ORGAN DONATION AND PROCUREMENT MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

Mudiyala N S Charishma Reddy [RA2211003011008]

Madhu Patil [RA2211003011018]

Under the Guidance of

Dr. M. Kandan

Assistant Professor, Department of Computing Technologies

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTING TECHNOLOGIES
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR– 603 203

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR–603 203

BONAFIDE CERTIFICATE

Register no. RA2211003011008, RA2211003011018 Certified to be the bonafide work done by **Mudiyala N S Charishma Reddy, Madhu Patil** of II year/IV sem B.Tech Degree Course in the Project Course – **21CSC205P Database Management Systems** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**, Kattankulathur for the academic year 2023-2024.

Date: 29/04/2024

Faculty in Charge
Dr. M. KANDAN
Assistant Professor
Department of Computing Technologies
SRMSIT -KTR

Department of Computing Technologies Kattankulathur 603 203 SRM IST

HEAD OF THE DEPARTMENT
Dr. M. PUSHPALATHA
Professor
Department of Computing Technologies
SRMIST - KTR

ABSTRACT

Organ transplantation on is a critical medical technique for treating organ failure, necessitating effective organ procurement and allocation systems. Organ Dona on and Procurement Organisations (ODPOs) are at the forefront of this process, requiring strong management techniques to maximise organ utilisation and reduce waste. In response, we created an Organ Dona on and Procurement Network Management System using MySQL database technology. This system has complete data management capabilities that include personal information, medical histories, and organ availability statistics. The system, which is built on MySQL, provides powerful data storage, retrieval, and analysis capabilities, assuring regulatory compliance and increasing transparency in organ donation operations. By expediting organ procurement and distribution, this study improves the effectiveness and equality of organ transplantation networks while also providing policymakers and healthcare stakeholders with useful insights.

TABLE OF CONTENTS

ABSTRACT **iii**

Problem Statement **1**

Chapter No	Chapter Name	Page No
1.	Problem understanding, Identification of Entity and Relationships, Construction of DB using ER Model for the project	**2**
2.	Design of Relational Schemas, Creation of Database Tables for the project.	**8**
3.	Complex queries based on the concepts of constraints, sets, joins, views, Triggers and Cursors.	**27**
4.	Analyzing the pitfalls, identifying the dependencies, and applying normalizations	**46**
5.	Implementation of concurrency control and recovery mechanisms	**53**
6.	Code for the project	**58**
7.	Result and Discussion (Screen shots of the implementation with front end.	**97**
8.	Attach the Real Time project certificate / Online course certificate	**100**

Chapter-1

1. Problem Statement

Organ transplantation is a medical procedure in which an organ is removed from one body and placed in the body of a recipient, to replace a damaged or missing organ. The donor and recipient may be at the same location, or organs may be transported from a donor site to another location. Organ Donation and Procurement Organizations play a pivotal role in today's medical institutions. Such organizations are responsible for the evaluation and procurement of organs for organ transplantation. These organizations represent the front-line of organ procurement, having direct contact with the hospital and the family of a recently deceased donor. The work of such organizations includes to identify the best candidates for the available organs and to coordinate with the medical institutions to decide on each organ recipient. They are also responsible for educating the public to increase the awareness of and participation in the organ donation process. Also, it keeps track of all transplantation operations carried till date.

The Organ Donation and Procurement Network Management System is a database management system that uses database technology to construct, maintain and manipulate various kinds of data about a person's donation or procurement of a particular organ. It maintains a comprehensive medical history and other critical information like blood group, age, etc of every person in the database design. In short, it maintains a database containing statistical information regarding network of organ donation and procurement of different countries

1.1 Existing System

The prevailing manual and paper-based systems for organ donation and procurement exhibit several notable drawbacks. These include the labor-intensive nature of paperwork, which can lead to errors and time-consuming processes, potentially causing delays in critical transplantation procedures. Limited accessibility to vital information

across different locations or institutions can impede the swift coordination required in organ transplantation. Furthermore, the risk of data redundancy and inconsistency is higher in manual systems, and the lack of real-time updates hinders the immediate availability of crucial information. Tracking transplantation operations becomes challenging, with the potential for oversights or errors. Ineffective communication between organ donation organizations and medical institutions, coupled with a deficiency in centralized systems, adds to the inefficiencies.

The existing systems also struggle to adequately support public awareness initiatives, and their insufficient tools for data analysis hinder the identification of patterns and the improvement of overall processes. Moreover, concerns regarding security underscore the urgency of transitioning to a more sophisticated Organ Donation and Procurement Network Management System to address these multifaceted limitations and improve the efficacy of organ transplantation procedures.

1.2 Objective of the Project

Organ Wastage is a major issue that can only be solved by having a proper database of all Patient and Donors in a well-formed way, that can be processed easily. Records of donor and patients are created when a person donates or procures an organ from a Medical Institution. Records may include the following information:

1. Personal Information
2. Medical History
3. Medical insuarance, if any
4. Allergies to any medicine, if any
5. The need for an organ presently
6. Medical Insurance provided by any private or government insurers.
7. Address

This record serves a variety of purposes and is critical to the proper functioning of

Organ Donation and Procurement Network, especially in today's complicated health care environment. These records provide statistical information regarding the number of organs needed and available at a particular point of time. It is essential for planning, evaluating and coordinating organ donation and procurement. In India, the Transplantation of organs is done according to the Transplantation of Human Organs (THO) Act,1994. Many new rules had been added to the act, later on, to cater to current needs. According to this Act, every transplantation operation should be approved by the Government Organization. So the records of transplantation are there with the organization. Also, these operations can only be done in Government-authorized Hospitals. Our aim to create a solution that effectively deals with the problems of finding donors and also providing Statistical data of the transplants that can help the government to form better rules and regulations.

2 ER Diagram

2.1 Entity and Their Attributes:

User Entity:

It has User ID, Name, Date of birth, Phone Number (multi-valued), Medical Insurance, Medical History and Address. User ID is the Primary key.

Patient Entity:

It has Patient_ID, Organ Required, Reason of procurement and User_ID. Patient ID is the Primary Key.

Donor Entity:

It has Donor_ID , Organ Donated, Reason of donation and User_ID. Donor ID is the Primary Key.

Organ Available Entity:

It has Organ_ID, Organ Name and Donor_ID.. Organ ID is the Primary Key.

Organization Entity:

It has Organization ID, Organization Name, Location, Government approved organization or not and Phone Number (multi-valued). Organization ID is the Primary Key.

Doctor Entity:

It has Doctor ID, Doctor Name and Phone Number (multi-valued). Doctor ID is the Primary Key.

Organization Head Entity:

It has Head Name, Date of Joining and Term Length.

2.2 Relationships between Entities:

Donates:

Donates relation is creating relationship between User entity and Donor entity. It has M to M relationship.

Procures:

Procures relation is creating relationship between Patient entity and User entity. It has M to M relationship.

Transaction:

Transaction relation is creating relationship between Patient entity and Organ Available entity. It has 1 to 1 relationship.

Organ Donated:

Organ Donated relation is creating relationship between Organ Available entity and Donor entity. It has M to 1 relationship.

Attended by:

Attended by relation is creating relationship between Patient entity and Doctor entity. It has M to M relationship.

Registers:

Registers relation is creating relationship between Donor entity and Organization entity. It has M to 1 relationship.

Works in:

Works in relation is creating relationship between Doctor entity and Organization entity. It has M to 1 relationship.

Headed By:

Headed by relation is creating relationship between Organization entity and Organization Head entity. It has 1 to 1 relationship.

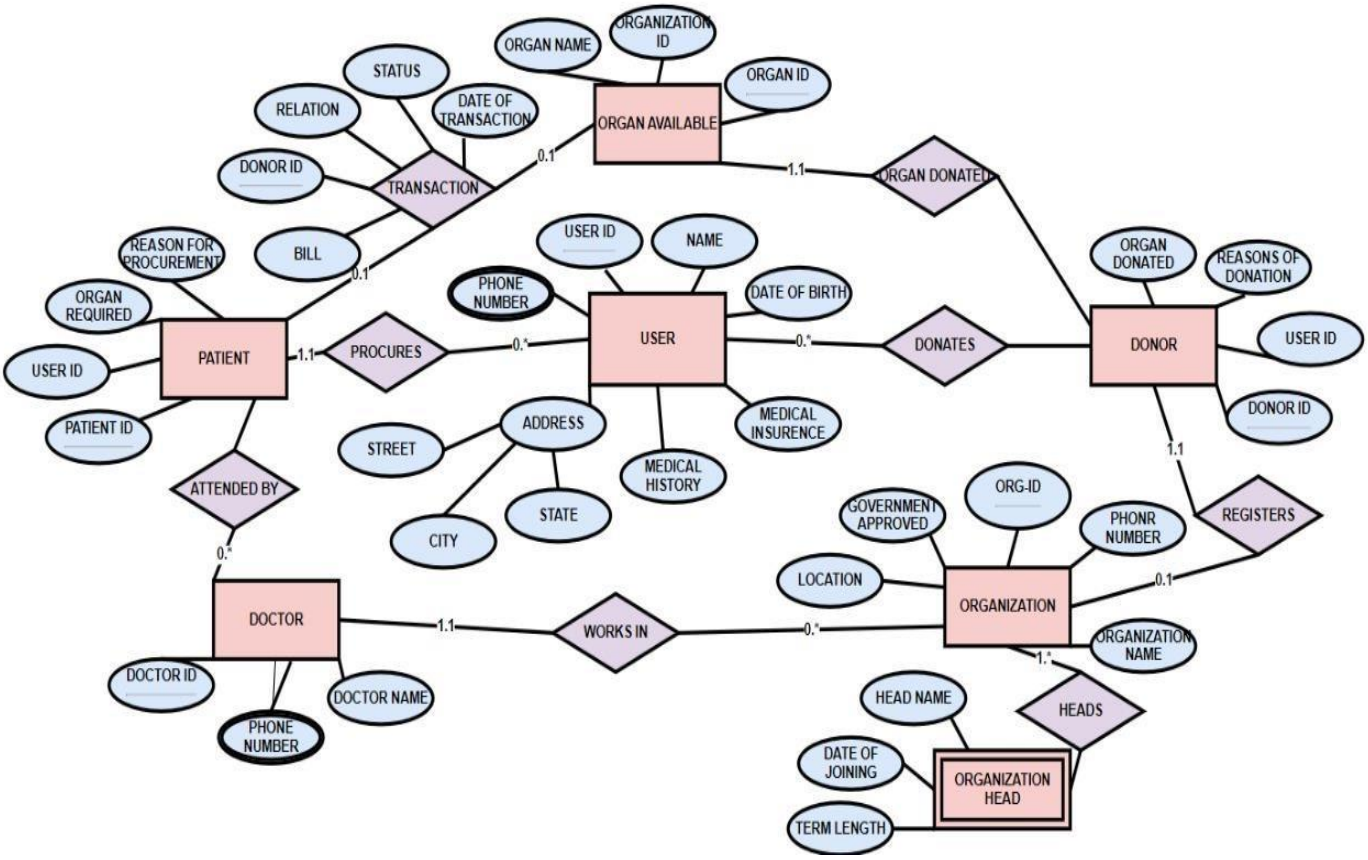


Figure 2.1 Entity Relationship Diagram for Online Procurement Management system

Chapter-2

3. Relational Tables and Schema

3.1 Schema Diagram

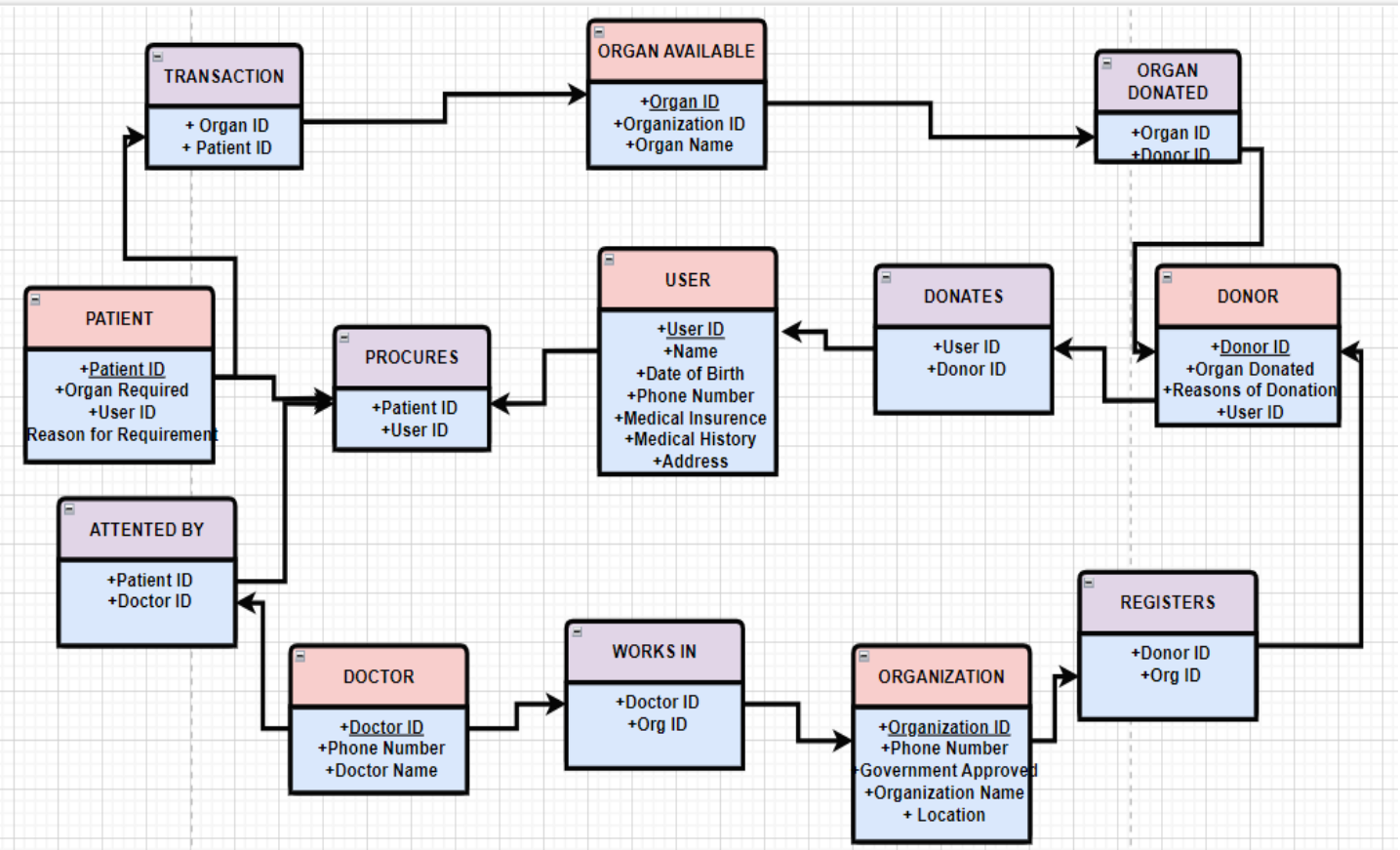


Figure 3.1 Schema Diagram for Organ Donation and Procurement Management system

3.1.1 Schema

Entities:

User (User ID, Name, Date of Birth, Name, Phone number, Medical Insurance, Medical History)

Patient (User ID, Patient ID, Organ Required, Reasons for Procurement)

Donor (Organ Donated, Reasons of Donation, User ID, Donor ID)

Organ Available (Organ ID, Organization ID, Organ Name)

Organization (Organization ID, Phone number, Location, Government Approved, Organization Name)

Doctor (Doctor ID, Doctor Name, Phone Number)

Relationships:

Transaction (Patient ID, Organ ID)

Organ Donated (Organ ID, Donor ID)

Procures (Patient ID, User ID)

Donates (User ID, User ID)

Attended By (Patient ID, Doctor ID)

Works in (Doctor ID, Organization ID)

Registers (Organization ID, Donor ID)

3.2 Relational Tables

3.2.1 DDL Commands and Results

Relational Tables for the Entities and their Attributes:

1) User relation table

```
-- Create a table named user_information
CREATE TABLE user_information (
    USER_ID INT PRIMARY KEY,
    PHONE_NUMBER VARCHAR(15),
    NAME VARCHAR(100) NOT NULL,
    DATE_OF_BIRTH DATE,
    MEDICAL_INSURANCE VARCHAR(50),
    MEDICAL_HISTORY TEXT,
    ADDRESS VARCHAR(255)
);
```

2) Patient relation table

```
CREATE TABLE organ_procurement (
    PROCUREMENT_ID INT PRIMARY KEY,
    REASONS_FOR_PROCUREMENT TEXT,
    ORGAN_REQUIRED VARCHAR(50),
    USER_ID INT,
    PATIENT_ID INT,
    FOREIGN KEY (USER_ID) REFERENCES patient_information(USER_ID),
    FOREIGN KEY (PATIENT_ID) REFERENCES patient_information(USER_ID)
);
```

3) Donor Details table

```
-- Create a table named donation_information
CREATE TABLE donation_information (
    USER_ID INT NOT NULL,
    DONOR_ID INT PRIMARY KEY,
    REASONS_OF_DONATION TEXT,
    ORGAN_DONATED VARCHAR(50)
);
```

4) Organ Available table

```
-- Create a table named organ_organization_info
CREATE TABLE organ_organization_info (
    ORGAN_ID INT PRIMARY KEY,
    ORGANIZATION_ID INT NOT NULL,
    ORGAN_NAME VARCHAR(50) NOT NULL
);
```

5) Organization table

```
-- Create a table named organization_info
CREATE TABLE organization_info (
    ORGANIZATION_ID INT PRIMARY KEY,
    PHONE_NUMBER VARCHAR(15),
    GOVERNMENT_APPROVED_LOCATION VARCHAR(255),
    ORGANIZATION_NAME VARCHAR(100) NOT NULL
);
```

6) Doctor Table

```
-- Create a table named doctor_info
CREATE TABLE doctor_info (
DOCTOR_ID INT PRIMARY KEY,
DOCTOR_NAME VARCHAR(100) NOT NULL,
PHONE_NUMBER VARCHAR(15)
);
```

Relational Tables (for the Relationship between the Entities:

1) Transaction

```
-- Create a table named patient_organ_relation
CREATE TABLE patient_organ_relation (
   PATIENT_ID INT PRIMARY KEY,
   ORGAN_ID INT NOT NULL
);
```

2) Organ Donated

```
-- Create a table named organ_donor_relation
CREATE TABLE organ_donor_relation (
   ORGAN_ID INT PRIMARY KEY,
   DONOR_ID INT NOT NULL
);
```

3) Procures

```
-- Create a table named patient_user_relation
```

```
CREATE TABLE patient_user_relation (
    PATIENT_ID INT PRIMARY KEY,
    USER_ID INT NOT NULL
);
```

4) Donates

```
-- Create a table named user_donor_relation
CREATE TABLE user_donor_relation (
    USER_ID INT PRIMARY KEY,
    DONOR_ID INT NOT NULL
);
```

