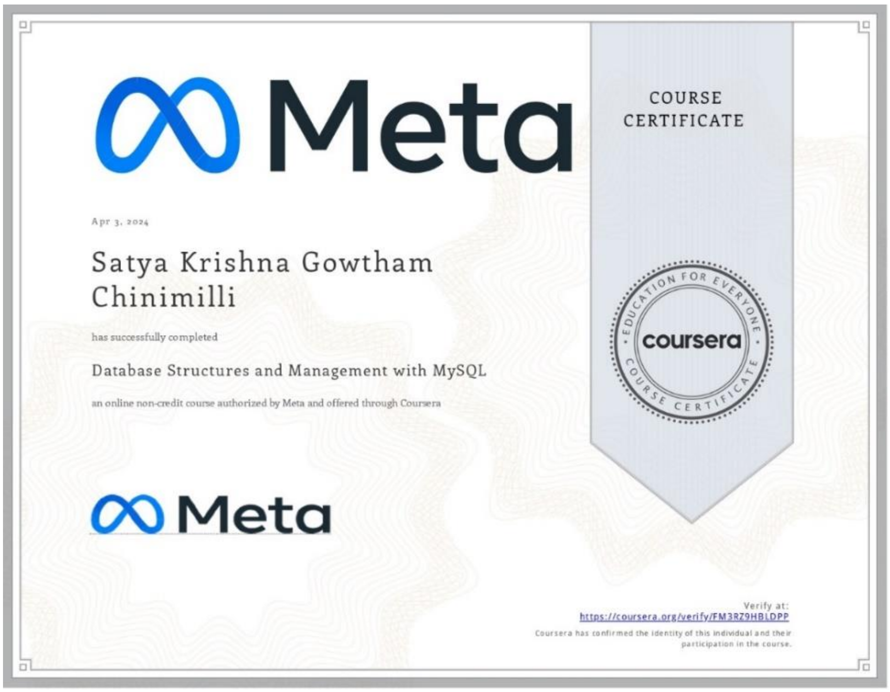
0 of 22 selected

☐	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
☐	asmitha				✗
☐	bhar				✗
☐	bhavi				✗
☐	chara	charanisannapureddy@gmail.com			✓
☐	charani_sannapureddy				✗
☐	cherry				✗
☐	chetu				✗
☐	cooper				✗
☐	gowtham				✗
☐	hemanth				✗
☐	honey				✗
☐	jagan				✗
☐	kiran				✗

CHAPTER 8

ONLINE CERTIFICATE

CH.S.K.Gowtham(RA2211027010149)



Meta

COURSE CERTIFICATE

Apr 3, 2024

Satya Krishna Gowtham Chinimilli

has successfully completed

Database Structures and Management with MySQL

an online non-credit course authorized by Meta and offered through Coursera

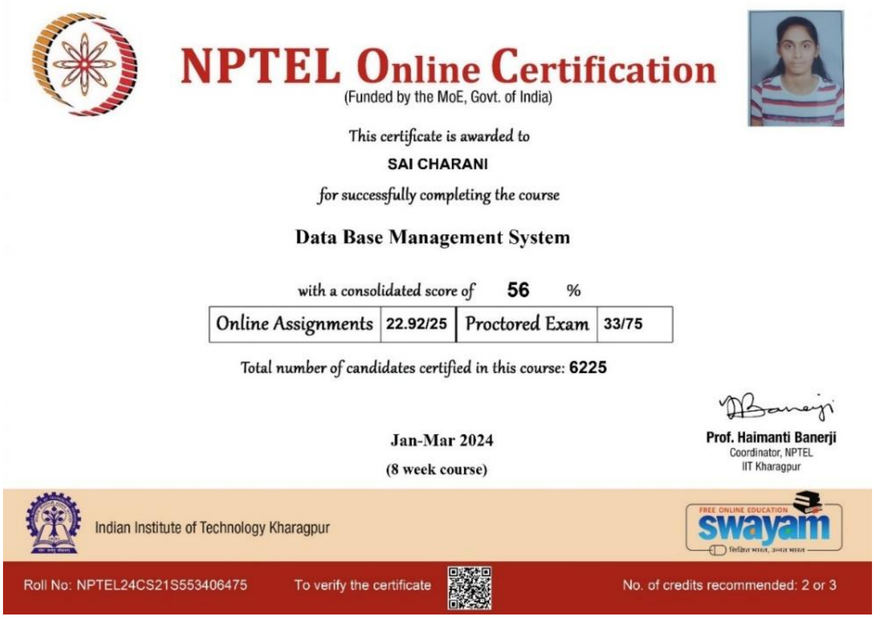
EDUCATION FOR EVERYONE · coursera · COURSE CERTIFICATE

Meta

Verify at:
https://coursera.org/verify/FM3RZ9HBLDPP

Coursera has confirmed the identity of this individual and their participation in the course.

S.Sai Charani(RA2211027010186)



NPTEL Online Certification

(Funded by the MoE, Govt. of India)

This certificate is awarded to

SAI CHARANI

for successfully completing the course

Data Base Management System

with a consolidated score of 56 %

Online Assignments	22.92/25	Proctored Exam	33/75

Total number of candidates certified in this course: 6225

Jan-Mar 2024

(8 week course)

Prof. Haimanti Banerji
Coordinator, NPTEL
IIT Kharagpur

Indian Institute of Technology Kharagpur

FREE ONLINE EDUCATION swayam

Roll No: NPTEL24CS21S553406475   To verify the certificate   No. of credits recommended: 2 or 3