**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**DATABASE MANAGEMENT SYSTEM - CO202**

****

**SUBMITTED TO SUBMITTED BY**

Mr. Rohit Beniwal Vatsalya Kumar Mishra

(2K21/CO/509)

Vicky Kumar

(2K21/CO/513)

# BLOOD BANK MANAGEMENT SYSTEM

**ABSTRACT**

This project aims to develop a Blood Bank Management System. A Blood Bank Management System can be used in any clinic, hospital, labs or any emergency situation which requires blood units for survival. Our system can be used to find required type of blood in emergency situations from either blood bank or even blood donors.

Current system uses a grapevine communication for finding blood in cases of emergency, may it be by a donor or blood bank. The intentions of proposing such a system are to abolish the panic caused during an emergency due to unavailability of blood.

**INTRODUCTION**

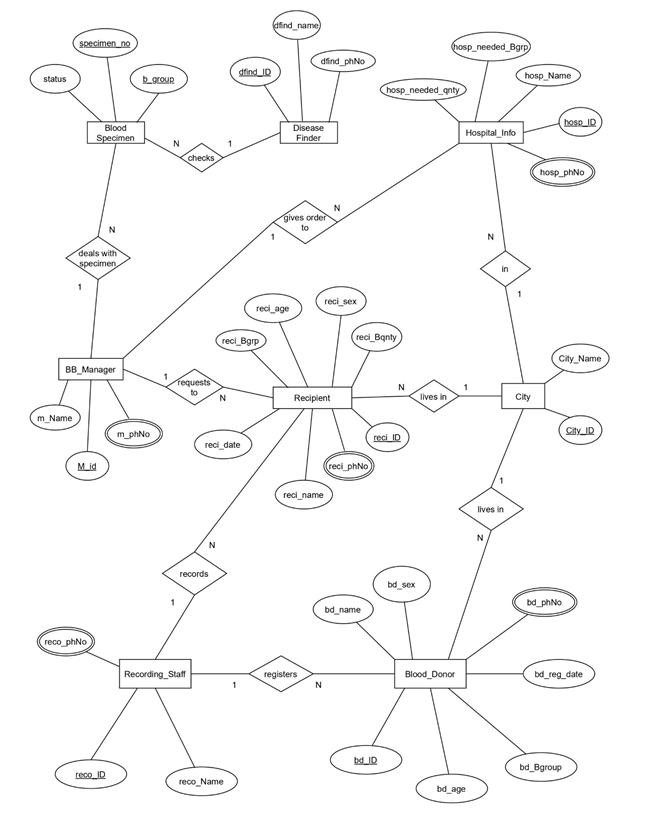
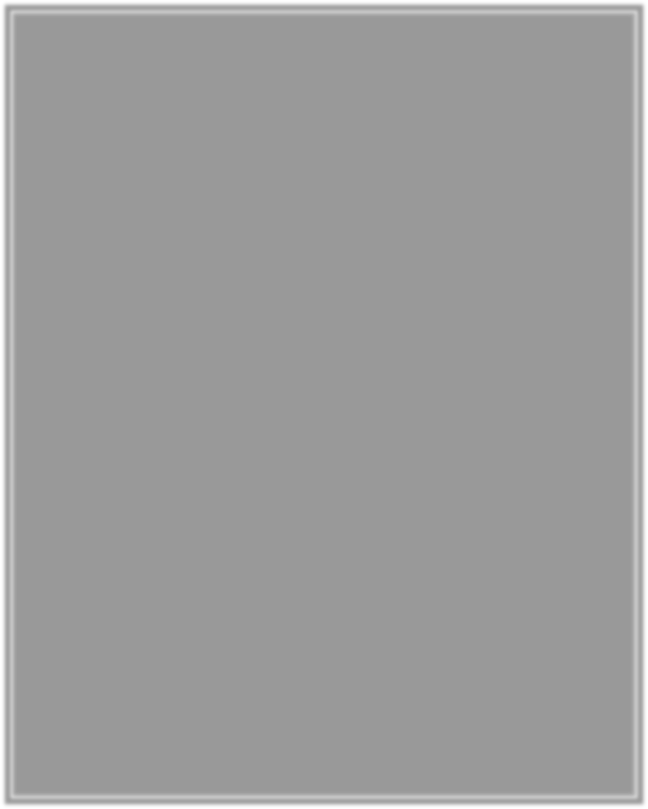
Blood banks collect, store and provide collected blood to the patients who are in need of blood. The people who donate blood are called donors. The banks then group the blood which they receive according to the blood groups. They also make sure that the blood is not contaminated. The main mission of the blood bank is to provide the blood to the hospitals and health care systems which saves the patient’s life. No hospital can maintain the health care system without pure and adequate blood.

The major concern each blood bank has is to monitor the quality of the blood and monitor the people who donates the blood, that is ‘donors’. But this a tough job. The existing system will not satisfy the need of maintaining quality blood and keep track of donors. To overcome all these limitations, we introduced a new system called ‘Blood Donation Management System’.

The ‘Blood Bank Management System’ allows us to keep track of quality of blood and also keeps track of available blood when requested by the acceptor. The existing systems are Manual systems which are time consuming and not so effective. ‘Blood Bank Management system’ automates the distribution of blood. This database consists of thousands of records of each blood bank.

By using this system searching the available blood becomes easy and saves lot of time than the manual system. It will hoard, operate, recover and analyze information concerned with the administrative and inventory management within a blood bank. This system is developed in a manner that it is manageable, time effective, cost effective, flexible and much man power is not required.