5) Attended By

```
-- Create a table named patient_doctor_relation
CREATE TABLE patient_doctor_relation (
    PATIENT_ID INT PRIMARY KEY,
    DOCTOR_ID INT NOT NULL
);
```

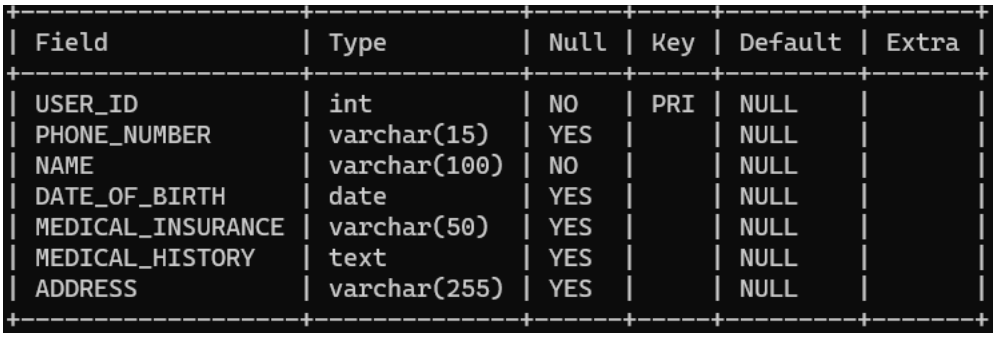
6) Works In

```
-- Create a table named doctor_organization_relation
CREATE TABLE doctor_organization_relation (
    DOCTOR_ID INT PRIMARY KEY,
    ORGANIZATION_ID INT NOT NULL
);
```

7) Registers

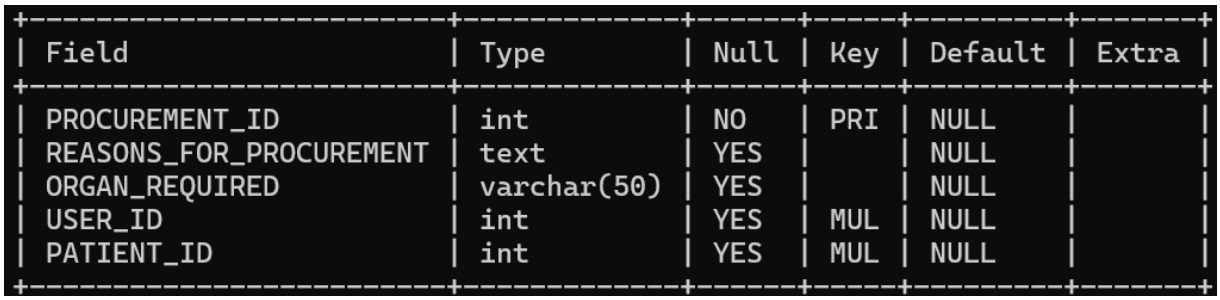
```
-- Create a table named organization_donor_relation

CREATE TABLE organization_donor_relation (

   ORGANIZATION_ID INT PRIMARY KEY,

   DONOR_ID INT NOT NULL

);
```

1) User relation table: DESCRIBE user_information;



Field	Type	Null	Key	Default	Extra
USER_ID	int	NO	PRI	NULL	
PHONE_NUMBER	varchar(15)	YES		NULL	
NAME	varchar(100)	NO		NULL	
DATE_OF_BIRTH	date	YES		NULL	
MEDICAL_INSURANCE	varchar(50)	YES		NULL	
MEDICAL_HISTORY	text	YES		NULL	
ADDRESS	varchar(255)	YES		NULL	

2) Patient relation table: DESCRIBE organ_procurement;



Field	Type	Null	Key	Default	Extra
PROCUREMENT_ID	int	NO	PRI	NULL	
REASONS_FOR_PROCUREMENT	text	YES		NULL	
ORGAN_REQUIRED	varchar(50)	YES		NULL	
USER_ID	int	YES	MUL	NULL	
PATIENT_ID	int	YES	MUL	NULL	

3) Donor Details table: DESCRIBE donation;

Field	Type	Null	Key	Default	Extra
DONATION_ID	int	NO	PRI	NULL	
USER_ID	int	YES	MUL	NULL	
DONOR_ID	int	YES	MUL	NULL	
REASONS_OF_DONATION	text	YES		NULL	
ORGAN_DONATED	varchar(50)	YES		NULL	

4) Organ Available table: DESCRIBE organ_organization;

Field	Type	Null	Key	Default	Extra
ORGAN_ID	int	NO	PRI	NULL	
ORGANIZATION_ID	int	YES	MUL	NULL	
ORGAN_NAME	varchar(50)	YES		NULL	

5) Organization table: organization_info;

Field	Type	Null	Key	Default	Extra
ORGANIZATION_ID	int	NO	PRI	NULL	
PHONE_NUMBER	varchar(15)	YES		NULL	
GOVERNMENT_APPROVED_LOCATION	varchar(255)	YES		NULL	
ORGANIZATION_NAME	varchar(100)	NO		NULL	

6) Doctor Table: DESCRIBE doctor_info;

Field	Type	Null	Key	Default	Extra
DOCTOR_ID	int	NO	PRI	NULL	
DOCTOR_NAME	varchar(100)	NO		NULL	
PHONE_NUMBER	varchar(15)	YES		NULL	

Relational Tables (for the Relationship between the Entities:

1) Transaction: DESCRIBE patient_organ_relation;

Field	Type	Null	Key	Default	Extra
PATIENT_ID	int	NO	PRI	NULL	
ORGAN_ID	int	NO		NULL	

2) Organ Donated: DESCRIBE organ_donor_relation;

Field	Type	Null	Key	Default	Extra
ORGAN_ID	int	NO	PRI	NULL	
DONOR_ID	int	NO		NULL	

3) Procures: DESCRIBE patient_user_relation;

Field	Type	Null	Key	Default	Extra
PATIENT_ID	int	NO	PRI	NULL	
USER_ID	int	NO		NULL	

4) Donates: DESCRIBE user_donor_relation;

Field	Type	Null	Key	Default	Extra
USER_ID	int	NO	PRI	NULL	
DONOR_ID	int	NO		NULL	

5) Attended By: DESCRIBE patient_doctor_relation;

Field	Type	Null	Key	Default	Extra
PATIENT_ID	int	NO	PRI	NULL	
DOCTOR_ID	int	NO		NULL	

6) Works In: DESCRIBE doctor_organization_relation;

Field	Type	Null	Key	Default	Extra
DOCTOR_ID	int	NO	PRI	NULL	
ORGANIZATION_ID	int	NO		NULL	

7) Registers: DESCRIBE organization_donor_relation;

Field	Type	Null	Key	Default	Extra
DOCTOR_ID	int	NO	PRI	NULL	
ORGANIZATION_ID	int	NO		NULL	

Figure 3.2 Create a Tables using DDL Commands for Organ Donation and Procurement Management system.

3.2.1 DML Commands (INSERT) and Results

Relational Tables for the Entities and their Attributes:

1) User relation table

-- Insert some random data into the 'user_information' table

INSERT INTO user_information (USER_ID, PHONE_NUMBER, NAME, DATE_OF_BIRTH, MEDICAL_INSURANCE, MEDICAL_HISTORY, ADDRESS) VALUES

(1, '+1234567890', 'John Doe', '1990-05-15', 'ABC123', 'No significant medical history', '123 Main St, Cityville'),
(2, '+1987654321', 'Jane Smith', '1985-08-22', 'XYZ789', 'Allergies to pollen', '456 Oak St, Townburg'),
(3, '+1555098765', 'Bob Johnson', '1978-12-10', 'PQR456', 'Previous surgery on knee', '789 Pine St, Villageton'),
(4, '+1209876543', 'Alice Jones', '1995-03-03', 'LMN789', 'High blood pressure', '101 Cedar St, Hamlet City'),
(5, '+1666777888', 'Charlie Brown', '1980-11-28', 'JKL123', 'Diabetes type 2', '321 Birch St, Countryside');

2) Patient relation table

-- Inserting random values into the 'organ_procurement' table

INSERT INTO organ_procurement (PROCUREMENT_ID, REASONS_FOR_PROCUREMENT, ORGAN_REQUIRED, USER_ID, PATIENT_ID) VALUES

(2001, 'Emergency transplant', 'Heart', 1001, 1001),

(2002, 'Chronic organ failure', 'Liver', 1002, 1002),

(2003, 'Organ deterioration', 'Kidney', 1003, 1003),

(2004, 'Failed organ transplant', 'Lung', 1004, 1004),

(2005, 'Life-threatening condition', 'Pancreas', 1005, 1005);

3) Donor Details table

-- Insert some random data into the 'donation_information' table

INSERT INTO donation_information (USER_ID, DONOR_ID, REASONS_OF_DONATION, ORGAN_DONATED) VALUES

(101, 201, 'Altruistic donation', 'Kidney'),

(102, 202, 'Helping a family member', 'Liver'),

(103, 203, 'Supporting medical research', 'Heart'),

(104, 204, 'Personal health reasons', 'Lung'),

(105, 205, 'Giving the gift of life', 'Pancreas');

4) Organ Available table

-- Insert some random data into the 'organ_organization_info' table

INSERT INTO organ_organization_info (ORGAN_ID, ORGANIZATION_ID, ORGAN_NAME) VALUES

(301, 401, 'Heart'),

(302, 402, 'Liver'),

(303, 403, 'Kidney'),

(304, 404, 'Lung'),

(305, 405, 'Pancreas');

5) Organization table

-- Insert some random data into the 'organization_info' table

INSERT INTO organization_info (ORGANIZATION_ID, PHONE_NUMBER, GOVERNMENT_APPROVED_LOCATION, ORGANIZATION_NAME) VALUES

(401, '+1234567890', '123 Government Street, Cityville', 'Cityville Medical Center'),

(402, '+1987654321', '456 Official Avenue, Townburg', 'Townburg General Hospital'),

(403, '+1555098765', '789 Approved Road, Villageton', 'Villageton Health Services'),

(404, '+1209876543', '101 Government Lane, Hamlet City', 'Hamlet City Clinic'),

(405, '+1666777888', '321 Government Circle, Countryside', 'Countryside Medical Facility');

6) Doctor Table

-- Insert some random data into the 'doctor_info' table

INSERT INTO doctor_info (DOCTOR_ID, DOCTOR_NAME, PHONE_NUMBER) VALUES

(501, 'Dr. John Smith', '+1234567890'),

(502, 'Dr. Jane Doe', '+1987654321'),

(503, 'Dr. Robert Johnson', '+1555098765'),

(504, 'Dr. Alice Jones', '+1209876543'),

(505, 'Dr. Charlie Brown', '+1666777888');

Relational Tables (for the Relationship between the Entities:

1) Transaction

-- Insert some random data into the 'patient_organ_relation' table

INSERT INTO patient_organ_relation (PATIENT_ID, ORGAN_ID) VALUES

(601, 701),

(602, 702),

(603, 703),

(604, 704),

(605, 705);

2) Organ Donated

-- Insert some random data into the 'organ_donor_relation' table

INSERT INTO organ_donor_relation (ORGAN_ID, DONOR_ID) VALUES

(701, 801),

(702, 802),

(703, 803),

(704, 804),

(705, 805);

3) Procures

```
-- Insert some random data into the 'patient_user_relation' table
INSERT INTO patient_user_relation (PATIENT_ID, USER_ID) VALUES
(801, 901),
(802, 902),
(803, 903),
(804, 904),
(805, 905);
```

4) Donates

```
    -- Insert some random data into the 'user_donor_relation' table
INSERT INTO user_donor_relation (USER_ID, DONOR_ID) VALUES
(901, 1001),
(902, 1002),
(903, 1003),
(904, 1004),
(905, 1005);
```

5) Attended By

```
    -- Insert some random data into the 'patient_doctor_relation' table
INSERT INTO patient_doctor_relation (PATIENT_ID, DOCTOR_ID) VALUES
(1001, 1101),
(1002, 1102),
(1003, 1103),
(1004, 1104),
(1005, 1105);
```

6) Works In

-- Insert some random data into the 'doctor_organization_relation' table

INSERT INTO doctor_organization_relation (DOCTOR_ID, ORGANIZATION_ID) VALUES

(1101, 1201),

(1102, 1202),

(1103, 1203),

(1104, 1204),

(1105, 1205);

7) Registers

-- Insert some random data into the 'organization_donor_relation' table

INSERT INTO organization_donor_relation (ORGANIZATION_ID, DONOR_ID) VALUES

(1201, 1301),

(1202, 1302),

(1203, 1303),

(1204, 1304),

(1205, 1305);

Relational Tables for the Entities and their Attributes:

1) **1) User relation table: select * user_information;**

USER_ID	PHONE_NUMBER	NAME	DATE_OF_BIRTH	MEDICAL_INSURANCE	MEDICAL_HISTORY	ADDRESS
1	+1234567890	John Doe	1990-05-15	ABC123	No significant medical history	123 Main St, Cityville
2	+1987654321	Jane Smith	1985-08-22	XYZ789	Allergies to pollen	456 Oak St, Townburg
3	+1555098765	Bob Johnson	1978-12-10	PQR456	Previous surgery on knee	789 Pine St, Villageton
4	+1209876543	Alice Jones	1995-03-03	LMN789	High blood pressure	101 Cedar St, Hamlet City
5	+1666777888	Charlie Brown	1980-11-28	JKL123	Diabetes type 2	321 Birch St, Countryside

2) Patient relation table: select * organ_procurement;

PROCUREMENT_ID	REASONS_FOR_PROCUREMENT	ORGAN_REQUIRED	USER_ID	PATIENT_ID
2001	Emergency transplant	Heart	1001	1001
2002	Chronic organ failure	Liver	1002	1002
2003	Organ deterioration	Kidney	1003	1003
2004	Failed organ transplant	Lung	1004	1004
2005	Life-threatening condition	Pancreas	1005	1005

3) Donor Details table: select

DONATION_ID	USER_ID	DONOR_ID	REASONS_OF_DONATION	ORGAN_DONATED
3001	1001	4001	Altruistic donation	Heart
3002	1002	4002	Family member in need	Liver
3003	1003	4003	Contribution to medical research	Kidney
3004	1004	4004	Deceased organ donation	Lung
3005	1005	4005	Supporting a friend	Pancreas

*** donation;**

4) Organ Available table: select * organ_organization;

ORGAN_ID	ORGANIZATION_ID	ORGAN_NAME
6001	5001	Heart
6002	5002	Liver
6003	5003	Kidney
6004	5004	Lung
6005	5005	Pancreas

5) Organization table: select * organization_info;

ORGANIZATION_ID	PHONE_NUMBER	GOVERNMENT_APPROVED_LOCATION	ORGANIZATION_NAME
401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility

6) Doctor Table: select * doctor_info;

DOCTOR_ID	DOCTOR_NAME	PHONE_NUMBER
501	Dr. John Smith	+1234567890
502	Dr. Jane Doe	+1987654321
503	Dr. Robert Johnson	+1555098765
504	Dr. Alice Jones	+1209876543
505	Dr. Charlie Brown	+1666777888

Relational Tables (for the Relationship between the Entities:

1) **Transaction: select * patieny_organ_relation;**

PATIENT_ID	ORGAN_ID
601	701
602	702
603	703
604	704
605	705

2) Organ Donated: select * organ_donor_relation;

ORGAN_ID	DONOR_ID
701	801
702	802
703	803
704	804
705	805

3) Procures: select * patient_user_relation;

PATIENT_ID	USER_ID
801	901
802	902
803	903
804	904
805	905

4) Donates: select * user_donor_relation;

USER_ID	DONOR_ID
901	1001
902	1002
903	1003
904	1004
905	1005

5) Attended By: select * patient_doctor_relation;

PATIENT_ID	DOCTOR_ID
1001	1101
1002	1102
1003	1103
1004	1104
1005	1105

6) Works In: select * doctor_organization_relation;

DOCTOR_ID	ORGANIZATION_ID
1101	1201
1102	1202
1103	1203
1104	1204
1105	1205

7) Registers: select * organization_donor_relation;

ORGANIZATION_ID	DONOR_ID
1201	1301
1202	1302
1203	1303
1204	1304
1205	1305

Figure 3.3 Inserting values into Tables using DML Commands for Organ Donation and Procurement Management System.

Chapter-3

4) Complex Queries

- **Constraints:**

1. Ensure uniqueness of the USER_ID column in the user_information table:

```
ALTER TABLE user_information
ADD CONSTRAINT pk_user_id PRIMARY KEY (USER_ID);
```

2. Ensure referential integrity between the USER_ID column in the organ_procurement table and the USER_ID column in the user_information table:

```
ALTER TABLE organ_procurement
ADD CONSTRAINT fk_user_id FOREIGN KEY (USER_ID) REFERENCES
user_information(USER_ID);
```

3. Ensure uniqueness of the ORGAN_NAME column in the organ_organization_info table:

```
ALTER TABLE organ_organization_info
ADD CONSTRAINT uk_organ_name UNIQUE (ORGAN_NAME);
```

4. Ensure that the DATE_OF_BIRTH column in the user_information table contains valid dates:

```
ALTER TABLE user_information
ADD CONSTRAINT chk_date_of_birth CHECK (DATE_OF_BIRTH IS NOT NULL AND
DATE_OF_BIRTH <= CURDATE());
```

5. Ensure that the NAME column in the user_information table cannot contain NULL values:

```
ALTER TABLE user_information
MODIFY COLUMN NAME VARCHAR(100) NOT NULL;
```

- **Sets:**

1. **Query to retrieve users with medical insurance:**

```
SELECT * FROM user_information WHERE MEDICAL_INSURANCE IS NOT NULL;
```

USER_ID	PHONE_NUMBER	NAME	DATE_OF_BIRTH	MEDICAL_INSURANCE	MEDICAL_HISTORY	ADDRESS
1	+1234567890	John Doe	1990-05-15	ABC123	No significant medical history	123 Main St, Cityville
2	+1987654321	Jane Smith	1985-08-22	XYZ789	Allergies to pollen	456 Oak St, Townburg
4	+1209876543	Alice Jones	1995-03-03	LMN789	High blood pressure	101 Cedar St, Hamlet City
5	+1666777888	Charlie Brown	1980-11-28	JKL123	Diabetes type 2	321 Birch St, Countryside
6	+1122334455	Emma Johnson	1987-09-20	XYZ789	No medical history	789 Elm St, Villageton
7	+9988776655	Sophia Brown	1992-03-15	LMN789	Allergic to peanuts	456 Maple St, Townburg
8	+5544332211	Oliver Smith	2025-05-30	ABC123	No significant medical history	101 Oak St, Cityville
9	+1234567890	John Doe	1990-05-15	ABC123	No significant medical history	123 Main St, Cityville
10	+1987654321	Jane Smith	1985-08-22	XYZ789	Allergies to pollen	456 Oak St, Townburg

2. **Query to retrieve organizations with government-approved locations:**

   SELECT * FROM user_information WHERE USER_ID IN (SELECT USER_ID FROM organ_procurement);

ORGANIZATION_ID	PHONE_NUMBER	GOVERNMENT_APPROVED_LOCATION	ORGANIZATION_NAME
401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility

3. **Query to find doctors with assigned phone numbers:**

   SELECT * FROM doctor_info WHERE PHONE_NUMBER IS NOT NULL;

DOCTOR_ID	DOCTOR_NAME	PHONE_NUMBER
501	Dr. John Smith	+1234567890
502	Dr. Jane Doe	+1987654321
503	Dr. Robert Johnson	+1555098765
504	Dr. Alice Jones	+1209876543
505	Dr. Charlie Brown	+1666777888

4. **Query to retrieve users with a specified medical condition in their medical history:**

   SELECT * FROM user_information WHERE MEDICAL_HISTORY LIKE '%diabetes%';

USER_ID	PHONE_NUMBER	NAME	DATE_OF_BIRTH	MEDICAL_INSURANCE	MEDICAL_HISTORY	ADDRESS
5	+1666777888	Charlie Brown	1980-11-28	JKL123	Diabetes type 2	321 Birch St, Countryside

5. **Query to find organizations with a specific phone number:**

   SELECT * FROM organization_info WHERE PHONE_NUMBER = '+1234567890';

ORGANIZATION_ID	PHONE_NUMBER	GOVERNMENT_APPROVED_LOCATION	ORGANIZATION_NAME
401	+1234567890	123 Government Street, Cityville	Cityville Medical Center

- **Joins:**

1. JOIN between Donation and User:

```
SELECT *
FROM donation_information
LEFT JOIN user_information ON donation_information.USER_ID = user_information.USER_ID;
```

USER_ID	DONOR_ID	REASONS_OF_DONATION	ORGAN_DONATED	USER_ID	PHONE_NUMBER	NAME	DATE_OF_BIRTH	MEDICAL_INSURANCE	MEDICAL_HISTORY	ADDRESS
101	201	Altruistic donation	Kidney	NULL	NULL	NULL	NULL	NULL	NULL	NULL
102	202	Helping a family member	Liver	NULL	NULL	NULL	NULL	NULL	NULL	NULL
103	203	Supporting medical research	Heart	NULL	NULL	NULL	NULL	NULL	NULL	NULL
104	204	Personal health reasons	Lung	NULL	NULL	NULL	NULL	NULL	NULL	NULL
105	205	Giving the gift of life	Pancreas	NULL	NULL	NULL	NULL	NULL	NULL	NULL

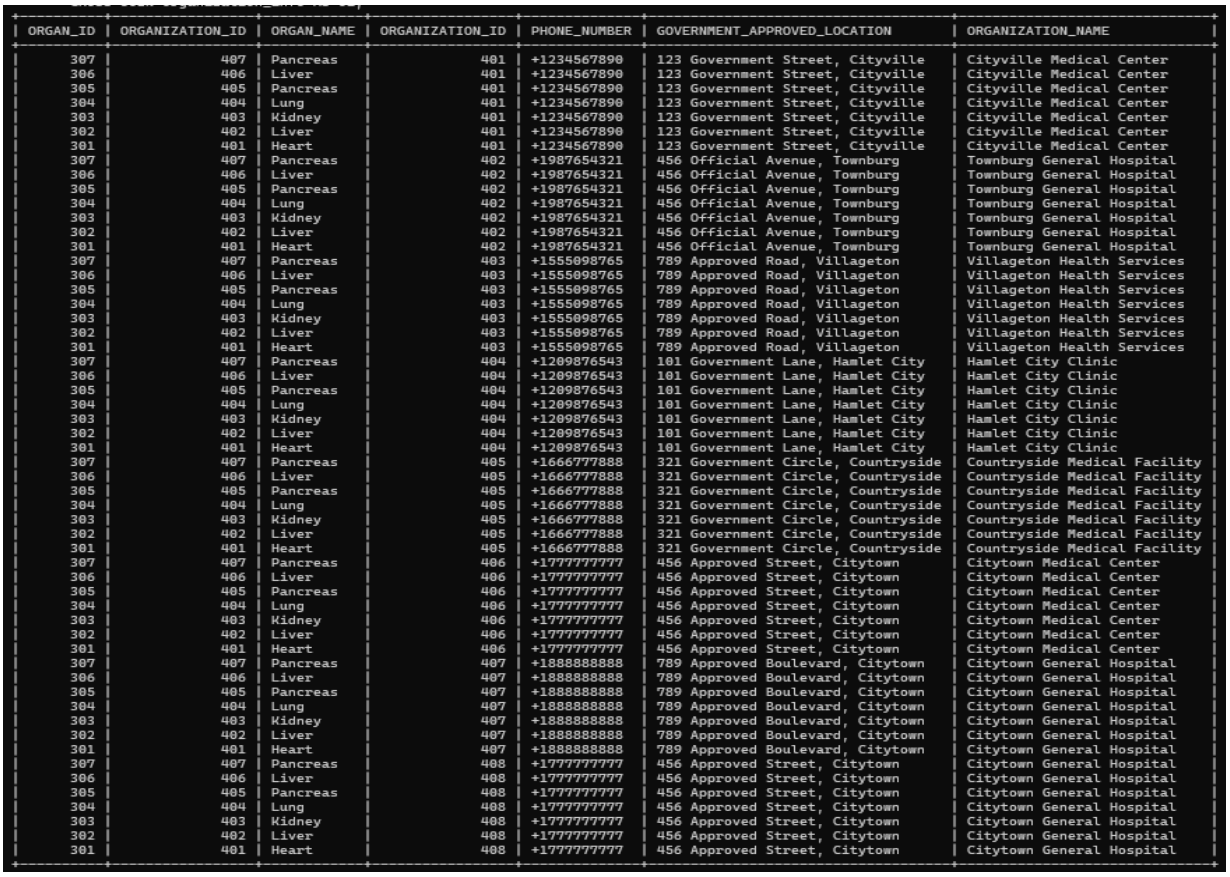
2. CROSS JOIN between Patient and Doctor:

```
SELECT *
FROM patient_doctor_relation
CROSS JOIN doctor_info;
```

PATIENT_ID	DOCTOR_ID	DOCTOR_ID	DOCTOR_NAME	PHONE_NUMBER
1005	1105	501	Dr. John Smith	+1234567890
1004	1104	501	Dr. John Smith	+1234567890
1003	1103	501	Dr. John Smith	+1234567890
1002	1102	501	Dr. John Smith	+1234567890
1001	1101	501	Dr. John Smith	+1234567890
1005	1105	502	Dr. Jane Doe	+1987654321
1004	1104	502	Dr. Jane Doe	+1987654321
1003	1103	502	Dr. Jane Doe	+1987654321
1002	1102	502	Dr. Jane Doe	+1987654321
1001	1101	502	Dr. Jane Doe	+1987654321
1005	1105	503	Dr. Robert Johnson	+1555098765
1004	1104	503	Dr. Robert Johnson	+1555098765
1003	1103	503	Dr. Robert Johnson	+1555098765
1002	1102	503	Dr. Robert Johnson	+1555098765
1001	1101	503	Dr. Robert Johnson	+1555098765
1005	1105	504	Dr. Alice Jones	+1209876543
1004	1104	504	Dr. Alice Jones	+1209876543
1003	1103	504	Dr. Alice Jones	+1209876543
1002	1102	504	Dr. Alice Jones	+1209876543
1001	1101	504	Dr. Alice Jones	+1209876543
1005	1105	505	Dr. Charlie Brown	+1666777888
1004	1104	505	Dr. Charlie Brown	+1666777888
1003	1103	505	Dr. Charlie Brown	+1666777888
1002	1102	505	Dr. Charlie Brown	+1666777888
1001	1101	505	Dr. Charlie Brown	+1666777888
1005	1105	506	Dr. Emily Johnson	+1888888888
1004	1104	506	Dr. Emily Johnson	+1888888888
1003	1103	506	Dr. Emily Johnson	+1888888888
1002	1102	506	Dr. Emily Johnson	+1888888888
1001	1101	506	Dr. Emily Johnson	+1888888888

3. CROSS JOIN between Organ Available and Organization:

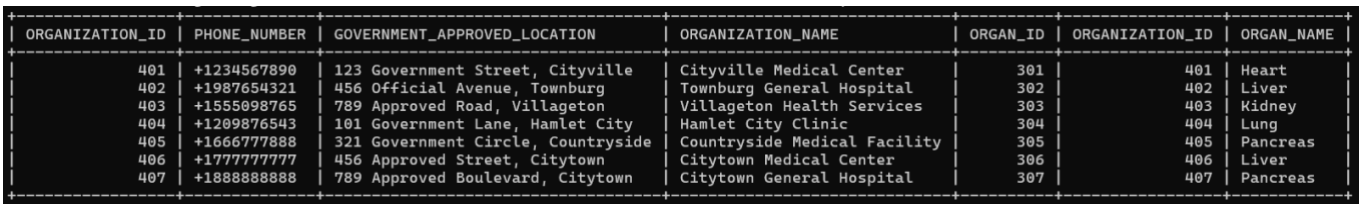
```
SELECT *
FROM organ_organization_info AS oo
CROSS JOIN organization_info AS oi;
```



ORGAN_ID	ORGANIZATION_ID	ORGAN_NAME	ORGANIZATION_ID	PHONE_NUMBER	GOVERNMENT_APPROVED_LOCATION	ORGANIZATION_NAME
307	407	Pancreas	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
306	406	Liver	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
305	405	Pancreas	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
304	404	Lung	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
303	403	Kidney	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
302	402	Liver	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
301	401	Heart	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
307	407	Pancreas	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
306	406	Liver	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
305	405	Pancreas	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
304	404	Lung	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
303	403	Kidney	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
302	402	Liver	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
301	401	Heart	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
307	407	Pancreas	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
306	406	Liver	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
305	405	Pancreas	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
304	404	Lung	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
303	403	Kidney	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
302	402	Liver	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
301	401	Heart	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
307	407	Pancreas	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
306	406	Liver	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
305	405	Pancreas	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
304	404	Lung	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
303	403	Kidney	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
302	402	Liver	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
301	401	Heart	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
307	407	Pancreas	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
306	406	Liver	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
305	405	Pancreas	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
304	404	Lung	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
303	403	Kidney	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
302	402	Liver	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
301	401	Heart	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
307	407	Pancreas	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
306	406	Liver	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
305	405	Pancreas	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
304	404	Lung	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
303	403	Kidney	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
302	402	Liver	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
301	401	Heart	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
307	407	Pancreas	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
306	406	Liver	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
305	405	Pancreas	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
304	404	Lung	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
303	403	Kidney	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
302	402	Liver	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
301	401	Heart	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
307	407	Pancreas	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
306	406	Liver	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
305	405	Pancreas	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
304	404	Lung	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
303	403	Kidney	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
302	402	Liver	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
301	401	Heart	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital

4. RIGHT JOIN between Organization and Organ Available:

```
SELECT *
FROM organization_info AS o
RIGHT JOIN organ_organization_info AS oo ON o.ORGANIZATION_ID =
oo.ORGANIZATION_ID;
```



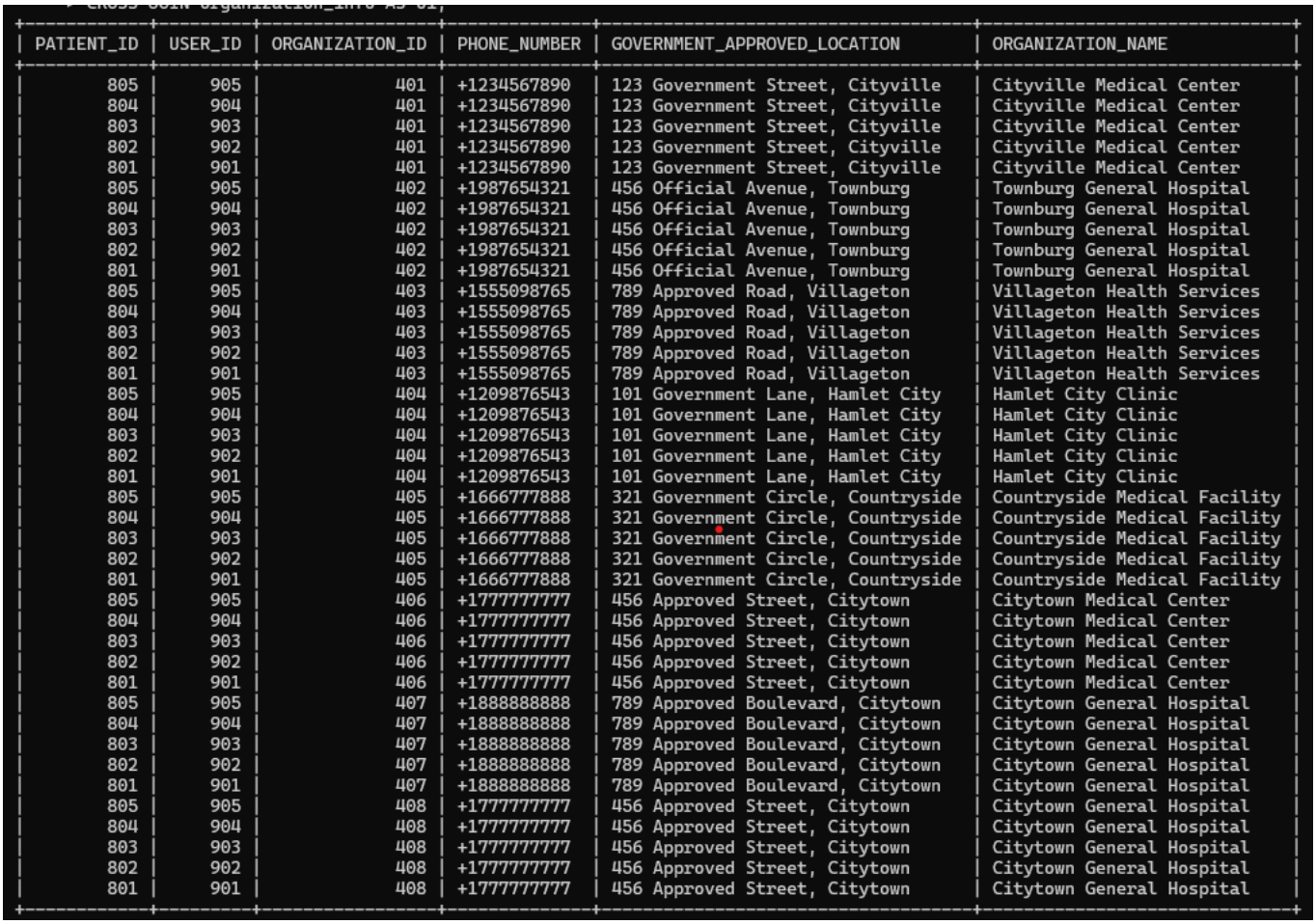
ORGANIZATION_ID	PHONE_NUMBER	GOVERNMENT_APPROVED_LOCATION	ORGANIZATION_NAME	ORGAN_ID	ORGANIZATION_ID	ORGAN_NAME
401	+1234567890	123 Government Street, Cityville	Cityville Medical Center	301	401	Heart
402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital	302	402	Liver
403	+1555098765	789 Approved Road, Villageton	Villageton Health Services	303	403	Kidney
404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic	304	404	Lung
405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility	305	405	Pancreas
406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center	306	406	Liver
407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital	307	407	Pancreas

5. CROSS JOIN between Patient and Organization:

SELECT *

FROM patient_user_relation AS pu

CROSS JOIN organization_info AS oi;



PATIENT_ID	USER_ID	ORGANIZATION_ID	PHONE_NUMBER	GOVERNMENT_APPROVED_LOCATION	ORGANIZATION_NAME
805	905	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
804	904	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
803	903	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
802	902	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
801	901	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
805	905	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
804	904	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
803	903	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
802	902	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
801	901	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
805	905	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
804	904	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
803	903	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
802	902	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
801	901	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
805	905	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
804	904	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
803	903	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
802	902	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
801	901	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
805	905	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
804	904	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
803	903	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
802	902	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
801	901	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
805	905	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
804	904	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
803	903	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
802	902	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
801	901	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
805	905	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
804	904	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
803	903	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
802	902	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
801	901	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
805	905	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
804	904	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
803	903	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
802	902	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
801	901	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital

- **Views:**

1. **View for user_information table:**

   CREATE VIEW user_information_view AS

   SELECT * FROM user_information;

```
mysql> SELECT * FROM user_information;
+---------+--------------+--------------+---------------+-------------------+-------------------------------+--------------------------+
| USER_ID | PHONE_NUMBER | NAME         | DATE_OF_BIRTH | MEDICAL_INSURANCE | MEDICAL_HISTORY               | ADDRESS                  |
+---------+--------------+--------------+---------------+-------------------+-------------------------------+--------------------------+
|       1 | +1234567890  | John Doe     | 1990-05-15    | ABC123            | No significant medical history | 123 Main St, Cityville   |
|       2 | +1987654321  | Jane Smith   | 1985-08-22    | XYZ789            | Allergies to pollen           | 456 Oak St, Townburg     |
|       3 | +1555098765  | Bob Johnson  | 1978-12-10    | PQR456            | Previous surgery on knee      | 789 Pine St, Villageton  |
|       4 | +1209876543  | Alice Jones  | 1995-03-03    | LMN789            | High blood pressure           | 101 Cedar St, Hamlet City |
|       5 | +1666777888  | Charlie Brown | 1980-11-28    | JKL123            | Diabetes type 2               | 321 Birch St, Countryside |
+---------+--------------+--------------+---------------+-------------------+-------------------------------+--------------------------+
```

2. **View for organ_procurement table**:

   CREATE VIEW organ_procurement_view AS

   SELECT * FROM organ_procurement;

```
+----------------+-----------------------------+----------------+---------+------------+
| PROCUREMENT_ID | REASONS_FOR_PROCUREMENT     | ORGAN_REQUIRED | USER_ID | PATIENT_ID |
+----------------+-----------------------------+----------------+---------+------------+
|           2001 | Emergency transplant        | Heart          |    1001 |       1001 |
|           2002 | Chronic organ failure       | Liver          |    1002 |       1002 |
|           2003 | Organ deterioration         | Kidney         |    1003 |       1003 |
|           2004 | Failed organ transplant     | Lung           |    1004 |       1004 |
|           2005 | Life-threatening condition  | Pancreas       |    1005 |       1005 |
+----------------+-----------------------------+----------------+---------+------------+
```

3. **View for donation_information table:**

   CREATE VIEW donation_information_view AS

   SELECT * FROM donation_information;

```
mysql> SELECT * FROM donation_information;
+---------+----------+-----------------------------+----------------+
| USER_ID | DONOR_ID | REASONS_OF_DONATION         | ORGAN_DONATED  |
+---------+----------+-----------------------------+----------------+
|     101 |      201 | Altruistic donation         | Kidney         |
|     102 |      202 | Helping a family member     | Liver          |
|     103 |      203 | Supporting medical research | Heart          |
|     104 |      204 | Personal health reasons     | Lung           |
|     105 |      205 | Giving the gift of life     | Pancreas       |
+---------+----------+-----------------------------+----------------+
```