**ER DIAGRAM**



**INFORMATION OF ENTITIES**

In total we have eight entities and information of each entity is mentioned below: -

1. **Blood\_Donor:**

#### (Attributes – bd\_ID, bd\_name, bd\_sex, bd\_age, bd\_Bgroup, bd\_reg\_date, bd\_phNo)

The donor is the person who donates blood, on donation a donor id (bd\_ID) is generated and used as primary key to identify the donor information. Other than that name, age, sex, blood group, phone number and registration dates will be stored in database under Blood\_Donor entity.

### Recipient:

#### (Attributes – reci\_ID, reci\_name, reci\_age, reci\_Bgrp, reci\_Bqnty , reci\_sex, reci\_reg\_date, reci\_phNo)

The Recipient is the person who receives blood from blood bank, when blood is given to a recipient a recipient ID (reci\_ID) is generated and used as primary key for the recipient entity to identify blood recipients information. Along with it name ,age, sex, blood group (needed), blood quantity(needed) , phone number, and registration dates are also stored in the data base under recipient entity.

### BB\_Manager:

#### (Attributes – m\_ID, m\_Name, m\_phNo)

The blood bank manager is the person who takes care of the available blood samples in the blood bank, he is also responsible for handling blood requests from recipients and hospitals. Blood manager has a unique identification number (m\_ID) used as primary key along with name and phone number of blood bank manager will be stored in data base under BB\_Manager entity.

### Recording\_Staff :

#### (Attributes – reco\_ID, reco\_Name, reco\_phNo)

The recording staff is a person who registers the blood donor and recipients and the Recording\_Staff entity has reco\_ID which is primary key along with recorder’s name and recorder’s phone number will also be stored in the data base under Recording\_Staff entity.

### BloodSpecimen :

#### (Attributes – specimen\_number, b\_group , status)

In data base, under Blood Specimen entity we will store the information of blood samples which are available in the blood bank. In this entity specimen\_number and b\_group together will be primary key along with status attribute which will show if the blood is contaminated on not.

### DiseaseFinder :

#### (Attributes - dfind\_ID, dfind\_name, dfind\_PhNo)

In data base , under DiseaseFinder entity we will store the information of the doctor who checks the blood for any kind of contaminations. To store that information, we have unique identification number (dfind\_ID) as primary key. Along with name and phone number of the doctor will also be stored under same entity.

### Hospital\_Info :

#### (Attributes – hosp\_ID, hosp\_name, hosp\_needed\_Bgrp, hosp\_needed\_Bqnty)

In the data base, under Hospital\_Info entity we will store the information of hospitals. In this hosp\_ID and hosp\_needed\_Bgrp together makes the primary key. We will store hospital name and the blood quantity required at the hospital.

### City:

#### (Attributes- city\_ID, city\_name)

This entity will store the information of cities where donors, recipients and hospitals are present. A unique identification number (City\_ID) will be used as primary key to identify the information about the city. Along with ID city names will also be stored under this entity.

**RELATIONSHIP BETWEEN ENTITIES**

1. **City and Hospital\_Info:**

Relationship = “in”

Type of relation = 1 to many

Explanation = A city can have many hospitals in it. One hospital will belong in one city.

### City and Blood\_Donor:

Relationship = “lives in”

Type of relation = 1 to many

Explanation = In a city, many donors can live. One donor will belong to one city.

### City and Recipient:

Relationship = “lives in”

Type of relation = 1 to many

Explanation = In a city, many recipients can live. One recipient will belong to one city.

### Recording\_Staff and Donor:

Relationship = “registers”

Type of relation = 1 to many

Explanation = One recording staff can register many donors. One donor will register with one recording officer.

### Recording\_Staff and Recipient:

Relationship = “records”

Type of relation = 1 to many

Explanation = One recording staff can record many recipients. One recipient will be recorded by one recording officer.

### Hospital\_Info and BB\_Manager:

Relationship = “gives order to”

Type of relation = 1 to many

Explanation = One Blood bank manager can handle and process requests from many hospitals. One hospital will place request to on blood bank manager.

### BB\_Manager and Blood Specimen:

Relationship = “deals with specimen”

Type of relation = 1 to many

Explanation = One Blood bank manager can manage many blood specimens and one specimen will be managed by one manager.

### Recipient and BB\_Manager:

Relationship = “requests to”

Type of relation = 1 to many

Explanation = One recipient can request blood to one manager and one manager can handle requests from many recipients.

### Disease\_finder and Blood Specimen:

Relationship = “checks”,

Type of relation = 1 to many

Explanation = A disease finder can check many blood samples. One blood sample is checked by one disease finder.

## RELATIONAL SCHEMAS

### Donor Table:

* The relationship with Recording staff and Donor is 1 to many. That’s

why primary key of Recording staff is used as a foreign key in Donor.

* The relationship with City and Donor is 1 to many. That’s why

primary key of City is used as a foreign key in Donor.

### Recipient Table:

* The relationship with Recording staff and Blood Recipient is 1 to many. That’s why primary key of Recording staff is used as a foreign key in Blood Recipient.
* The relationship with City and Blood Recipient is 1 to many. That’s

why primary key of City is used as a foreign key in Blood Recipient.

* The relationship with Blood Bank Manager and Blood Recipient is 1 to many. That’s why primary key of Blood Specimen is used as a foreign key in Blood Recipient.

### City Table:

* The relationship between City and Recipients, Donor, Hospital info are all of 1 to many. So that’s why primary key of City is used as a foreign key in Recipients, Donor and Hospital info.