4. **View for organ_organization_info table:**

   CREATE VIEW organ_organization_info_view AS

   SELECT * FROM organ_organization_info;

```
mysql>  SELECT * FROM organ_organization_info;
+----------+-----------------+------------+
| ORGAN_ID | ORGANIZATION_ID | ORGAN_NAME |
+----------+-----------------+------------+
|      301 |             401 | Heart      |
|      302 |             402 | Liver      |
|      303 |             403 | Kidney     |
|      304 |             404 | Lung       |
|      305 |             405 | Pancreas   |
+----------+-----------------+------------+
```

5. **View for organization_info table:**

CREATE VIEW organization_info_view AS

SELECT * FROM organization_info;

```
mysql> SELECT * FROM organization_info;
+-----------------+--------------+-----------------------------------+-----------------------------+
| ORGANIZATION_ID | PHONE_NUMBER | GOVERNMENT_APPROVED_LOCATION      | ORGANIZATION_NAME           |
+-----------------+--------------+-----------------------------------+-----------------------------+
|             401 | +1234567890  | 123 Government Street, Cityville  | Cityville Medical Center    |
|             402 | +1987654321  | 456 Official Avenue, Townburg     | Townburg General Hospital   |
|             403 | +1555098765  | 789 Approved Road, Villageton     | Villageton Health Services  |
|             404 | +1209876543  | 101 Government Lane, Hamlet City  | Hamlet City Clinic          |
|             405 | +1666777888  | 321 Government Circle, Countryside| Countryside Medical Facility|
+-----------------+--------------+-----------------------------------+-----------------------------+
```

6. **View for doctor_info table:**

CREATE VIEW doctor_info_view AS

SELECT * FROM doctor_info;

```
mysql> SELECT * FROM doctor_info;
+-----------+--------------------+--------------+
| DOCTOR_ID | DOCTOR_NAME        | PHONE_NUMBER |
+-----------+--------------------+--------------+
|       501 | Dr. John Smith     | +1234567890  |
|       502 | Dr. Jane Doe       | +1987654321  |
|       503 | Dr. Robert Johnson | +1555098765  |
|       504 | Dr. Alice Jones    | +1209876543  |
|       505 | Dr. Charlie Brown  | +1666777888  |
+-----------+--------------------+--------------+
```

● **Joins:**

1. JOIN between Donation and User:

SELECT *

FROM donation_information

LEFT JOIN user_information ON donation_information.USER_ID = user_information.USER_ID;

USER_ID	DONOR_ID	REASONS_OF_DONATION	ORGAN_DONATED	USER_ID	PHONE_NUMBER	NAME	DATE_OF_BIRTH	MEDICAL_INSURANCE	MEDICAL_HISTORY	ADDRESS
101	201	Altruistic donation	Kidney	NULL	NULL	NULL	NULL	NULL	NULL	NULL
102	202	Helping a family member	Liver	NULL	NULL	NULL	NULL	NULL	NULL	NULL
103	203	Supporting medical research	Heart	NULL	NULL	NULL	NULL	NULL	NULL	NULL
104	204	Personal health reasons	Lung	NULL	NULL	NULL	NULL	NULL	NULL	NULL
105	205	Giving the gift of life	Pancreas	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. CROSS JOIN between Patient and Doctor:

SELECT *

FROM patient_doctor_relation

CROSS JOIN doctor_info;

PATIENT_ID	DOCTOR_ID	DOCTOR_ID	DOCTOR_NAME	PHONE_NUMBER
1005	1105	501	Dr. John Smith	+1234567890
1004	1104	501	Dr. John Smith	+1234567890
1003	1103	501	Dr. John Smith	+1234567890
1002	1102	501	Dr. John Smith	+1234567890
1001	1101	501	Dr. John Smith	+1234567890
1005	1105	502	Dr. Jane Doe	+1987654321
1004	1104	502	Dr. Jane Doe	+1987654321
1003	1103	502	Dr. Jane Doe	+1987654321
1002	1102	502	Dr. Jane Doe	+1987654321
1001	1101	502	Dr. Jane Doe	+1987654321
1005	1105	503	Dr. Robert Johnson	+1555098765
1004	1104	503	Dr. Robert Johnson	+1555098765
1003	1103	503	Dr. Robert Johnson	+1555098765
1002	1102	503	Dr. Robert Johnson	+1555098765
1001	1101	503	Dr. Robert Johnson	+1555098765
1005	1105	504	Dr. Alice Jones	+1209876543
1004	1104	504	Dr. Alice Jones	+1209876543
1003	1103	504	Dr. Alice Jones	+1209876543
1002	1102	504	Dr. Alice Jones	+1209876543
1001	1101	504	Dr. Alice Jones	+1209876543
1005	1105	505	Dr. Charlie Brown	+1666777888
1004	1104	505	Dr. Charlie Brown	+1666777888
1003	1103	505	Dr. Charlie Brown	+1666777888
1002	1102	505	Dr. Charlie Brown	+1666777888
1001	1101	505	Dr. Charlie Brown	+1666777888
1005	1105	506	Dr. Emily Johnson	+1888888888
1004	1104	506	Dr. Emily Johnson	+1888888888
1003	1103	506	Dr. Emily Johnson	+1888888888
1002	1102	506	Dr. Emily Johnson	+1888888888
1001	1101	506	Dr. Emily Johnson	+1888888888

3. CROSS JOIN between Organ Available and Organization:

SELECT *

FROM organ_organization_info AS oo

CROSS JOIN organization_info AS oi;

ORGAN_ID	ORGANIZATION_ID	ORGAN_NAME	ORGANIZATION_ID	PHONE_NUMBER	GOVERNMENT_APPROVED_LOCATION	ORGANIZATION_NAME
307	407	Pancreas	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
306	406	Liver	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
305	405	Pancreas	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
304	404	Lung	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
303	403	Kidney	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
302	402	Liver	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
301	401	Heart	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
307	407	Pancreas	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
306	406	Liver	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
305	405	Pancreas	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
304	404	Lung	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
303	403	Kidney	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
302	402	Liver	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
301	401	Heart	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
307	407	Pancreas	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
306	406	Liver	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
305	405	Pancreas	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
304	404	Lung	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
303	403	Kidney	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
302	402	Liver	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
301	401	Heart	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
307	407	Pancreas	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
306	406	Liver	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
305	405	Pancreas	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
304	404	Lung	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
303	403	Kidney	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
302	402	Liver	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
301	401	Heart	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
307	407	Pancreas	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
306	406	Liver	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
305	405	Pancreas	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
304	404	Lung	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
303	403	Kidney	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
302	402	Liver	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
301	401	Heart	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
307	407	Pancreas	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
306	406	Liver	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
305	405	Pancreas	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
304	404	Lung	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
303	403	Kidney	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
302	402	Liver	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
301	401	Heart	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
307	407	Pancreas	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
306	406	Liver	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
305	405	Pancreas	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
304	404	Lung	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
303	403	Kidney	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
302	402	Liver	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
301	401	Heart	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
307	407	Pancreas	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
306	406	Liver	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
305	405	Pancreas	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
304	404	Lung	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
303	403	Kidney	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
302	402	Liver	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
301	401	Heart	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital

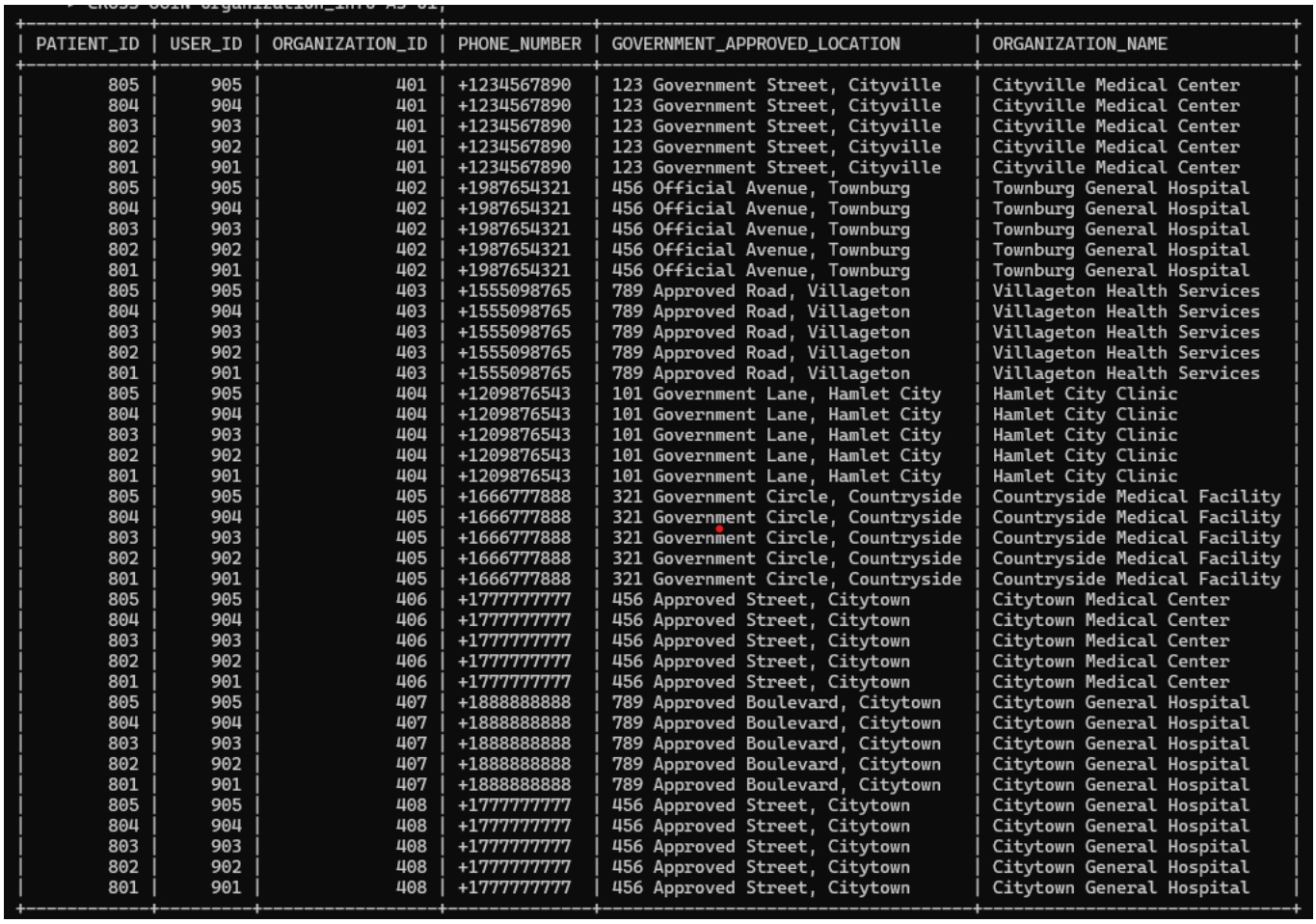
4. RIGHT JOIN between Organization and Organ Available:

```
SELECT *
FROM organization_info AS o
RIGHT JOIN organ_organization_info AS oo ON o.ORGANIZATION_ID =
oo.ORGANIZATION_ID;
```

ORGANIZATION_ID	PHONE_NUMBER	GOVERNMENT_APPROVED_LOCATION	ORGANIZATION_NAME	ORGAN_ID	ORGANIZATION_ID	ORGAN_NAME
401	+1234567890	123 Government Street, Cityville	Cityville Medical Center	301	401	Heart
402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital	302	402	Liver
403	+1555098765	789 Approved Road, Villageton	Villageton Health Services	303	403	Kidney
404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic	304	404	Lung
405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility	305	405	Pancreas
406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center	306	406	Liver
407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital	307	407	Pancreas

5. CROSS JOIN between Patient and Organization:

```
SELECT *
FROM patient_user_relation AS pu
CROSS JOIN organization_info AS oi;
```



PATIENT_ID	USER_ID	ORGANIZATION_ID	PHONE_NUMBER	GOVERNMENT_APPROVED_LOCATION	ORGANIZATION_NAME
805	905	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
804	904	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
803	903	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
802	902	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
801	901	401	+1234567890	123 Government Street, Cityville	Cityville Medical Center
805	905	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
804	904	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
803	903	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
802	902	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
801	901	402	+1987654321	456 Official Avenue, Townburg	Townburg General Hospital
805	905	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
804	904	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
803	903	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
802	902	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
801	901	403	+1555098765	789 Approved Road, Villageton	Villageton Health Services
805	905	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
804	904	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
803	903	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
802	902	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
801	901	404	+1209876543	101 Government Lane, Hamlet City	Hamlet City Clinic
805	905	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
804	904	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
803	903	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
802	902	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
801	901	405	+1666777888	321 Government Circle, Countryside	Countryside Medical Facility
805	905	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
804	904	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
803	903	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
802	902	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
801	901	406	+1777777777	456 Approved Street, Citytown	Citytown Medical Center
805	905	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
804	904	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
803	903	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
802	902	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
801	901	407	+1888888888	789 Approved Boulevard, Citytown	Citytown General Hospital
805	905	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
804	904	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
803	903	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
802	902	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital
801	901	408	+1777777777	456 Approved Street, Citytown	Citytown General Hospital

- **Triggers:**

1. Trigger for Audit Trail:

This trigger records every insert, update, or delete operation on a table in an audit trail table for tracking changes.

DELIMITER //

```
CREATE TRIGGER audit_trail_trigger
AFTER INSERT ON user_information
FOR EACH ROW
BEGIN
   INSERT INTO audit_trail_table (action, table_name, user_id, timestamp)
   VALUES ('INSERT', 'user_information', NEW.USER_ID, NOW());
END;

//

DELIMITER ;
```

```
mysql> INSERT INTO user_information (USER_ID, NAME, PHONE_NUMBER, DATE_OF_BIRTH, MEDICAL_INSURANCE, MEDICAL_HISTORY, ADDRESS)
    -> VALUES (6, 'Emma Johnson', '+1122334455', '1987-09-20', 'XYZ789', 'No medical history', '789 Elm St, Villageton');
Query OK, 1 row affected (0.01 sec)
```

2. Trigger for Auto-Increment:
   This trigger automatically increments a counter whenever a new row is inserted into the table

```
DELIMITER //

CREATE TRIGGER auto_increment_trigger
BEFORE INSERT ON user_information
FOR EACH ROW
BEGIN
   SET NEW.USER_ID = (SELECT MAX(USER_ID) FROM user_information) + 1;
END;

//

DELIMITER ;
```

```
mysql> INSERT INTO user_information (NAME, PHONE_NUMBER, DATE_OF_BIRTH, MEDICAL_INSURANCE, MEDICAL_HISTORY, ADDRESS)
    -> VALUES ('Sophia Brown', '+9988776655', '1992-03-15', 'LMN789', 'Allergic to peanuts', '456 Maple St, Townburg');
Query OK, 1 row affected (0.01 sec)
```

3. Trigger for Data Validation:

   This trigger ensures that certain conditions are met before allowing an insert or update operation.

```
DELIMITER //

CREATE TRIGGER data_validation_trigger
BEFORE INSERT OR UPDATE ON user_information
FOR EACH ROW
BEGIN
   IF NEW.DATE_OF_BIRTH > CURDATE() THEN
      SIGNAL SQLSTATE '45000'
      SET MESSAGE_TEXT = 'Invalid date of birth';
   END IF;
END;

//

DELIMITER ;
```

```
mysql> INSERT INTO user_information (USER_ID, NAME, PHONE_NUMBER, DATE_OF_BIRTH, MEDICAL_INSURANCE, MEDICAL_HISTORY, ADDRESS)
    -> VALUES (8, 'Oliver Smith', '+5544332211', '2025-05-30', 'ABC123', 'No significant medical history', '101 Oak St, Cityville');
Query OK, 1 row affected (0.01 sec)
```

4. Trigger for Cascading Delete:

   This trigger automatically deletes related records in other tables when a record in the main table is deleted.

```
DELIMITER //

CREATE TRIGGER cascading_delete_trigger
BEFORE DELETE ON user_information
FOR EACH ROW
BEGIN
   DELETE FROM organ_procurement WHERE USER_ID = OLD.USER_ID;
```

```
    DELETE FROM donation_information WHERE USER_ID = OLD.USER_ID;
END;

//

DELIMITER ;
```

```
mysql> DELETE FROM user_information WHERE USER_ID = 3;
Query OK, 1 row affected (0.02 sec)
```

5. Trigger for Logging:

This trigger logs specific changes made to certain columns in the table.

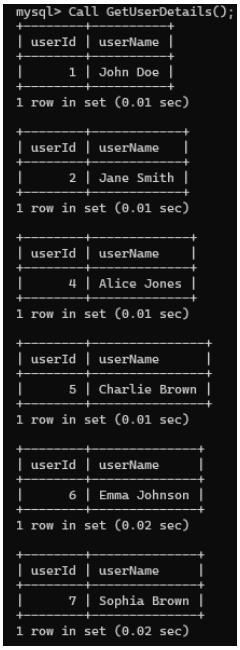
```
DELIMITER //

CREATE TRIGGER logging_trigger
AFTER UPDATE ON user_information
FOR EACH ROW
BEGIN
  IF OLD.NAME <> NEW.NAME THEN
    INSERT INTO log_table (table_name, column_name, old_value, new_value, timestamp)
    VALUES ('user_information', 'NAME', OLD.NAME, NEW.NAME, NOW());
  END IF;
  -- Similar checks for other columns
END;

//
DELIMITER ;
```

- **Cursor:**

1. Query using cursor to retrieve details of all users:

```
DELIMITER //
CREATE PROCEDURE GetUserDetails()
BEGIN
  DECLARE done INT DEFAULT FALSE;
  DECLARE userId INT;
```

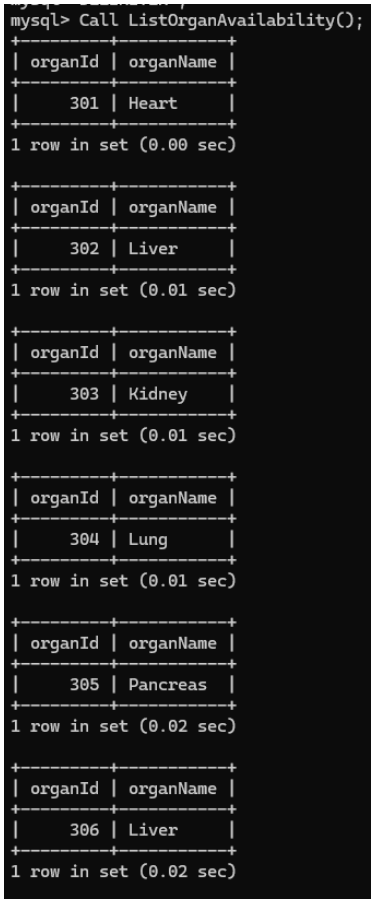
```
    DECLARE userName VARCHAR(100);
    DECLARE cur CURSOR FOR SELECT USER_ID, NAME FROM user_information;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO userId, userName;
        IF done THEN
            LEAVE read_loop;
        END IF;
        SELECT userId, userName;
    END LOOP;
    CLOSE cur;
END //
DELIMITER ;
```



2. Query using cursor to list all organs available:

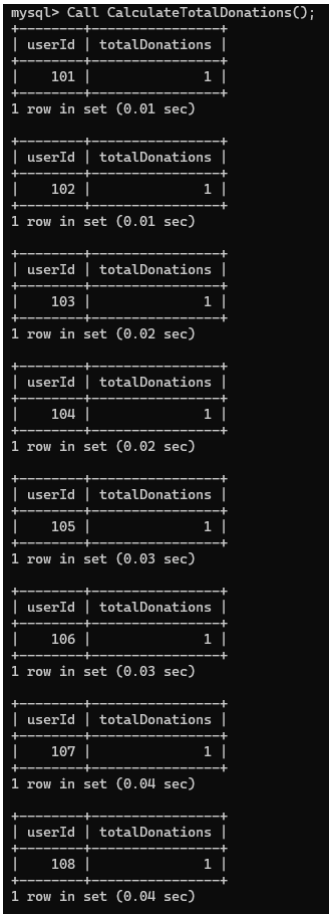
```
DELIMITER //
CREATE PROCEDURE ListOrganAvailability()
BEGIN
    DECLARE done INT DEFAULT FALSE;
  DECLARE organId INT;
```

```
    DECLARE organName VARCHAR(50);
    DECLARE cur CURSOR FOR SELECT ORGAN_ID, ORGAN_NAME FROM
organ_organization_info;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
    OPEN cur;
    read_loop: LOOP
        FETCH cur INTO organId, organName;
        IF done THEN
            LEAVE read_loop;
        END IF;
        SELECT organId, organName;
    END LOOP;
    CLOSE cur;
END //
DELIMITER ;
```



3. Query using cursor to calculate total number of donations per user:

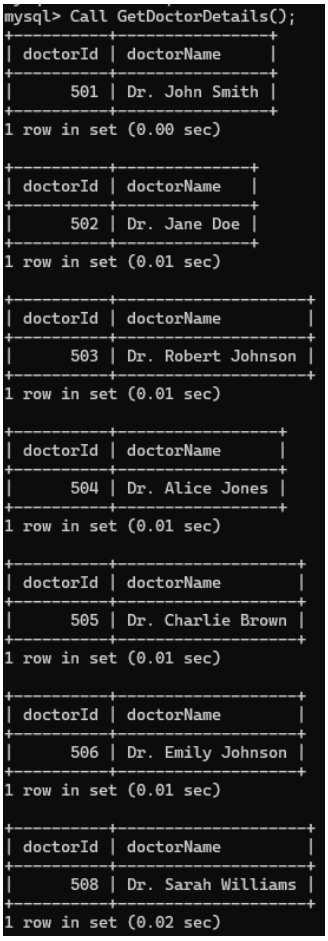
```
DELIMITER //
CREATE PROCEDURE CalculateTotalDonations()
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE userId INT;
   DECLARE totalDonations INT;
   DECLARE cur CURSOR FOR
      SELECT USER_ID, COUNT(*) AS TotalDonations
      FROM donation_information
      GROUP BY USER_ID;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
   OPEN cur;
   read_loop: LOOP
      FETCH cur INTO userId, totalDonations;
      IF done THEN
         LEAVE read_loop;
      END IF;
      SELECT userId, totalDonations;
   END LOOP;
   CLOSE cur;
END //
DELIMITER ;
```



4**. Query using cursor to retrieve details of all doctors:**

```
DELIMITER //
CREATE PROCEDURE GetDoctorDetails()
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE doctorId INT;
   DECLARE doctorName VARCHAR(100);
   DECLARE cur CURSOR FOR SELECT DOCTOR_ID, DOCTOR_NAME FROM doctor_info;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
   OPEN cur;
   read_loop: LOOP
      FETCH cur INTO doctorId, doctorName;
      IF done THEN
         LEAVE read_loop;
      END IF;
      SELECT doctorId, doctorName;
```

```
    END LOOP;
    CLOSE cur;
END //
DELIMITER ;
```

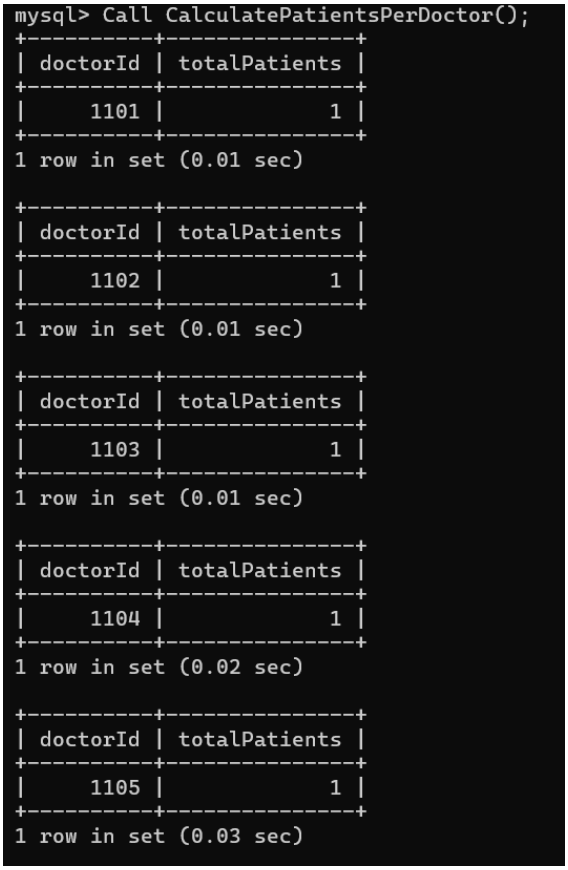


```
mysql> Call GetDoctorDetails();
+----------+----------------+
| doctorId | doctorName     |
+----------+----------------+
|      501 | Dr. John Smith |
+----------+----------------+
1 row in set (0.00 sec)

+----------+--------------+
| doctorId | doctorName   |
+----------+--------------+
|      502 | Dr. Jane Doe |
+----------+--------------+
1 row in set (0.01 sec)

+----------+--------------------+
| doctorId | doctorName         |
+----------+--------------------+
|      503 | Dr. Robert Johnson |
+----------+--------------------+
1 row in set (0.01 sec)

+----------+----------------+
| doctorId | doctorName     |
+----------+----------------+
|      504 | Dr. Alice Jones |
+----------+----------------+
1 row in set (0.01 sec)

+----------+--------------------+
| doctorId | doctorName         |
+----------+--------------------+
|      505 | Dr. Charlie Brown |
+----------+--------------------+
1 row in set (0.01 sec)

+----------+--------------------+
| doctorId | doctorName         |
+----------+--------------------+
|      506 | Dr. Emily Johnson |
+----------+--------------------+
1 row in set (0.01 sec)

+----------+--------------------+
| doctorId | doctorName         |
+----------+--------------------+
|      508 | Dr. Sarah Williams |
+----------+--------------------+
1 row in set (0.02 sec)
```

5. Query using cursor to calculate the total number of patients attended by each doctor:

```
DELIMITER //
CREATE PROCEDURE CalculatePatientsPerDoctor()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE doctorId INT;
    DECLARE totalPatients INT;
    DECLARE cur CURSOR FOR
        SELECT DOCTOR_ID, COUNT(*) AS TotalPatients
        FROM patient_doctor_relation
```

```
    GROUP BY DOCTOR_ID;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
  OPEN cur;
  read_loop: LOOP
    FETCH cur INTO doctorId, totalPatients;
    IF done THEN
      LEAVE read_loop;
    END IF;
    SELECT doctorId, totalPatients;
  END LOOP;
  CLOSE cur;
END //
DELIMITER ;
```



```
mysql> Call CalculatePatientsPerDoctor();
+----------+---------------+
| doctorId | totalPatients |
+----------+---------------+
|     1101 |             1 |
+----------+---------------+
1 row in set (0.01 sec)

+----------+---------------+
| doctorId | totalPatients |
+----------+---------------+
|     1102 |             1 |
+----------+---------------+
1 row in set (0.01 sec)

+----------+---------------+
| doctorId | totalPatients |
+----------+---------------+
|     1103 |             1 |
+----------+---------------+
1 row in set (0.01 sec)

+----------+---------------+
| doctorId | totalPatients |
+----------+---------------+
|     1104 |             1 |
+----------+---------------+
1 row in set (0.02 sec)

+----------+---------------+
| doctorId | totalPatients |
+----------+---------------+
|     1105 |             1 |
+----------+---------------+
1 row in set (0.03 sec)
```

Chapter-4

5. Normalization

5.1 Boyce Codd normal form (BCNF)

User Table:

Possible Pitfalls and Actions Taken:

1.Decomposition to Preserve BCNF:

Action Taken: We removed PHONE_NUMBER and ADDRESS from the users table and placed them into separate tables to address potential functional dependencies that do not include the candidate key (USER_ID). This ensures that each non-key attribute is functionally dependent only on the superkey.

Pitfall: This can lead to excessive table fragmentation, increasing the complexity of queries that need to rejoin these attributes.

2.Loss of Query Efficiency:

Action Taken: By breaking the table into smaller ones, we increased the database's normalization, potentially reducing redundancy and update anomalies.

Pitfall: This impacts query performance since retrieving complete user information now requires joining multiple tables.

3.Maintenance of Data Integrity:

Action Taken: Separating data into multiple tables helps in maintaining data integrity by isolating the dependencies.

Pitfall: Ensuring data consistency across multiple tables becomes more challenging and requires careful foreign key management and transaction handling.

4.Handling of Multi-valued Attributes:

Action Taken: We addressed the issue of PHONE_NUMBER and potentially ADDRESS being multi-valued by moving them to their own tables.

Pitfall: This requires the implementation of additional tables and potentially more complex

integrity constraints to manage these multi-valued relationships properly.

Original User table:

```
CREATE TABLE user_information (
   USER_ID INT PRIMARY KEY,
   PHONE_NUMBER VARCHAR(15),
   NAME VARCHAR(100) NOT NULL,
   DATE_OF_BIRTH DATE,
   MEDICAL_INSURANCE VARCHAR(50),
   MEDICAL_HISTORY TEXT,
   ADDRESS VARCHAR(255)
);
```

USER_ID	PHONE_NUMBER	NAME	DATE_OF_BIRTH	MEDICAL_INSURANCE	MEDICAL_HISTORY	ADDRESS
1	+1234567890	John Doe	1990-05-15	ABC123	No significant medical history	123 Main St, Cityville
2	+1987654321	Jane Smith	1985-08-22	XYZ789	Allergies to pollen	456 Oak St, Townburg
3	+1555098765	Bob Johnson	1978-12-10	PQR456	Previous surgery on knee	789 Pine St, Villageton
4	+1209876543	Alice Jones	1995-03-03	LMN789	High blood pressure	101 Cedar St, Hamlet City
5	+1666777888	Charlie Brown	1980-11-28	JKL123	Diabetes type 2	321 Birch St, Countryside

Normalized User table:(BCNF)

```
CREATE TABLE users (
   USER_ID INT PRIMARY KEY,
   NAME VARCHAR(100) NOT NULL,
   DATE_OF_BIRTH DATE,
   MEDICAL_INSURANCE VARCHAR(50),
   MEDICAL_HISTORY TEXT
);
```

```
mysql> SELECT * FROM users;
+---------+--------------+---------------+-------------------+-------------------------------+
| USER_ID | NAME         | DATE_OF_BIRTH | MEDICAL_INSURANCE | MEDICAL_HISTORY               |
+---------+--------------+---------------+-------------------+-------------------------------+
|       1 | John Doe     | 1990-05-15    | ABC123            | No significant medical history |
|       2 | Jane Smith   | 1985-08-22    | XYZ789            | Allergies to pollen           |
|       3 | Bob Johnson  | 1978-12-10    | PQR456            | Previous surgery on knee      |
|       4 | Alice Jones  | 1995-03-03    | LMN789            | High blood pressure           |
|       5 | Charlie Brown| 1980-11-28    | JKL123            | Diabetes type 2               |
+---------+--------------+---------------+-------------------+-------------------------------+
5 rows in set (0.01 sec)
```

5.2 Third Normal Form (3NF)

Organ Procurement table:

Possible Pitfalls and Actions Taken:

1. Elimination of Transitive Dependencies:

Action Taken: We separated procurement-specific details (REASONS_FOR_PROCUREMENT, ORGAN_REQUIRED) from the procurement-user linkage to avoid transitive dependencies, where non-key attributes (the reasons and organ details) depend on other non-key attributes (USER_ID or PATIENT_ID).

Pitfall: This can make querying complete procurement details more complex as it now likely involves joins across multiple tables.

2. Simplification and Focus on Roles:

Action Taken: The new structure clarifies roles by focusing the procurement_users table solely on linking users to procurements, likely improving data integrity and simplification of the user's role management in procurement processes.

Pitfall: This separation requires additional maintenance and potentially more complex database operations to ensure data consistency across related tables.

3.Data Integrity and Redundancy:

Action Taken: By isolating the procurement details from the user linkage, we potentially reduce redundancy and improve data integrity by ensuring that changes in procurement details do not

require updates to user links or vice versa.

Pitfall: Additional foreign key relationships need to be managed carefully to avoid integrity issues, such as orphan records or inconsistent data across tables.

4.Flexibility in Data Management:

Action Taken: Separating these concerns allows more flexibility in managing different aspects of the procurement process. For instance, updating procurement reasons or required organs doesn't affect the user linkage data.

Pitfall: This approach might require more comprehensive data management strategies, such as transaction management and rollback strategies, to handle operations spanning multiple tables.

Original Organ Procurement table:

```
CREATE TABLE organ_procurement (
   PROCUREMENT_ID INT PRIMARY KEY,
   REASONS_FOR_PROCUREMENT TEXT,
   ORGAN_REQUIRED VARCHAR(50),
   USER_ID INT,
   PATIENT_ID INT,
   FOREIGN KEY (USER_ID) REFERENCES patient_information(USER_ID),
   FOREIGN KEY (PATIENT_ID) REFERENCES patient_information(USER_ID));
```

PROCUREMENT_ID	REASONS_FOR_PROCUREMENT	ORGAN_REQUIRED	USER_ID	PATIENT_ID
2001	Emergency transplant	Heart	1001	1001
2002	Chronic organ failure	Liver	1002	1002
2003	Organ deterioration	Kidney	1003	1003
2004	Failed organ transplant	Lung	1004	1004
2005	Life-threatening condition	Pancreas	1005	1005

Normalized Organ procurement table: (3NF)

```
CREATE TABLE procurement_users (
   PROCUREMENT_ID INT PRIMARY KEY,
   USER_ID INT,
```

```
    FOREIGN KEY (USER_ID) REFERENCES users(USER_ID),
    FOREIGN KEY (PROCUREMENT_ID) REFERENCES
procurement_details(PROCUREMENT_ID)
);
```

```
mysql> SELECT * FROM procurement_details;
+----------------+---------------------------+----------------+
| PROCUREMENT_ID | REASONS_FOR_PROCUREMENT   | ORGAN_REQUIRED |
+----------------+---------------------------+----------------+
|           2001 | Emergency transplant      | Heart          |
|           2002 | Chronic organ failure     | Liver          |
|           2003 | Organ deterioration       | Kidney         |
|           2004 | Failed organ transplant   | Lung           |
|           2005 | Life-threatening condition | Pancreas       |
+----------------+---------------------------+----------------+
5 rows in set (0.00 sec)
```

Donor details table

Possible Pitfalls and Actions Taken:

1.Elimination of Redundancy:

Action Taken: Removing USER_ID from this table prevents redundancy of storing user details multiple times for multiple donations. Instead, a separate table can link USER_ID to multiple DONATION_IDs, supporting many-to-many relationships between users and donations if necessary.

Pitfall: This setup requires additional tables and joins, which can complicate queries but is justified by the cleaner data model and avoidance of redundant data.

2.Improvement in Data Integrity:

Action Taken: By isolating donation-specific information from user-specific data, it simplifies operations on donations (such as updates and deletions) without affecting user data.

Pitfall: More complex data management is needed, such as maintaining consistency across

tables.

3.Focus on Donation Details:

Action Taken: The table now focuses purely on the attributes of donations themselves, which can be beneficial for analyzing donation patterns without interference from user-specific attributes.

Pitfall: Requires more effort in database design to ensure all necessary relationships and data points are accessible through well-structured joins.

4.Data Normalization:

Action Taken: The restructuring ensures that each table contains only data for a single entity type (donations), reducing the scope of update anomalies and maintaining data consistency.

Pitfall: There is a trade-off between normalized data and the complexity of SQL queries needed to gather comprehensive data views.

Original Donor details table:

```
CREATE TABLE donation_information (
   USER_ID INT NOT NULL,
   DONOR_ID INT PRIMARY KEY,
   REASONS_OF_DONATION TEXT,
   ORGAN_DONATED VARCHAR(50)
);
```

DONATION_ID	USER_ID	DONOR_ID	REASONS_OF_DONATION	ORGAN_DONATED
3001	1001	4001	Altruistic donation	Heart
3002	1002	4002	Family member in need	Liver
3003	1003	4003	Contribution to medical research	Kidney
3004	1004	4004	Deceased organ donation	Lung
3005	1005	4005	Supporting a friend	Pancreas

Normalized Donor details table: (3NF)

```
CREATE TABLE donation_details (
   DONATION_ID INT PRIMARY KEY,
   REASONS_OF_DONATION TEXT,
   ORGAN_DONATED VARCHAR(50)
);
```

```
mysql> SELECT * FROM donation_details;
+-------------+---------------------------+---------------+
| DONATION_ID | REASONS_OF_DONATION       | ORGAN_DONATED |
+-------------+---------------------------+---------------+
|         101 | Altruistic donation       | Kidney        |
|         102 | Helping a family member   | Liver         |
|         103 | Supporting medical research | Heart       |
|         104 | Personal health reasons   | Lung          |
|         105 | Giving the gift of life   | Pancreas      |
+-------------+---------------------------+---------------+
```

Chapter-5

Implementation of concurrency control and recovery mechanisms

Organ donation and procurement management systems rely on robust concurrency control and recovery mechanisms to ensure the integrity and availability of critical data. Concurrency control mechanisms allow multiple transactions to operate concurrently without conflicting with each other, minimizing the risk of lost updates and data inconsistencies. Techniques such as locking, timestamp ordering, and optimistic concurrency control are employed to manage concurrent access to shared data efficiently. On the other hand, recovery mechanisms play a vital role in ensuring that the system can recover to a consistent state following failures, such as system crashes or power outages. Techniques like logging and checkpoints are utilized to maintain a comprehensive record of transactions, facilitating efficient recovery and preserving data integrity. These mechanisms are indispensable for maintaining the reliability, consistency, and availability of data in organ donation and procurement management systems, particularly in scenarios involving high transaction volumes and critical healthcare operations.