### Recording Staff Table:

* The relationship between Recording Staff and Blood Donor, Recipients are all of 1 to many. That’s why the primary key of Recording staff is used as a foreign key in Donor and Recipient.

### Blood Specimen Table:

* The relationship with Disease finder and Blood Specimen is 1 to many. That’s why primary key of Disease finder is used as a foreign key in Blood Specimen.
* The relationship with Blood Bank manager and Blood Specimen is 1 to many. That’s why primary

key of Blood Bank manager is used as a foreign key in Blood Specimen

.

### Disease Finder Table:

* The relationship with Disease finder and Blood Specimen is of 1 to many. Therefore, the primary key of Disease finder is used as a foreign key in Blood Specimen.

### Blood Bank Manager Table:

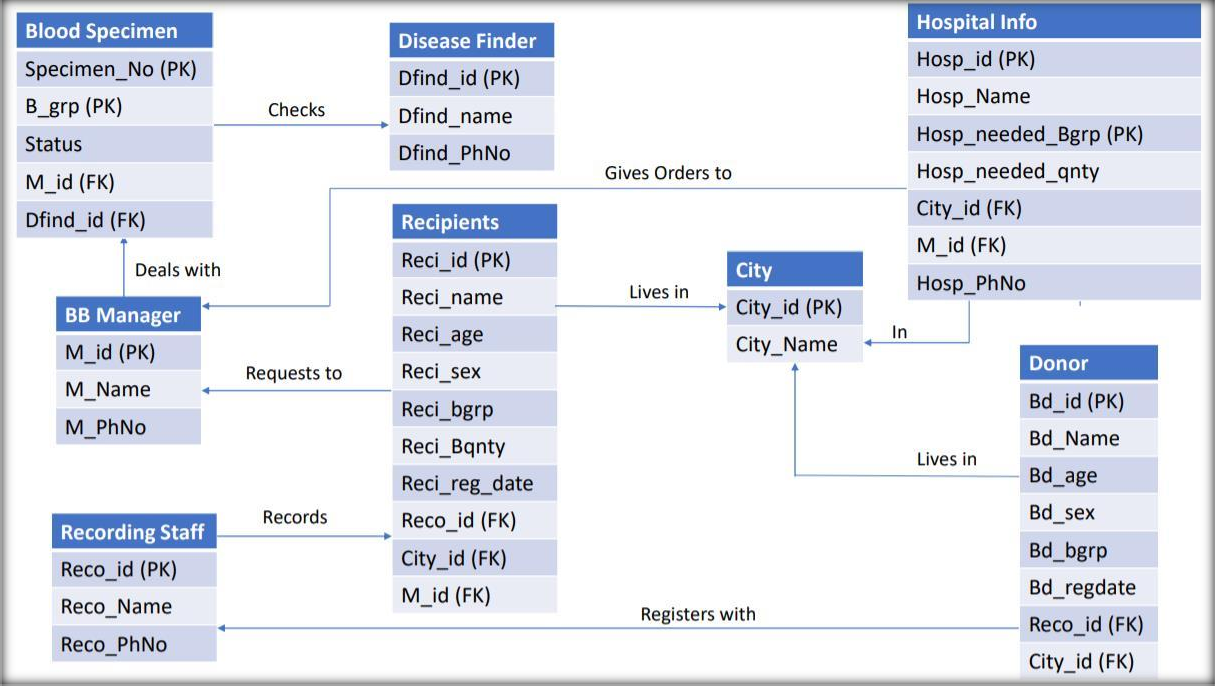
* The relationship between Blood Bank Manager and Blood Specimen, Recipient, Hospital info are all of 1 to many. So therefore, the primary key of Blood Bank Manager is used as a foreign key in Blood Specimen, Recipient and Hospital info.

### Hospital info Table:

* The relationship with City and Hospital info is 1 to many. That’s why

primary key of City is used as a foreign key in Hospital info.

* The relationship with Blood Bank Manager and Hospital info is 1 to many. That’s why primary key of Blood Bank manager is used as a foreign key in Hospital info.



**NORMALIZATION**

**Normalization Rule**

Normalization rules are divided into the following normal forms:

1. First Normal Form
2. Second Normal Form
3. Third Normal Form

### First Normal Form (1NF)

For a table to be in the First Normal Form, it should follow the following 4 rules:

1. It should only have single (atomic) valued attributes/columns.
2. Values stored in a column should be of the same domain.
3. All the columns in a table should have unique names.
4. And the order in which data is stored, does not matter.

### Second Normal Form (2NF)

For a table to be in the Second Normal Form,

1. It should be in the First Normal form.
2. And, it should not have Partial Dependency.

### Third Normal Form (3NF)

A table is said to be in the Third Normal Form when,

1. It is in the Second Normal form.
2. And, it doesn't have Transitive Dependency.

## Normalization of Blood Bank Database:

1. **Blood\_Donor** *(bd\_Id, bd\_name, bd\_phNo bd\_sex, bd\_age, bd\_reg\_date, bd\_Bgroup, reco\_ID, City\_ID)*

{bd\_Id} = > {bd\_name} (functional dependency exists, because two different bd\_name do not correspond to the same bd\_Id).

{bd\_ID} = > {bd\_sex} (functional dependency exists).

{bd\_ID} = > {bd\_age} (functional dependency exists).

{bd\_ID} = > {bd\_reg\_date} date (functional dependency exists).

{bd\_ID} = > {reco\_id} (functional dependency exists).

{bd\_ID} = > {city\_id} (functional dependency exists).

{bd\_ID} = > {bd\_Bgroup} (functional dependency exists).

As the attributes of this table does not have sub attributes, it is in first normal form. Because every non-primary key attribute is fully functionally dependent on the primary key of the table and it is already in first normal form, this table is now in second normal form. Since the table is in second normal form and no non-primary key attribute is transitively dependent on the primary key, the table is now in 3NF.

1. **City** *(city\_id , city\_name)*

{city\_id}= > {city\_name}

The table is in first normal form. The table is in second normal form. The table is in third normal form.

1. **Recording\_staff** *(reco\_name, reco\_ID, reco\_phNo)*

{reco\_id} = > {reco\_name} (functional dependency exists).

{reco\_id} = > {reco\_phNo} (functional dependency exists).

The table is in first normal form. The table is in second normal form. The table is in third normal form.

1. **Blood\_recipient** *(reci\_Id, reci\_sex, reci\_phNo, reci\_age, reci\_date, reci\_name, reci\_Bqnty, reci\_Bgrp, reco\_id, city\_id, m\_id)*

{reci\_Id} = > {reci\_sex} (functional dependency exists).

{reci\_Id} = > {reci\_age} (functional dependency exists).

{reci\_Id} = > {reci\_date} (functional dependency exists).

{reci\_Id} = > {reci\_name} (functional dependency exists).

{reci\_Id} = > {reci\_bqnty} (functional dependency exists).

{reci\_Id} = > {reci\_Bgrp} (functional dependency exists).

{reci\_Id} = > {reco\_id} (functional dependency exists).

{reci\_Id} = > {city\_id} (functional dependency exists).

{reci\_Id} = > {m\_id} (functional dependency exists).

The table is in first normal form. The table is in second normal form. The table is in third normal form.

1. **Blood Specimen** *( b\_group, specimen\_no, status, dfind\_id, m\_id )*

{b\_group, specimen \_no} = > {status} (functional dependency exists).

{b\_group, specimen \_no} = > {dfind \_id} (functional dependency exists).

{b\_group, specimen \_no} = > {m\_id} (functional dependency exists).

The table is in first normal form. The table is in second normal form. The table is in third normal form.

1. **Disease\_finder** *( dfind\_id, dfind\_name, dfind\_PhNo)*

{ dfind\_id } = > { dfind\_name }

{ dfind\_id } = > { dfind\_PhNo } (functional dependency exists).

The table is in first normal form. The table is in second normal form. The table is in third normal form.

1. **BB\_manager** *( M\_id, m\_name, m\_phNo)*

{M\_id} = >{m\_name}

{M\_id} = > {m\_phNo} (functional dependency exists)

The table is in first normal form. The table is in second normal form. The table is in third normal form.

1. **Hospital\_Info** *( hosp\_Id, hosp\_Name, hosp\_phNo, hosp\_needed\_Bgrp, hosp\_needed\_qty, city\_id, m\_id)*

{hosp\_Id}= > {hosp\_Name, hosp\_phNo, city\_id, m\_id}

{hosp\_Id, hosp\_needed\_Bgrp } = > hosp\_needed\_qty (functional dependency exists)

The table is in first normal form.

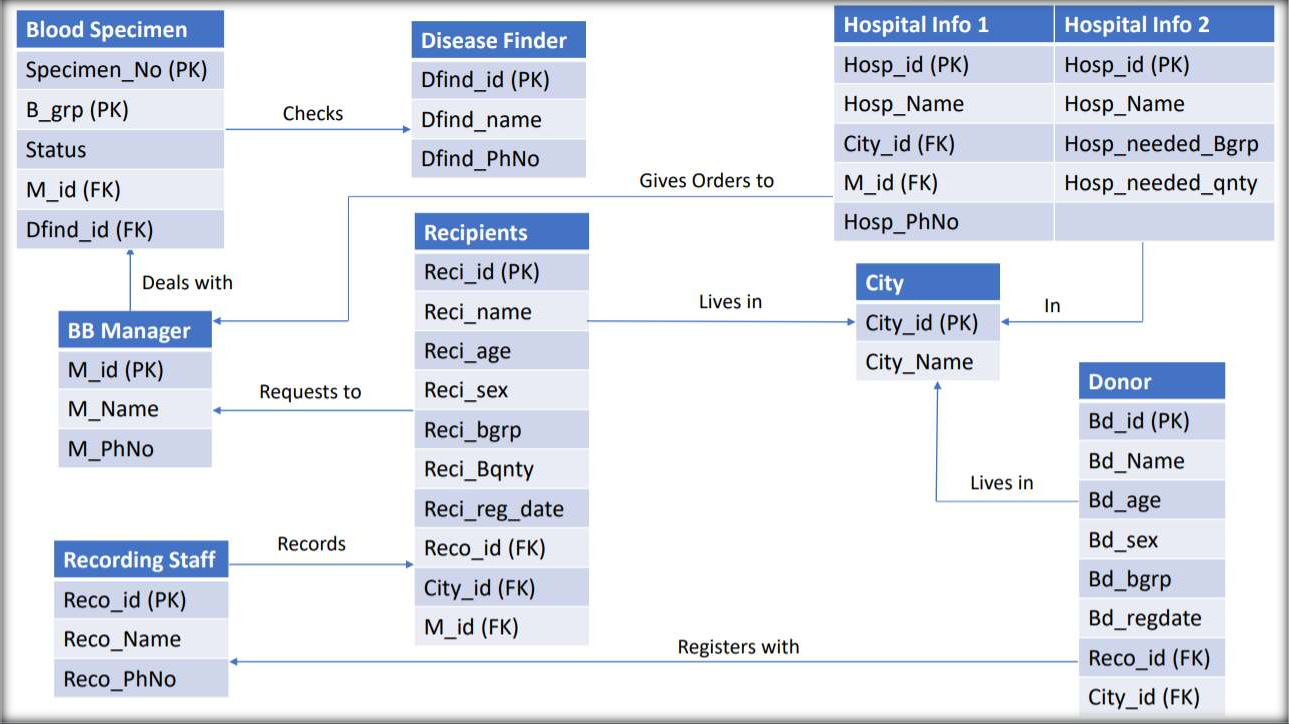
Since every non-primary key attribute is not fully functionally dependent on the primary key of the table, this table is not in second normal form. Hence, we have to split the table.