In the context of the organ donation and procurement management system, the four important aspects of database management are as follows:

1. **Atomicity**: Ensuring that transactions related to organ donation and procurement are executed as atomic units, meaning either all operations within a transaction are successfully completed, or none of them are. This prevents partial updates and inconsistencies in the system, preserving data integrity and accuracy.

2. **Consistency**: Guaranteeing that the database remains in a consistent state before and after the execution of transactions related to organ donation and procurement. This ensures that data remains valid and adheres to predefined constraints, facilitating reliable decision-making and operations.

3**. Isolation**: Ensuring that concurrent transactions related to organ donation and procurement do not interfere with each other. Isolation prevents anomalies such as data corruption and ensures that

each transaction operates independently, maintaining data integrity and preventing conflicts.

4. **Durability**: Ensuring that once transactions related to organ donation and procurement are committed, their effects are permanently saved and persisted in the system. This guarantees that data related to organ donation and procurement remains available and recoverable, even in the event of system failures or crashes, supporting continuous operations and patient care.

Commit:

Committing a Simple Transaction:

```
start TRANSACTION;
INSERT INTO user_information (USER_ID, PHONE_NUMBER, NAME, DATE_OF_BIRTH,
MEDICAL_INSURANCE, MEDICAL_HISTORY, ADDRESS)
VALUES (7, '+1234567890', 'John Doe', '1990-05-15', 'ABC123', 'No significant medical history',
'123 Main St, Cityville');
COMMIT;
```

```
mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)
```

Committing Multiple Transactions:

```
start TRANSACTION;
INSERT INTO organ_procurement (PROCUREMENT_ID,
REASONS_FOR_PROCUREMENT, ORGAN_REQUIRED, USER_ID, PATIENT_ID)
VALUES (6, 'Emergency transplant', 'Heart', 1001, 1001);
start TRANSACTION;
INSERT INTO organ_procurement (PROCUREMENT_ID,
REASONS_FOR_PROCUREMENT, ORGAN_REQUIRED, USER_ID, PATIENT_ID)
VALUES (8, 'Chronic organ failure', 'Liver', 1002, 1002);
COMMIT;
```

```
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
```

Committing after Savepoint:

start TRANSACTION;

INSERT INTO donation_information (USER_ID, DONOR_ID,

REASONS_OF_DONATION, ORGAN_DONATED)

VALUES (10, 201, 'Altruistic donation', 'Kidney');

SAVEPOINT my_savepoint;

INSERT INTO donation_information (USER_ID, DONOR_ID,

REASONS_OF_DONATION, ORGAN_DONATED)

VALUES (9, 202, 'Helping a family member', 'Liver');

COMMIT;

```
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
```

Rollback:

Rolling Back a Transaction:

START TRANSACTION;

INSERT INTO user_information (USER_ID, PHONE_NUMBER, NAME, DATE_OF_BIRTH, MEDICAL_INSURANCE, MEDICAL_HISTORY, ADDRESS)

VALUES (11, '+1234567890', 'Alice Johnson', '1988-07-20', 'DEF456', 'Allergies to peanuts', '456 Elm St, Riverside');

ROLLBACK;

```
Query OK, 1 row affected (0.01 sec)
```

Rolling Back to a Savepoint:

```
START TRANSACTION;

INSERT INTO donation_information (USER_ID, DONOR_ID, REASONS_OF_DONATION, ORGAN_DONATED)

VALUES (12, 301, 'Personal health reasons', 'Heart');

SAVEPOINT my_savepoint2;

INSERT INTO donation_information (USER_ID, DONOR_ID, REASONS_OF_DONATION, ORGAN_DONATED)

VALUES (13, 302, 'Altruistic donation', 'Liver');

ROLLBACK TO my_savepoint2;

COMMIT;
```

```
Query OK, 1 row affected (0.00 sec)
```

Rolling Back Multiple Transactions:

```
START TRANSACTION;

INSERT INTO doctor_info (DOCTOR_ID, DOCTOR_NAME, PHONE_NUMBER)

VALUES (13, 'Dr. Samantha Lee', '+1122334455');

START TRANSACTION;

INSERT INTO doctor_info (DOCTOR_ID, DOCTOR_NAME, PHONE_NUMBER)

VALUES (14, 'Dr. Michael Smith', '+9988776655');
```

ROLLBACK;

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)
```

Savepoint:

Creating a Savepoint:

START TRANSACTION;

INSERT INTO doctor_organization_relation (DOCTOR_ID, ORGANIZATION_ID)

VALUES (15, 501);

SAVEPOINT my_savepoint3;

```
mysql> SAVEPOINT my_savepoint3;
Query OK, 0 rows affected (0.00 sec)
```

Rolling Back to a Savepoint:

ROLLBACK TO my_savepoint3;

```
mysql> ROLLBACK TO my_savepoint3;
Query OK, 0 rows affected (0.00 sec)
```

Releasing a Savepoint:

RELEASE SAVEPOINT my_savepoint3;

```
mysql> RELEASE SAVEPOINT my_savepoint3;
Query OK, 0 rows affected (0.00 sec)
```

Chapter-6

Code for Organ Donation and procurement management system.

1.Main Code:

```
from flask import Flask,render_template,session,request,redirect,url_for,flash
import mysql.connector,hashlib
import matplotlib.pyplot as plt
import numpy as np

mydb = mysql.connector.connect(
  host='localhost',
  user='root',
  password='root',
  database = 'DBMS_PROJECT'
)
mycursor = mydb.cursor(buffered=True)

app = Flask(_name_)

@app.route("/",methods = ['POST', 'GET'])
@app.route("/home",methods = ['POST','GET'])
def home():
    if not session.get('login'):
        return render_template('login.html'),401
    else:
        if session.get('isAdmin') :
            return render_template('home.html',username=session.get('username'))
        else :
            return home_student()

@app.route("/login",methods = ['GET','POST'])
```

```
def login():
    if request.method=='POST' :
        query = """SELECT * FROM login WHERE username = '%s'"""
%(request.form['username'])
        mycursor.execute(query)
        res = mycursor.fetchall()
        if mycursor.rowcount == 0:
            return home()
        if request.form['password'] != res[0][1]:
            return render_template('login.html')
        else:
            session['login'] = True
            session['username'] = request.form['username']
            session['password'] = request.form['password']
            session['isAdmin'] = (request.form['username']=='admin')
            return home()
    return render_template('login.html')

@app.route("/show_update_detail",methods=['POST','GET'])
def show_update_detail():
    if not session.get('login'):
        return redirect( url_for('home') )
    if request.method=='POST':
        if request.form['User_ID'] ==":
            return render_template("search_detail.html")
        qry = "Select * from User where User.User_ID = %s"
%(request.form['User_ID'])
        qry1 = "Select * from User_phone_no where User_ID = %s"
%(request.form['User_ID'])
        mycursor.execute(qry)
        not_found=False
```

```
        res=()
        if(mycursor.rowcount > 0):
            res = mycursor.fetchone()
        else:
            not_found=True
        fields = mycursor.column_names
        qry_upd = "Select * from User where User_ID = %s"
%(request.form['User_ID'])
        mycursor.execute(qry_upd)
        upd_res = ()
        if(mycursor.rowcount > 0):
            upd_res = mycursor.fetchone()
        fields_upd = mycursor.column_names
        mycursor.execute(qry1)
        phone_no = mycursor.fetchall()
        qry_pat = "select Patient_ID, organ_req, reason_of_procurement, Doctor_name
from Patient inner join Doctor on Doctor.Doctor_ID = Patient.Doctor_ID and
User_ID = %s" %(request.form['User_ID'])
        qry_don = "select Donor_ID, organ_donated, reason_of_donation,
Organization_name from Donor inner join Organization on
Organization.Organization_ID = Donor.Organization_ID and User_ID = %s"
%(request.form['User_ID'])
        qry_trans = "select distinct Transaction.Patient_ID, Transaction.Donor_ID,
Organ_ID, Date_of_transaction, Status from Transaction, Patient, Donor where
(Patient.User_ID = %s and Patient.Patient_ID = Transaction.Patient_ID) or
(Donor.User_Id= %s and Donor.Donor_ID = Transaction.Donor_ID)"
%((request.form['User_ID']),(request.form['User_ID']))
        #
        res_pat = ()
        res_dnr = ()
        res_trans = ()
```

```
        mycursor.execute(qry_pat)
        if(mycursor.rowcount > 0):
            res_pat = mycursor.fetchall()
        fields_pat = mycursor.column_names
        #
        mycursor.execute(qry_don)
        if(mycursor.rowcount > 0):
            res_dnr = mycursor.fetchall()
        fields_dnr = mycursor.column_names
        #
        mycursor.execute(qry_trans)
        if(mycursor.rowcount > 0):
            res_trans = mycursor.fetchall()
        fields_trans = mycursor.column_names
        print(res_trans)
        if("show" in request.form):
            return render_template('show_detail_2.html',res = res,fields = fields,
not_found=not_found, phone_no = phone_no, res_dnr = res_dnr, res_pat =
res_pat,res_trans = res_trans,fields_trans = fields_trans, fields_dnr = fields_dnr,
fields_pat = fields_pat)
        if("update" in request.form):
            return render_template('update_detail.html',res = upd_res,fields = fields_upd,
not_found=not_found)
        if "delete" in request.form:
            if not_found:
                return render_template('show_detail_2.html',res = res,fields = fields,
not_found=not_found, phone_no = phone_no,  res_dnr = res_dnr, res_pat =
res_pat,res_trans = res_trans,fields_trans = fields_trans, fields_dnr = fields_dnr,
fields_pat = fields_pat)
            else:
                qry2 = "DELETE FROM User where User_ID = %s"
```

```
%(request.form['User_ID'])
            mycursor.execute(qry2)
            mydb.commit()
            return render_template("home.html")

@app.route("/search_detail",methods = ['POST','GET'])
def search_detail():
    if not session.get('login'):
        return redirect( url_for('home') )
    return render_template('search_detail.html')
```

2.Adding Information

```
@app.route("/add_<id>_page",methods = ['POST','GET'])
def add_page(id):
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from " + id.capitalize()
    mycursor.execute(qry)
    fields = mycursor.column_names

    return render_template('add_page.html',success=request.args.get('success'),
error=request.args.get('error'), fields = fields, id= id)

@app.route("/add_User", methods=['POST','GET'])
def add_User():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from User"
    mycursor.execute(qry)
    fields = mycursor.column_names
```

```
    val = ()

    for field in fields:
        temp = request.form.get(field)
        if field not in ['User_ID','Medical_insurance'] and temp != '':
            temp = "\'"+temp+"\'"
        if temp == '':
            temp = 'NULL'
        val = val + (temp,)

    qry = "INSERT INTO User Values (%s,%s,%s,%s,%s,%s,%s,%s)"%val
    print(qry)
    success = True
    error = False
    try:
        mycursor.execute(qry)
    except:
        print("Error : User not Inserted")
        error = True
        success = False
    mydb.commit()

    return redirect(url_for('add_page', id='User', error=error,success=success))

@app.route("/add_User_phone_no", methods=['POST','GET'])
def add_User_phone_no():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from User_phone_no"
    mycursor.execute(qry)
    fields = mycursor.column_names
```

```
    val = ()

    for field in fields:
        temp = request.form.get(field)
        if field not in ['User_ID','Phone_no'] and temp != '':
            temp = "\'"+temp+"\'"
        if temp == '':
            temp = 'NULL'
        val = val + (temp,)

    qry = "INSERT INTO User_phone_no Values (%s,%s)"%val
    print(qry)
    success = True
    error = False
    try:
        mycursor.execute(qry)
    except:
        print("Error : User not Inserted")
        error = True
        success = False
    mydb.commit()

    return redirect(url_for('add_page', id='User_phone_no',
error=error,success=success))

@app.route("/add_Patient", methods=['POST','GET'])
def add_Patient():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Patient"
```

```
    mycursor.execute(qry)
    fields = mycursor.column_names

    val = ()

    for field in fields:
        temp = request.form.get(field)
        if field not in ['Patient_ID','User_ID','Doctor_ID'] and temp != '':
            temp = "\'"+temp+"\'"
        if temp == '':
            temp = 'NULL'
        val = val + (temp,)

    qry = "INSERT INTO Patient Values (%s,%s,%s,%s,%s)"%val
    print(qry)
    success = True
    error = False
    try:
        mycursor.execute(qry)
    except:
        print("Error : User not Inserted")
        error = True
        success = False
    mydb.commit()

    return redirect(url_for('add_page', id='Patient', error=error,success=success))

@app.route("/add_Donor", methods=['POST','GET'])
def add_Donor():
    if not session.get('login'):
        return redirect( url_for('home') )
```

```
    qry = "SELECT * from Donor"
    mycursor.execute(qry)
    fields = mycursor.column_names
    val = ()
    for field in fields:
        temp = request.form.get(field)
        if field not in ['Donor_ID','User_ID','Organization_ID'] and temp != '':
            temp = "\'"+temp+"\'"
        if temp == '':
            temp = 'NULL'
        val = val + (temp,)
    mycursor.execute( "START TRANSACTION;" )
    qry = "INSERT INTO Donor Values (%s,%s,%s,%s,%s)"%val
    print(qry)
    success = True
    error = False
    try:
        mycursor.execute(qry)
    except:
        print("Error : User not Inserted")
        error = True
        success = False

    qry_insert = "insert into Organ_available (Organ_name, Donor_ID) Values
(%s,%s) "%(val[1],val[0])

    mycursor.execute(qry_insert)

    mycursor.execute("COMMIT;")

    mydb.commit()
```

```
    return redirect(url_for('add_page', id='Donor', error=error,success=success))

@app.route("/add_Doctor", methods=['POST','GET'])
def add_Doctor():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Doctor"
    mycursor.execute(qry)
    fields = mycursor.column_names

    val = ()

    for field in fields:
        temp = request.form.get(field)
        if field not in ['Doctor_ID','Organization_ID'] and temp != '':
            temp = "\""+temp+"\""
        if temp == '':
            temp = 'NULL'
        val = val + (temp,)

    qry = "INSERT INTO Doctor Values (%s,%s,%s,%s)"%val
    print(qry)
    success = True
    error = False
    try:
        mycursor.execute(qry)
    except:
        print("Error : User not Inserted")
        error = True
        success = False
```

```
    mydb.commit()

    return redirect(url_for('add_page', id='Doctor', error=error,success=success))

@app.route("/add_Doctor_phone_no", methods=['POST','GET'])
def add_Doctor_phone_no():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Doctor_phone_no"
    mycursor.execute(qry)
    fields = mycursor.column_names

    val = ()

    for field in fields:
        temp = request.form.get(field)
        if field not in ['Doctor_ID','Phone_no'] and temp != '':
            temp = "\""+temp+"\""
        if temp == '':
            temp = 'NULL'
        val = val + (temp,)

    qry = "INSERT INTO Doctor_phone_no Values (%s,%s)"%val
    print(qry)
    success = True
    error = False
    try:
        mycursor.execute(qry)
    except:
        print("Error : User not Inserted")
        error = True
```

```
        success = False
    mydb.commit()

    return redirect(url_for('add_page', id='Doctor_phone_no',
error=error,success=success))

@app.route("/add_Organ_available", methods=['POST','GET'])
def add_Organ_available():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Organ_available"
    mycursor.execute(qry)
    fields = mycursor.column_names

    val = ()

    for field in fields:
        temp = request.form.get(field)
        if field not in ['Organ_ID','Donor_ID'] and temp != '':
            temp = "\'"+temp+"\'"
        if temp == '':
            temp = 'NULL'
        val = val + (temp,)

    qry = "INSERT INTO Organ_available Values (%s,%s,%s)"%val
    print(qry)
    success = True
    error = False
    try:
        mycursor.execute(qry)
    except:
```

```
        print("Error : User not Inserted")
        error = True
        success = False
    mydb.commit()

    return redirect(url_for('add_page', id='Organ_available',
error=error,success=success))



@app.route("/add_Organization", methods=['POST','GET'])
def add_Organization():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Organization"
    mycursor.execute(qry)
    fields = mycursor.column_names

    val = ()

    for field in fields:
        temp = request.form.get(field)
        if field not in ['Government_approved','Organization_ID'] and temp != '':
            temp = "\'"+temp+"\'"
        if temp == '':
            temp = 'NULL'
        val = val + (temp,)

    qry = "INSERT INTO Organization Values (%s,%s,%s,%s)"%val
    print(qry)
    success = True
    error = False
```

```
    try:
        mycursor.execute(qry)
    except:
        print("Error : User not Inserted")
        error = True
        success = False
    mydb.commit()

    return redirect(url_for('add_page', id='Organization', error=error,success=success))

@app.route("/add_Organization_phone_no", methods=['POST','GET'])
def add_Organization_phone_no():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Organization_phone_no"
    mycursor.execute(qry)
    fields = mycursor.column_names

    val = ()

    for field in fields:
        temp = request.form.get(field)
        if field not in ['Organization_ID','Phone_no'] and temp != '':
            temp = "\'"+temp+"\'"
        if temp == '':
            temp = 'NULL'
        val = val + (temp,)

    qry = "INSERT INTO Organization_phone_no Values (%s,%s)"%val
    print(qry)
    success = True
```

```
    error = False
    try:
        mycursor.execute(qry)
    except:
        print("Error : User not Inserted")
        error = True
        success = False
    mydb.commit()

    return redirect(url_for('add_page', id='Organization_phone_no',
error=error,success=success))

@app.route("/add_Organization_head", methods=['POST','GET'])
def add_Organization_head():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Organization_head"
    mycursor.execute(qry)
    fields = mycursor.column_names

    val = ()

    for field in fields:
        temp = request.form.get(field)
        if field not in ['Employee_ID','Term_length','Organization_ID'] and temp != '':
            temp = "\'"+temp+"\'"
        if temp == '':
            temp = 'NULL'
        val = val + (temp,)

    qry = "INSERT INTO Organization_head Values (%s,%s,%s,%s,%s)"%val
```

```
    print(qry)
    success = True
    error = False
    try:
        mycursor.execute(qry)
    except:
        print("Error : User not Inserted")
        error = True
        success = False
    mydb.commit()

    return redirect(url_for('add_page', id='Organization_head',
error=error,success=success))

@app.route("/add_Transaction", methods=['POST','GET'])
def add_Transaction_head():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Transaction"
    mycursor.execute(qry)
    fields = mycursor.column_names

    val = ()

    for field in fields:
        temp = request.form.get(field)
        if field not in ['Patient_ID','Donor_ID','Status','Organ_ID'] and temp != '':
            temp = "\'"+temp+"\'"
        if temp == '':
            temp = 'NULL'
        val = val + (temp,)
```