Hospital\_1 (hosp\_Id, hosp\_phNo, hosp\_Name, city\_id, m\_id).

Hospital\_2 (hosp\_Id, hosp\_needed\_Bgrp, hosp\_needed\_qty)

Now it is in second normal form. The table is in third normal form.

**RELATIONAL SCHEMA AFTER NORMALIZATION**



**SQL IMPLEMENTATION**

*The implementation on SQL Server is given below :-*

-- Creation of 'BB\_Manager' table

-- Creation of 'BB\_Manager' table

CREATE TABLE BB\_Manager (

    M\_id int NOT NULL PRIMARY KEY,

    mName varchar(100) NOT NULL,

    m\_phNo bigint

);

-- Value insertion

INSERT into BB\_Manager VALUES

(101, 'Vatsalya', 9693959671),

(102, 'Vicky', 9693959672),

(103, 'Light', 9693959673),

(104, 'Eren', 9693959674),

(105, 'Mikasa', 9693959675),

(106, 'Goku', 9693959676),

(107, 'Itachi', 9693959677),

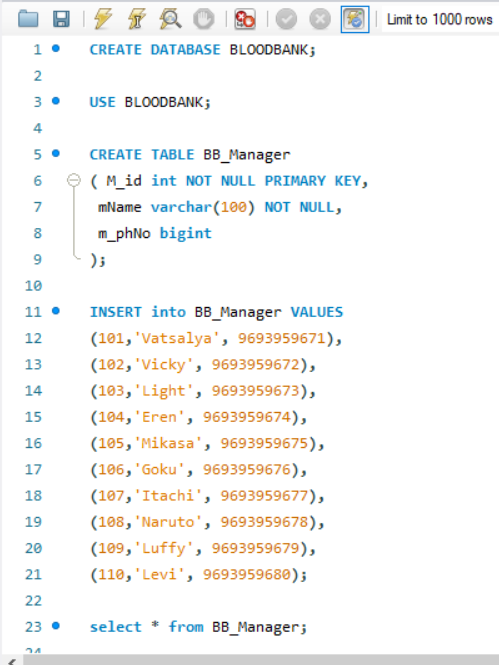
(108, 'Naruto', 9693959678),

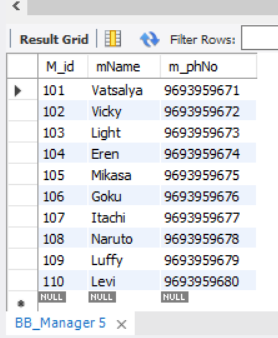
(109, 'Luffy', 9693959679),

(110, 'Levi', 9693959680);

-- Display table

select \* from BB\_Manager;





-- Creation of 'Blood\_Donor' table

-- Creation of 'Blood\_Donor' table

CREATE TABLE Blood\_Donor (

    bd\_ID int NOT NULL PRIMARY KEY,

    bd\_name varchar(100) NOT NULL,

    bd\_age varchar(100),

    bd\_sex varchar(100),

    bd\_Bgroup varchar(10),

    bd\_reg\_date date,

    reco\_ID int NOT NULL,

    City\_ID int NOT NULL,

    FOREIGN KEY(reco\_ID) REFERENCES Recording\_Staff(reco\_ID),

    FOREIGN KEY(City\_ID) REFERENCES City(City\_ID)

);