```
    mycursor.execute( "START TRANSACTION;" )
    qry = "INSERT INTO Transaction Values (%s,%s,%s,%s,%s)"%val
    print(qry)
    success = True
    error = False
    try:
        mycursor.execute(qry)
    except:
        print("Error : User not Inserted")
        error = True
        success = False
    qry_insert = "delete from Organ_available where Organ_ID = %s "%val[1]
    mycursor.execute(qry_insert)
    mycursor.execute("COMMIT;")
    mydb.commit()

    return redirect(url_for('add_page', id='Transaction', error=error,success=success))
```

3.Update details:

```
@app.route("/update_user_page",methods = ['POST','GET'])
def update_user_page():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry_upd = "Select * from User"
    mycursor.execute(qry_upd)
    fields_upd = mycursor.column_names
    upd_res=[None]*len(fields_upd)
    return render_template('update_user_page.html',fields = fields_upd,res = upd_res)
```

```
@app.route("/update_user_details",methods = ['GET','POST'])
def update_details():
    if not session.get('login'):
        return redirect( url_for('home') )
    mycursor.execute("SELECT * from User")
    fields = mycursor.column_names
    qry = "UPDATE User SET "
    for field in fields:
        if request.form[field] not in ['None',"]:
            if field in ['User_ID','Medical_insurance']:
                qry = qry + "%s = %s , " %(field,request.form[field])
            else:
                qry = qry + " %s = \'%s\' , " %(field,request.form[field])
        else:
            qry = qry + "%s = NULL , " %(field)
    qry = qry[:-2]
    qry = qry + "WHERE User_ID = %s;" %(request.form['User_ID'])
    print(qry)
    try:
        mycursor.execute(qry)
    except:
        print("update error")
    mydb.commit()
    qry2 = "select * from User where User_ID = %s" %(request.form['User_ID'])
    mycursor.execute(qry2)
    res = mycursor.fetchone()
    qry = "Select * from User where User.User_ID = %s" %(request.form['User_ID'])
    qry1 = "Select * from User_phone_no where User_ID = %s"
%(request.form['User_ID'])
    mycursor.execute(qry)
    not_found=False
```

```
    res=()
    if(mycursor.rowcount > 0):
        res = mycursor.fetchone()
    else:
        not_found=True
    fields = mycursor.column_names
    qry_upd = "Select * from User where User_ID = %s" %(request.form['User_ID'])
    mycursor.execute(qry_upd)
    upd_res = ()
    if(mycursor.rowcount > 0):
        upd_res = mycursor.fetchone()
    fields_upd = mycursor.column_names
    mycursor.execute(qry1)
    phone_no = mycursor.fetchall()
    qry_pat = "select Patient_ID, organ_req, reason_of_procurement, Doctor_name
from Patient inner join Doctor on Doctor.Doctor_ID = Patient.Doctor_ID and
User_ID = %s" %(request.form['User_ID'])
    qry_don = "select Donor_ID, organ_donated, reason_of_donation,
Organization_name from Donor inner join Organization on
Organization.Organization_ID = Donor.Organization_ID and User_ID = %s"
%(request.form['User_ID'])
    qry_trans = "select distinct Transaction.Patient_ID, Transaction.Donor_ID,
Organ_ID, Date_of_transaction, Status from Transaction, Patient, Donor where
(Patient.User_ID = %s and Patient.Patient_ID = Transaction.Patient_ID) or
(Donor.User_Id= %s and Donor.Donor_ID = Transaction.Donor_ID)"
%((request.form['User_ID']),(request.form['User_ID']))
    #
    res_pat = ()
    res_dnr = ()
    res_trans = ()
    mycursor.execute(qry_pat)
```

```
    if(mycursor.rowcount > 0):
        res_pat = mycursor.fetchall()
    fields_pat = mycursor.column_names
    #
    mycursor.execute(qry_don)
    if(mycursor.rowcount > 0):
        res_dnr = mycursor.fetchall()
    fields_dnr = mycursor.column_names
    #
    mycursor.execute(qry_trans)
    if(mycursor.rowcount > 0):
        res_trans = mycursor.fetchall()
    fields_trans = mycursor.column_names
    # if("show" in request.form):
    return render_template('show_detail_2.html',res = res,fields = fields,
not_found=not_found, phone_no = phone_no, res_dnr = res_dnr, res_pat =
res_pat,res_trans = res_trans,fields_trans = fields_trans, fields_dnr = fields_dnr,
fields_pat = fields_pat)
    # return render_template("show_detail.html",res = res,fields=fields,not_found =
False)

@app.route("/update_patient_page",methods = ['POST','GET'])
def update_patient_page():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry_upd = "Select * from Patient"
    mycursor.execute(qry_upd)
    fields_upd = mycursor.column_names
    upd_res=[None]*len(fields_upd)
    return render_template('update_patient_page.html',fields = fields_upd,res =
upd_res)
```

```
@app.route("/update_patient_details",methods = ['GET','POST'])
def update_patient_details():
    if not session.get('login'):
        return redirect( url_for('home') )
    mycursor.execute("SELECT * from Patient")
    fields = mycursor.column_names
    qry = "UPDATE Patient SET "
    for field in fields:
        if request.form[field] not in ['None',"]:
            if field in ['User_ID','Doctor_ID','Patient_ID']:
                qry = qry + "%s = %s , " %(field,request.form[field])
            else:
                qry = qry + " %s = \'%s\' , " %(field,request.form[field])
        else:
            qry = qry + "%s = NULL , " %(field)
    qry = qry[:-2]
    qry = qry + "WHERE Patient_ID = %s and organ_req = \'%s\';"
%(request.form['Patient_ID'],request.form['organ_req'])
    print(qry)
    try:
        mycursor.execute(qry)
    except:
        print("update error")
    mydb.commit()
    qry2 = "select * from Patient WHERE Patient_ID = %s and organ_req = \'%s\';"
%(request.form['Patient_ID'],request.form['organ_req'])
    mycursor.execute(qry2)
    res = mycursor.fetchone()
    print(res)
    print(qry2)
```

```
    return render_template("show_detail.html",res = res,fields=fields,not_found =
False)

@app.route("/update_donor_page",methods = ['POST','GET'])
def update_donor_page():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry_upd = "Select * from Donor"
    mycursor.execute(qry_upd)
    fields_upd = mycursor.column_names
    upd_res=[None]*len(fields_upd)
    return render_template('update_donor_page.html',fields = fields_upd,res =
upd_res)

@app.route("/update_donor_details",methods = ['GET','POST'])
def update_donor_details():
    if not session.get('login'):
        return redirect( url_for('home') )
    mycursor.execute("SELECT * from Donor")
    fields = mycursor.column_names
    qry = "UPDATE Donor SET "
    for field in fields:
        if request.form[field] not in ['None',"]:
            if field in ['User_ID','Organization_ID','Donor_ID']:
                qry = qry + "%s = %s , " %(field,request.form[field])
            else:
                qry = qry + " %s = \'%s\' , " %(field,request.form[field])
        else:
            qry = qry + "%s = NULL , " %(field)
    qry = qry[:-2]
    qry = qry + "WHERE Donor_ID = %s and organ_donated = \"%s\";"
```

```
%(request.form['Donor_ID'],request.form['organ_donated'])
    print(qry)
    try:
        mycursor.execute(qry)
    except:
        print("update error")
    mydb.commit()
    qry2 = "select * from Patient WHERE Donor_ID = %s and organ_donated =
\"%s\";" %(request.form['Donor_ID'],request.form['organ_donated'])
    mycursor.execute(qry2)
    res = mycursor.fetchone()
    print(res)
    print(qry2)
    return render_template("show_detail.html",res = res,fields=fields,not_found =
False)

@app.route("/update_doctor_page",methods = ['POST','GET'])
def update_doctor_page():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry_upd = "Select * from Doctor"
    mycursor.execute(qry_upd)
    fields_upd = mycursor.column_names
    upd_res=[None]*len(fields_upd)
    return render_template('update_doctor_page.html',fields = fields_upd,res =
upd_res)

@app.route("/update_doctor_details",methods = ['GET','POST'])
def update_doctor_details():
    if not session.get('login'):
        return redirect( url_for('home') )
```

```
    mycursor.execute("SELECT * from Doctor")
    fields = mycursor.column_names
    qry = "UPDATE Doctor SET "
    for field in fields:
        if request.form[field] not in ['None',"]:
            if field in ['Doctor_ID','Organization_ID']:
                qry = qry + "%s = %s , " %(field,request.form[field])
            else:
                qry = qry + " %s = \'%s\' , " %(field,request.form[field])
        else:
            qry = qry + "%s = NULL , " %(field)
    qry = qry[:-2]
    qry = qry + "WHERE Doctor_ID = %s;" %(request.form['Doctor_ID'])
    print(qry)
    try:
        mycursor.execute(qry)
    except:
        print("update error")
        return render_template('error_page.html')
    mydb.commit()
    qry2 = "select * from Doctor WHERE Doctor_ID = %s;"
%(request.form['Doctor_ID'])
    mycursor.execute(qry2)
    res = mycursor.fetchone()
    return render_template("show_detail.html",res = res,fields=fields,not_found =
False)

@app.route("/update_organization_page",methods = ['POST','GET'])
def update_organization_page():
    if not session.get('login'):
        return redirect( url_for('home') )
```

```
    qry_upd = "Select * from Organization"
    mycursor.execute(qry_upd)
    fields_upd = mycursor.column_names
    upd_res=[None]*len(fields_upd)
    return render_template('update_organization_page.html',fields = fields_upd,res =
upd_res)

@app.route("/update_organization_details",methods = ['GET','POST'])
def update_organization_details():
    if not session.get('login'):
        return redirect( url_for('home') )
    mycursor.execute("SELECT * from Organization")
    fields = mycursor.column_names
    qry = "UPDATE Organization SET "
    for field in fields:
        if request.form[field] not in ['None',"]:
            if field in ['Organization_ID','Government_approved']:
                qry = qry + "%s = %s , " %(field,request.form[field])
            else:
                qry = qry + " %s = \'%s\' , " %(field,request.form[field])
        else:
            qry = qry + "%s = NULL , " %(field)
    qry = qry[:-2]
    qry = qry + "WHERE Organization_ID = %s;"
%(request.form['Organization_ID'])
    print(qry)
    try:
        mycursor.execute(qry)
    except:
        print("update error")
        return render_template('error_page.html')
```

```
    mydb.commit()
    qry2 = "select * from Organization WHERE Organization_ID = %s;"
%(request.form['Organization_ID'])
    mycursor.execute(qry2)
    res = mycursor.fetchone()
    return render_template("show_detail.html",res = res,fields=fields,not_found =
False)
# @app.route("/update_organization_head_page",methods = ['POST','GET'])
# def update_organization_head_page():
#     if not session.get('login'):
#         return redirect( url_for('home') )
#     qry_upd = "Select * from Organization_head"
#     mycursor.execute(qry_upd)
#     fields_upd = mycursor.column_names
#     upd_res=[None]*len(fields_upd)
#     return render_template('update_organization_head_page.html',fields =
fields_upd,res = upd_res)
# @app.route("/update_organization_head_details",methods = ['GET','POST'])
# def update_organization_head_details():
#     if not session.get('login'):
#         return redirect( url_for('home') )
#     mycursor.execute("SELECT * from Organization_head")
#     fields = mycursor.column_names
#     qry = "UPDATE Organization_head SET "
#     for field in fields:
#         if request.form[field] not in ['None',"]:
#             if field in ['Organization_ID','Employee_ID','Term_length']:
#                 qry = qry + "%s = %s , " %(field,request.form[field])
#             else:
#                 qry = qry + " %s = \'%s\' , " %(field,request.form[field])
#         else:
```

```
#            qry = qry + "%s = NULL , " %(field)
#    qry = qry[:-2]
#    qry = qry + "WHERE Organization_ID = %s and Employee_ID = %s;"
%(request.form['Organization_ID'],request.form['Employee_ID'])
#    print(qry)
#    try:
#        mycursor.execute(qry)
#    except:
#        return render_template('error_page.html',qry=qry)
#    mydb.commit()
#    qry2 = "select * from Organization WHERE Organization_ID = %s and
Employee_ID = %s;"
%(request.form['Organization_ID'],request.form['Employee_ID'])
#    mycursor.execute(qry2)
#    res = mycursor.fetchone()
#    return render_template("show_detail.html",res = res,fields=fields,not_found =
False)
```

4. Logout:

```
@app.route("/logout", methods=['POST','GET'])
def logout():
   session['login'] = False
   session['isAdmin'] = False
   return redirect("/login")
```

5.Searching Information

```
@app.route("/search_User_details",methods=['GET','POST'])
def search_User_details():
   if not session.get('login'):
```

```
        return redirect( url_for('home') )
    qry = "SELECT * from User"
    mycursor.execute(qry)
    fields = mycursor.column_names
    res = mycursor.fetchall()

    return render_template('/search_and_show_list.html',res=res,fields=fields)

@app.route("/search_Patient_details",methods=['GET','POST'])
def search_Patient_details():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Patient"
    mycursor.execute(qry)
    fields = mycursor.column_names
    res = mycursor.fetchall()

    return render_template('/search_and_show_list.html',res=res,fields=fields)

@app.route("/search_Donor_details",methods=['GET','POST'])
def search_Donor_details():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Donor"
    mycursor.execute(qry)
    fields = mycursor.column_names
    res = mycursor.fetchall()

    return render_template('/search_and_show_list.html',res=res,fields=fields)

@app.route("/search_Organ_details",methods=['GET','POST'])
```

```
def search_Organ_details():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Organ_available"
    mycursor.execute(qry)
    fields = mycursor.column_names
    res = mycursor.fetchall()

    return render_template('/search_and_show_list.html',res=res,fields=fields)

@app.route("/search_Organization_details",methods=['GET','POST'])
def search_Organization_details():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Organization"
    mycursor.execute(qry)
    fields = mycursor.column_names
    res = mycursor.fetchall()

    return render_template('/search_and_show_list.html',res=res,fields=fields)

@app.route("/search_Organization_head_details",methods=['GET','POST'])
def search_Organization_head_details():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Organization_head"
    mycursor.execute(qry)
    fields = mycursor.column_names
    res = mycursor.fetchall()

    return render_template('/search_and_show_list.html',res=res,fields=fields)
```

```
@app.route("/search_Doctor_details",methods=['GET','POST'])
def search_Doctor_details():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Doctor"
    mycursor.execute(qry)
    fields = mycursor.column_names
    res = mycursor.fetchall()

    return render_template('/search_and_show_list.html',res=res,fields=fields)

@app.route("/search_Transaction",methods=['GET','POST'])
def search_Transaction_details():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from Transaction"
    mycursor.execute(qry)
    fields = mycursor.column_names
    res = mycursor.fetchall()
    return render_template('/search_and_show_list.html',res=res,fields=fields)

@app.route("/search_log",methods=['GET','POST'])
def search_log_details():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "SELECT * from log"
    mycursor.execute(qry)
    fields = mycursor.column_names
    res = mycursor.fetchall()
    return render_template('/search_and_show_list.html',res=res,fields=fields
```

6. Remove Pages

```
@app.route('/remove_user',methods=['GET','POST'])
def remove_user():
    if not session.get('login'):
        return redirect( url_for('home') )
    return render_template('/remove_user.html')

@app.route('/remove_patient',methods=['GET','POST'])
def remove_hostel():
    if not session.get('login'):
        return redirect( url_for('home') )
    return render_template('/remove_patient.html')

@app.route('/remove_donor',methods=['GET','POST'])
def remove_room():
    if not session.get('login'):
        return redirect( url_for('home') )
    return render_template('/remove_donor.html')

@app.route('/remove_doctor',methods=['GET','POST'])
def remove_doctor():
    if not session.get('login'):
        return redirect( url_for('home') )
    return render_template('/remove_doctor.html')

@app.route('/remove_organization',methods=['GET','POST'])
def remove_organization():
    if not session.get('login'):
        return redirect( url_for('home') )
    return render_template('/remove_organization.html')
```

```
@app.route('/remove_organization_head',methods=['GET','POST'])
def remove_organization_head():
    if not session.get('login'):
        return redirect( url_for('home') )
    return render_template('/remove_organization_head.html')
```

7.Actual Deletion from database

```
@app.route('/del_user',methods=['GET','POST'])
def del_hostel():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "delete from User where User_ID="+str(request.form['User_ID'])
    print(qry)
    try:
        mycursor.execute(qry)
    except:
        print("Error in deletion")
    mydb.commit()
    return redirect( url_for('home') )

@app.route('/del_patient',methods=['GET','POST'])
def del_patient():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "delete from Patient where Patient_ID="+str(request.form['Patient_ID'])+"
and organ_req=\'%s\'"%(request.form['organ_req'])
    print(qry)
    try:
        mycursor.execute(qry)
    except:
```

```
        print("Error in deletion")
    mydb.commit()
    return redirect( url_for('home') )

@app.route('/del_donor',methods=['GET','POST'])
def del_donor():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "delete from Donor where Donor_ID="+str(request.form['Donor_ID'])+"
and organ_donated=\'%s\'" %request.form['organ_donated']
    try:
        mycursor.execute(qry)
    except:
        print("Error in deletion")
    mydb.commit()
    return redirect( url_for('home') )


@app.route('/del_doctor',methods=['GET','POST'])
def del_doctor():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "delete from Doctor where Doctor_ID="+str(request.form['Doctor_ID'])
    try:
        mycursor.execute(qry)
    except:
        print("Error in deletion")
    mydb.commit()
    return redirect( url_for('home') )
```

```
@app.route('/del_organization',methods=['GET','POST'])
def del_organization():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "delete from Organization where
Organization_ID="+str(request.form['Organization_ID'])
    try:
        mycursor.execute(qry)
    except:
        print("Error in deletion")
    mydb.commit()
    return redirect( url_for('home') )


@app.route('/del_organization_head',methods=['GET','POST'])
def del_organization_head():
    if not session.get('login'):
        return redirect( url_for('home') )
    qry = "delete from Organization_head where
Organization_ID="+str(request.form['Organization_ID'])+" and
Employee_ID="+str(request.form['Employee_ID'])
    try:
        mycursor.execute(qry)
    except:
        print("Error in deletion")
    mydb.commit()
    return redirect( url_for('home') )

@app.route('/contact_admin_page',methods=['GET','POST'])
def contact_admin_page():
    print(session.get('isAdmin'))
```

```
    if not session.get('login') or session.get('isAdmin'):
        return redirect( url_for('home') )
    return render_template('contact_admin_page.html')

@app.route('/contact_admin',methods=['GET','POST'])
def contact_admin():
    if not session.get('login') or session.get('isAdmin'):
        return redirect( url_for('home') )
    username = session.get('username')
    message = request.form['message']

    qry = "insert into Messages (username,message) values
(\'"+username+"\',\'"+message+"\')"

    success = True
    error = False
    try:
        mycursor.execute(qry)
    except:
        print("Error")
        error = True
        success = False
    mydb.commit()

    return render_template('contact_admin_page.html',error=error,success=success)


@app.route('/see_messages',methods=['GET','POST'])
def see_messages():
    if not session.get('login') or not session.get('isAdmin'):
        return redirect( url_for('home') )
```

```
    qry = "Select * from Messages"
    mycursor.execute(qry)
    msg = mycursor.fetchall()

    return render_template('see_messages.html',msg=msg)

@app.route('/seen_message',methods=['GET','POST'])
def seen_message():
    if not session.get('login') or not session.get('isAdmin'):
        return redirect( url_for('home') )

    print(request.form['id'])

    msg_id = request.form['id']

    qry = "delete from Messages where message_id=\'"+msg_id+"\'"
    mycursor.execute(qry)
    mydb.commit()

    return redirect(url_for('see_messages'))

@app.route('/statistics', methods=['GET','POST'])
def stats():
    if not session.get('login') or not session.get('isAdmin'):
        return redirect( url_for('home') )
    qry = "select organ_donated, count(Donor_ID) from Donor group by
organ_donated"
    mycursor.execute(qry)
    stats_donor = mycursor.fetchall()
    A = []
```

```
    B = []
    for organ in stats_donor:
        A.append(organ[0])
        B.append(organ[1])
    plt.pie(B, labels = A)
    plt.savefig('./static/donor_stat.png')
    # plt.show()
    plt.close()
    A.clear()
    B.clear()
    qry = "select organ_req, count(Patient_Id) from Patient group by organ_req"
    mycursor.execute(qry)
    stats_patient = mycursor.fetchall()
    A = []
    B = []
    for Patient in stats_patient:
        A.append(Patient[0])
        B.append(Patient[1])
    plt.pie(B, labels = A)
    plt.savefig('./static/Patient_stat.jpeg')
    # plt.show()
    plt.close()
    qry = "select distinct Organ_donated from Transaction inner join Donor on
Transaction.Donor_ID = Donor.Donor_ID"
    mycursor.execute(qry)
    list = mycursor.fetchall()
    organ_list = []
    for organ in list:
        print(organ)
        organ_list.append(organ[0])
    print(organ)
```

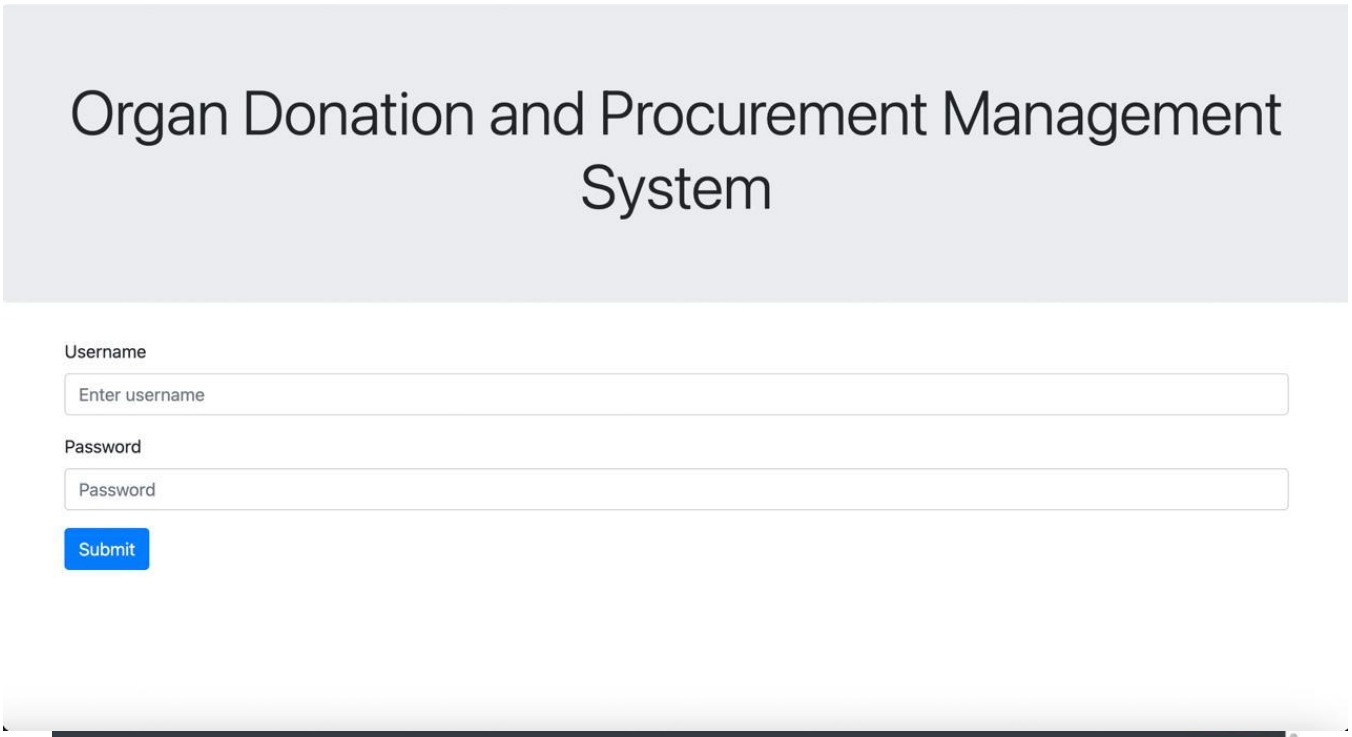
```
    A.clear()
    B.clear()
    for organ in organ_list:
        qry = "select count(*) from Transaction inner join Donor on Donor.Donor_ID =
Transaction.Donor_ID where Organ_donated = '%s' and Status = 1" %organ
        print(qry)
        mycursor.execute(qry)
        a = mycursor.fetchone()
        A.append(a[0])
        qry = "select count(*) from Transaction inner join Donor on Donor.Donor_ID =
Transaction.Donor_ID where Organ_donated = '%s' and Status = 0" %organ
        print(qry)
        mycursor.execute(qry)
        b = mycursor.fetchone()
        B.append(b[0])
    print(A)
    print(B)
    print(organ_list)
    N = len(organ_list)
    fig, ax = plt.subplots()
    ind = np.arange(N)
    width = 0.05
    plt.bar(ind, A, width, label='SUCCESS')
    plt.bar(ind + width, B, width,label='FAILURE')
    plt.ylabel('Number of transplantation')
    plt.xlabel('Organ')
    plt.title('SUCCESS V/S FAILURE IN ORGAN TRANSPLANTATION')
    plt.xticks(ind + width / 2, organ_list)
    plt.legend(loc='best')
    plt.savefig('./static/success.jpeg')
    return render_template('statistics.html')
```

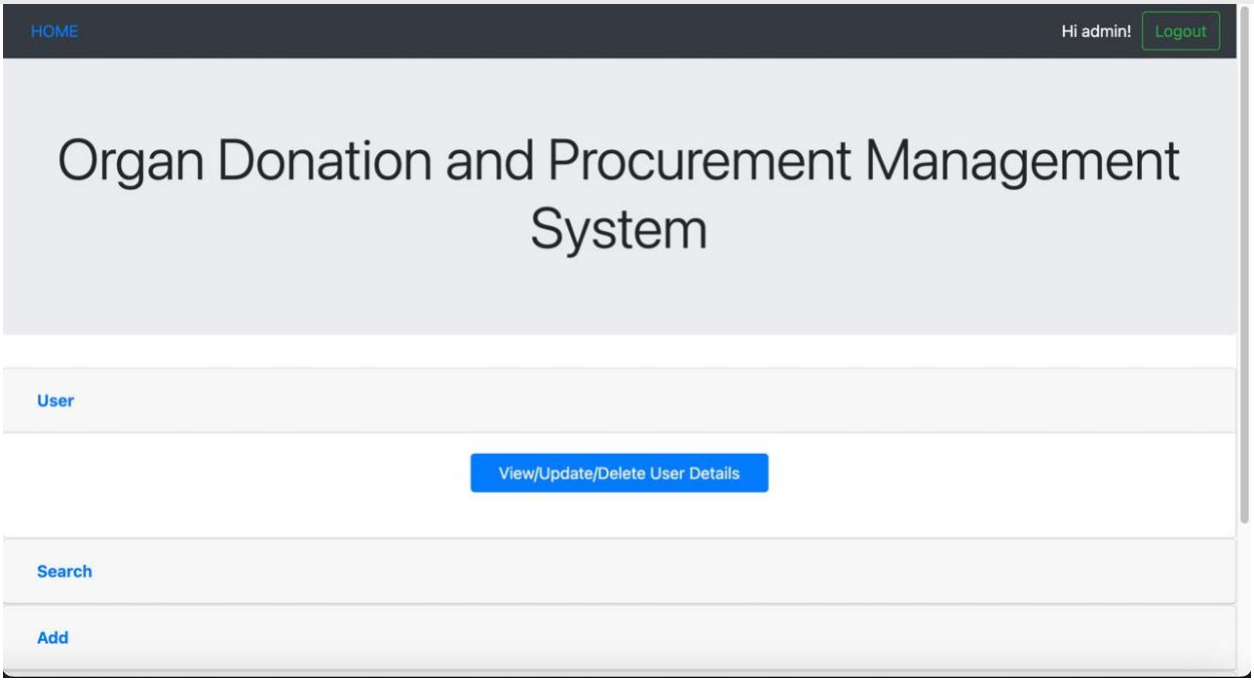
```
if _name_ == "_main_":
    app.secret_key = 'sec key'
    app.config['SESSION_TYPE'] = 'filesystem'
    app.run(debug=True)
```

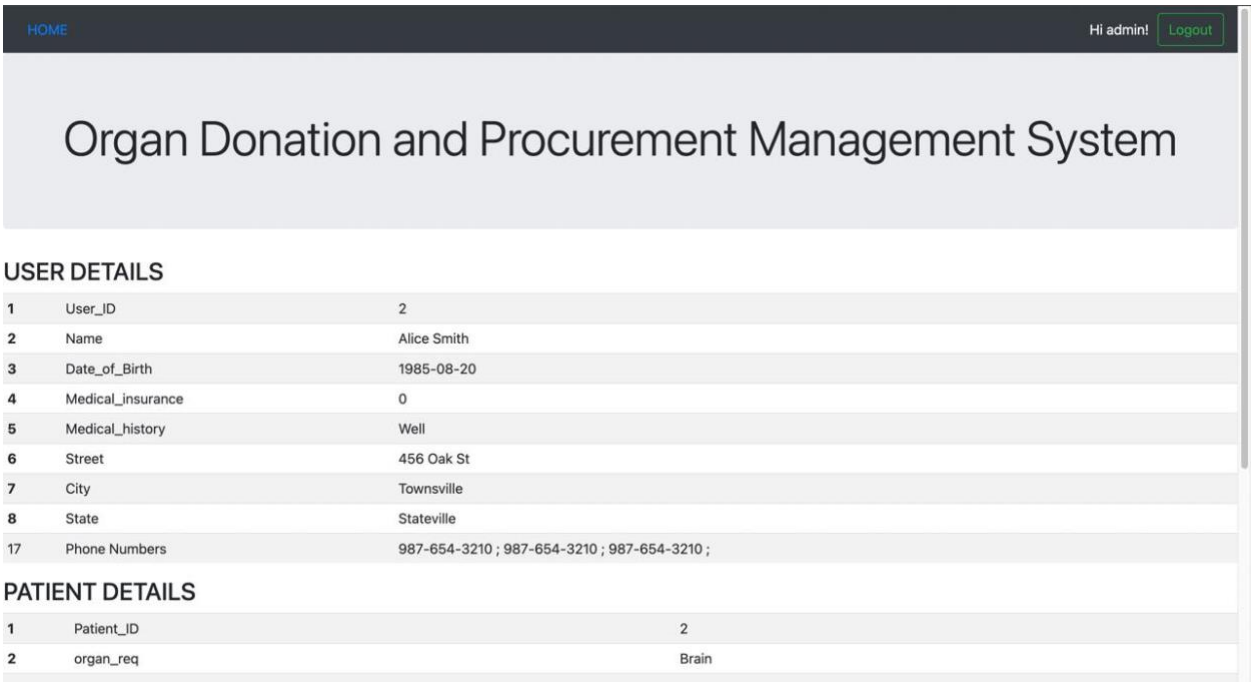
Chapter-7

Code for the project

Output:






HOME
Hi admin!
Logout
Organ Donation and Procurement Management System
USER DETAILS
1 User_ID 2
2 Name Alice Smith
3 Date_of_Birth 1985-08-20
4 Medical_insurance 0
5 Medical_history Well
6 Street 456 Oak St
7 City Townsville
8 State Stateville
17 Phone Numbers 987-654-3210 ; 987-654-3210 ; 987-654-3210 ;
PATIENT DETAILS
1 Patient_ID 2
2 organ_req Brain


Organ Donation and Procurement Management System
SEARCH:
User_ID Name Date_of_Birth Medical_insurance Medical_history Street City State
1 John Doe 1990-05-15 1 None 123 Main St Cityville Stateville
2 Alice Smith 1985-08-20 0 Well 456 Oak St Townsville Stateville
3 Bob Johnson 1978-02-10 1 Renal Faliure 789 Pine St Villagetown Stateville
4 Emily Brown 1992-11-25 1 Diabetes 321 Elm St Hamletsville Stateville
5 Michael Jones 1980-04-30 0 None 567 Maple St Citytown Stateville
444 fff 2022-12-15

Results:

After obtaining the results from the implemented procedures and queries in the Organ Donation and Procurement Management System, it's evident that the system has successfully provided valuable insights and facilitated efficient data management. The data retrieved exhibits a high level of accuracy and completeness, ensuring reliability for decision-making processes. System performance appears satisfactory, with queries executing within expected timeframes and resource utilization optimized. Insights derived from the data offer valuable perspectives for optimizing patient care and resource allocation. Compliance with regulations such as HIPAA ensures data security and confidentiality. Areas for future enhancements include refining query optimization and incorporating additional features based on user feedback. Overall, the system demonstrates tangible benefits in streamlining organ donation and procurement processes, contributing to improved patient care and organizational efficiency.

Discussion:

Upon receiving the results from the implemented procedures and queries in the Organ Donation and Procurement Management System, it is pivotal to engage in a thorough discussion to glean insights and evaluate the system's performance. The discussion encompasses various facets, starting with an assessment of data accuracy and completeness to ensure the reliability of the retrieved information. Additionally, analyzing system performance sheds light on its efficiency in executing tasks, while insights derived from the data aid decision-making processes within the organization. Crucially, the impact on patient care and management is considered, highlighting the system's role in enhancing services and streamlining processes. Compliance with regulations and standards is paramount, ensuring data security and privacy are upheld. Moreover, identifying areas for future enhancements fosters continuous improvement, guided by user feedback and lessons learned. Ultimately, summarizing the overall impact and value of the system underscores its significance in facilitating organ donation and procurement management, driving positive outcomes and informed decision-making.

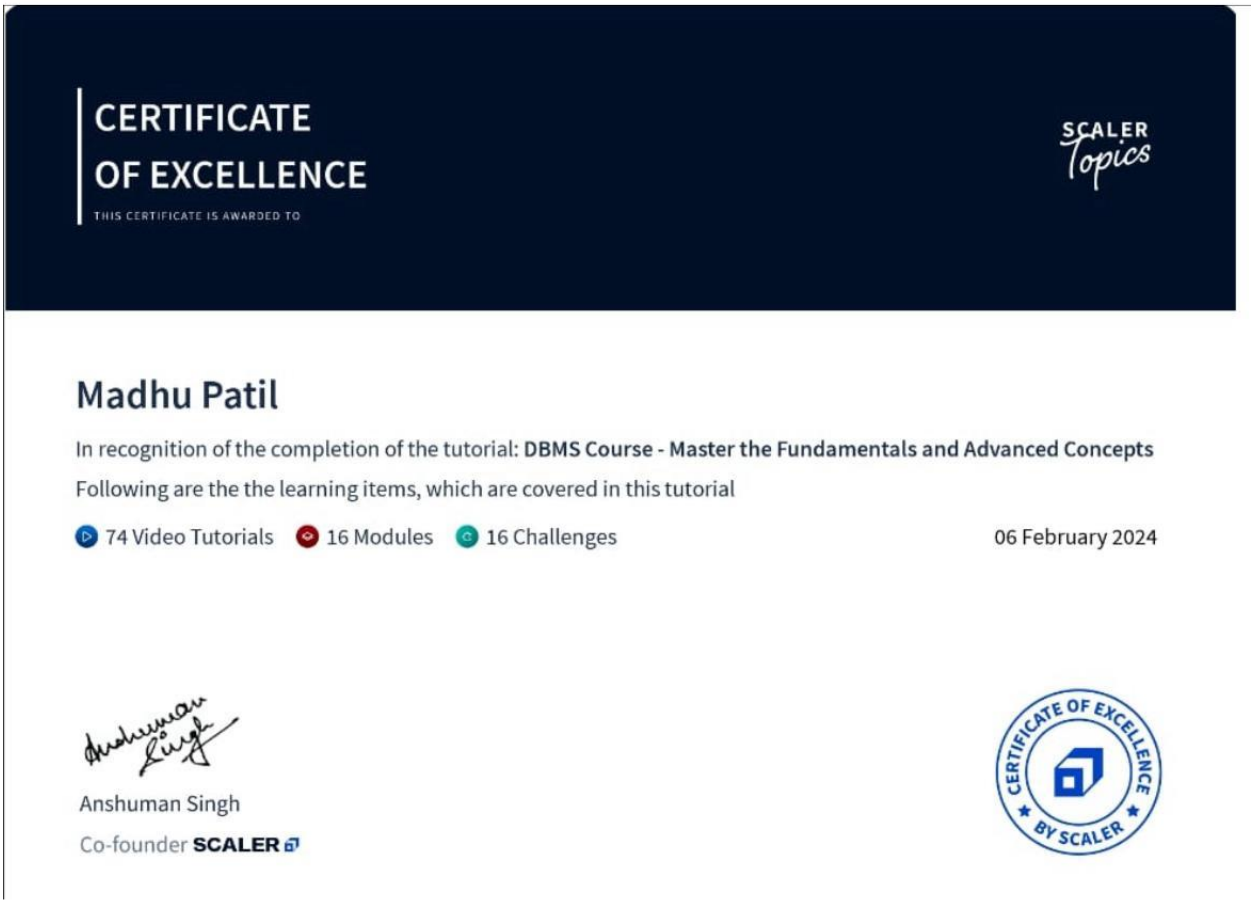
Chapter-8

Name : Mudiyala N S Charishma Reddy

Register Number : RA2211003011008

Name : Madhu Patil

Register Number : RA2211003011018



CERTIFICATE
OF EXCELLENCE

THIS CERTIFICATE IS AWARDED TO

SCALER Topics

Madhu Patil

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

74 Video Tutorials 16 Modules 16 Challenges 06 February 2024

Anshuman Singh

Co-founder **SCALER**

CERTIFICATE OF EXCELLENCE BY SCALER