-- Value insertion

INSERT into Blood\_Donor VALUES

(150011,'Steven',25,'M','O+','2015-07-19',101412,1100),

(150012,'Tony', 35,'M','A-','2015-12-24',101412,1100),

(150013,'Bruce',22,'M','AB+','2015-08-28',101212,1200),

(150014,'Natasha',29,'M','B+','2015-12-17',101212,1300),

(150015,'Hermoine',42,'M','A+','2016-11-22',101212,1300),

(150016,'Harry',44,'F','AB-','2016-02-06',101212,1200),

(150017,'Sherlock',33,'M','B-','2016-10-15',101312,1400),

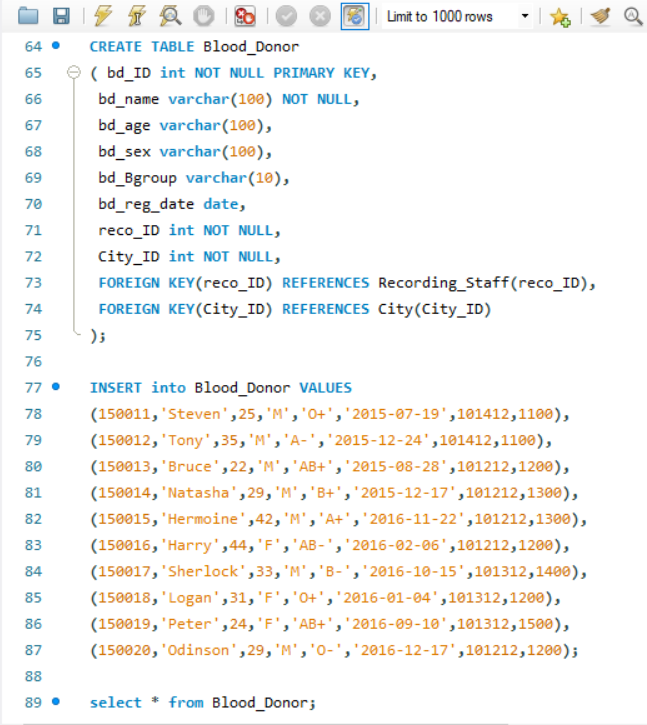
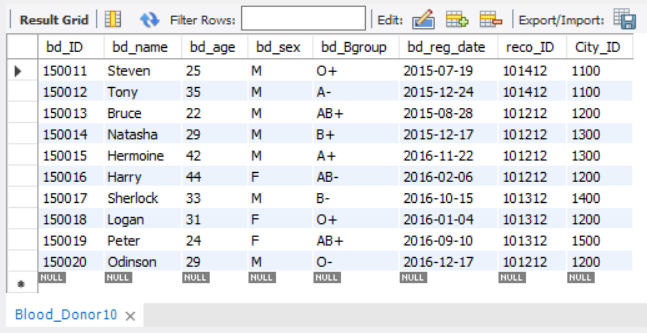
(150018,'Logan',31,'F','O+','2016-01-04',101312,1200),

(150019,'Peter',24,'F','AB+','2016-09-10',101312,1500),

(150020,'Odinson',29,'M','O-','2016-12-17',101212,1200);

-- Display table

select \* from Blood\_Donor;



-- Creation of 'BloodSpecimen' table

-- Creation of 'BloodSpecimen' table

CREATE TABLE BloodSpecimen (

    specimen\_number int NOT NULL,

    b\_group varchar(10) NOT NULL,

    status int,

    dfind\_ID int NOT NULL,

    M\_id int NOT NULL,

    primary key (specimen\_number, b\_group),

    FOREIGN KEY(M\_id) REFERENCES BB\_Manager(M\_id),

    FOREIGN KEY(dfind\_ID) REFERENCES DiseaseFinder(dfind\_ID)

);

-- Value insertion

INSERT into BloodSpecimen

VALUES (1001, 'B+', 1, 11, 101),

    (1002, 'O+', 1, 12, 102),

    (1003, 'AB+', 1, 11, 102),

    (1004, 'O-', 1, 13, 103),

    (1005, 'A+', 0, 14, 101),

    (1006, 'A-', 1, 13, 104),

    (1007, 'AB-', 1, 15, 104),

    (1008, 'AB-', 0, 11, 105),

    (1009, 'B+', 1, 13, 105),

    (1010, 'O+', 0, 12, 105),

    (1011, 'O+', 1, 13, 103),

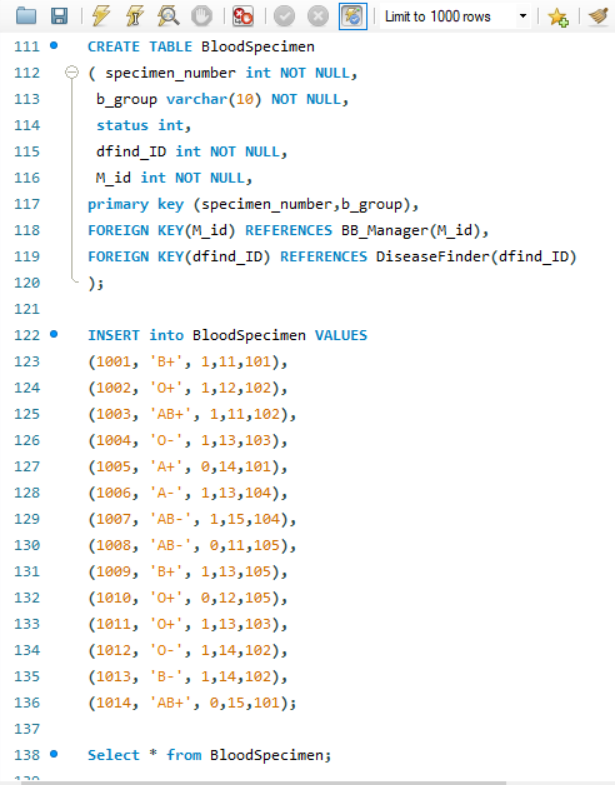
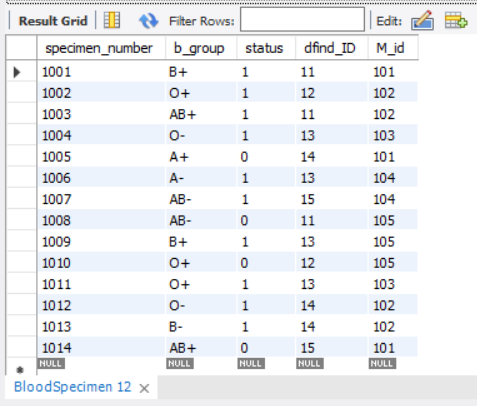
    (1012, 'O-', 1, 14, 102),

    (1013, 'B-', 1, 14, 102),

    (1014, 'AB+', 0, 15, 101);

-- Display table

Select \* from BloodSpecimen;



-- Creation of 'City' table

-- Creation of 'City' table

CREATE TABLE City (

    City\_ID int NOT NULL PRIMARY KEY,

    City\_name varchar(100) NOT NULL

);

-- Value insertion

INSERT into City

VALUES (1100, 'Asgard'),

    (1200, 'Paradis'),

    (1300, 'Marley'),

    (1400, 'Wakanda'),

    (1500, 'Valhalla'),

    (1600, 'Madripoor'),

    (1700, 'Hogwarts'),

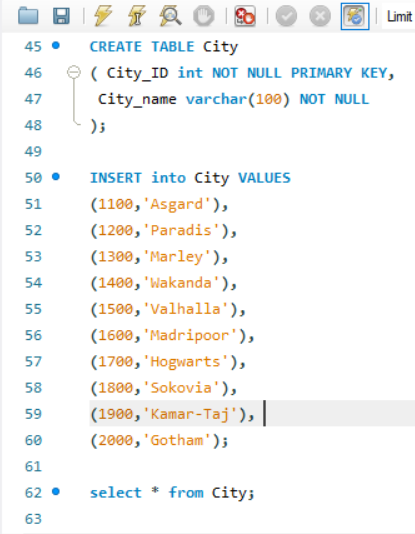
    (1800, 'Sokovia'),

    (1900, 'Kamar-Taj'),

    (2000, 'Gotham');

-- Display table

select \* from City;

-- Creation of 'DiseaseFinder' table

-- Creation of 'DiseaseFinder' table

CREATE TABLE DiseaseFinder (

    dfind\_ID int NOT NULL PRIMARY KEY,

    dfind\_name varchar(100) NOT NULL,

    dfind\_PhNo bigint

);

-- Value insertion

INSERT into DiseaseFinder

VALUES (11, 'Indiana', 9693959681),

    (12, 'Stephen', 9693959682),

    (13, 'Christine', 9693959683),

    (14, 'Gwen', 9693959672),

    (15, 'Viktor', 9693959679),

    (16, 'Skywalker', 9693959684),

    (17, 'Julius', 9693959685),

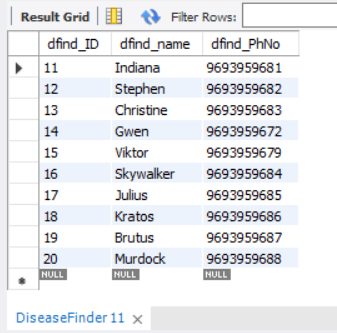
    (18, 'Kratos', 9693959686),

    (19, 'Brutus', 9693959687),

    (20, 'Murdock', 9693959688);

-- Display table

select \* from DiseaseFinder;

-- Creation of 'Hospital\_Info\_1' table

-- Creation of 'Hospital\_Info\_1' table

CREATE TABLE Hospital\_Info\_1 (

    hosp\_ID int NOT NULL,

    hosp\_name varchar(100) NOT NULL,

    City\_ID int NOT NULL,

    M\_id int NOT NULL,

    primary key(hosp\_ID),

    FOREIGN KEY(M\_id) REFERENCES BB\_Manager(M\_id),

    FOREIGN KEY(City\_ID) REFERENCES City(City\_ID)

);

-- Value insertion

INSERT into Hospital\_Info\_1

VALUES (1, 'Springfield', 1100, 101),

    (2, 'Hampshire', 1200, 103),

    (3, 'Winterfell', 1300, 103),

    (4, 'Riverrun', 1400, 104),

    (5, 'Hogsmeade', 1800, 103),

    (6, 'Greenoaks', 1300, 106),

    (7, 'Forestpark', 1300, 102),

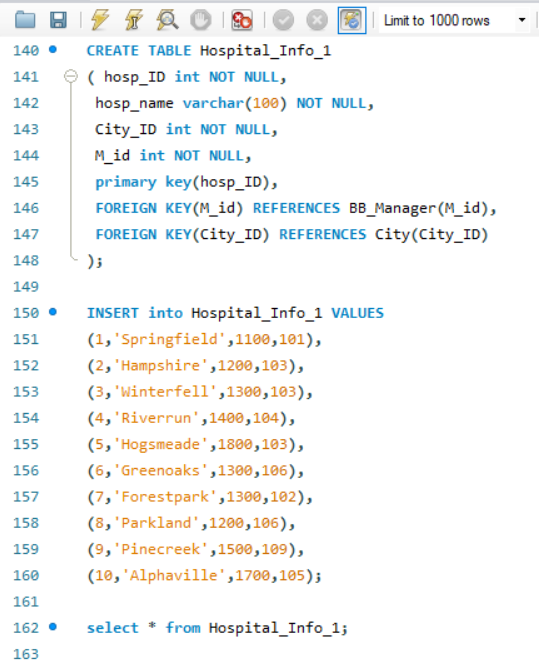
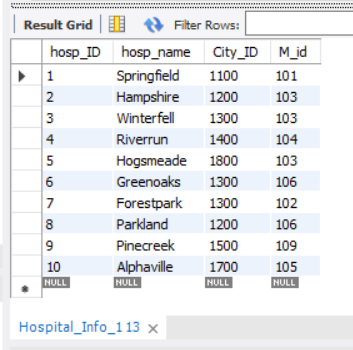
    (8, 'Parkland', 1200, 106),

    (9, 'Pinecreek', 1500, 109),

    (10, 'Alphaville', 1700, 105);

-- Display table

select \* from Hospital\_Info\_1;



-- Creation of 'Hospital\_Info\_2' table

-- Creation of 'Hospital\_Info\_2' table

CREATE TABLE Hospital\_Info\_2 (

    hosp\_ID int NOT NULL,

    hosp\_name varchar(100) NOT NULL,

    hosp\_needed\_Bgrp varchar(10),

    hosp\_needed\_qnty int,

    primary key(hosp\_ID, hosp\_needed\_Bgrp)

);

-- Value insertion

INSERT into Hospital\_Info\_2

VALUES (1, 'Springfield', 'A+', 20),

    (1, 'Springfield', 'A-', 0),

    (1, 'Springfield', 'AB+', 40),

    (1, 'Springfield', 'AB-', 10),

    (1, 'Springfield', 'B-', 20),

    (2, 'Hampshire', 'A+', 40),

    (2, 'Hampshire', 'AB+', 20),

    (2, 'Hampshire', 'A-', 10),

    (2, 'Hampshire', 'B-', 30),

    (2, 'Hampshire', 'B+', 0),

    (2, 'Hampshire', 'AB-', 10),

    (3, 'Winterfell', 'A+', 0),

    (3, 'Winterfell', 'AB+', 0),

    (3, 'Winterfell', 'A-', 0),

    (3, 'Winterfell', 'B-', 20),

    (3, 'Winterfell', 'B+', 10),

    (3, 'Winterfell', 'AB-', 0),

    (4, 'Riverrun', 'A+', 10),

    (4, 'Riverrun', 'A-', 40),

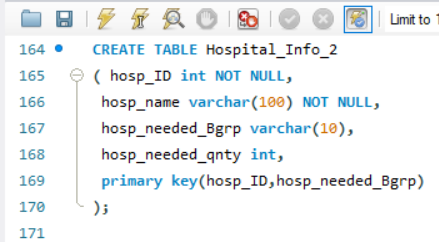
    (7, 'Forestpark', 'B-', 40),

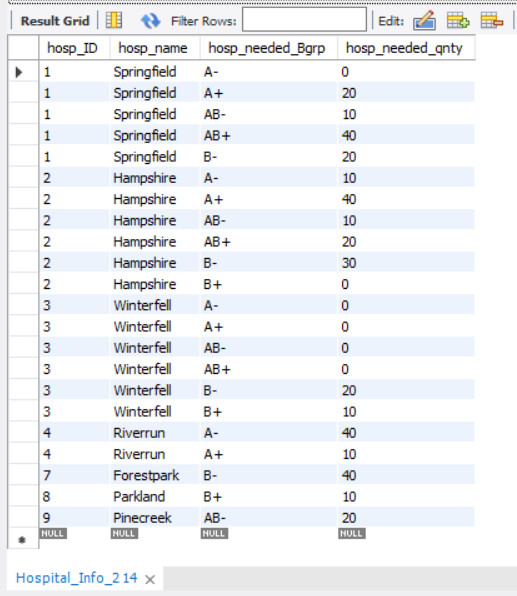
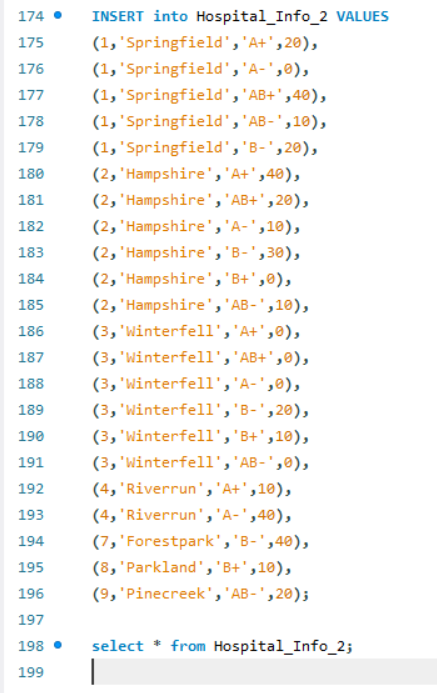
    (8, 'Parkland', 'B+', 10),

    (9, 'Pinecreek', 'AB-', 20);

-- Display table

select \* from Hospital\_Info\_2;





-- Creation of 'Recipient' table

-- Creation of 'Recipient' table

CREATE TABLE Recipient (

    reci\_ID int NOT NULL PRIMARY KEY,

    reci\_name varchar(100) NOT NULL,

    reci\_age varchar(10),

    reci\_Brgp varchar(100),

    reci\_Bqnty float,

    reco\_ID int NOT NULL,

    City\_ID int NOT NULL,

    M\_id int NOT NULL,

    FOREIGN KEY(M\_id) REFERENCES BB\_Manager(M\_id),

    FOREIGN KEY(City\_ID) REFERENCES City(City\_ID)

);

Alter table Recipient

ADD reci\_sex varchar(100);

Alter table Recipient

ADD reci\_reg\_date date;

-- Value insertion

INSERT into Recipient VALUES

(10001,'Indiana',25,'B+',1.5,101212,1100,101,'F','2015-12-17'),

(10002,'Bruce',60,'A+',1,101312,1100,102,'M','2015-12-16'),

(10003,'Goku',35,'AB+',0.5,101312,1200,102,'M','2015-10-17'),

(10004,'Stephen',66,'B+',1,101212,1300,104,'M','2016-11-17'),

(10005,'Itachi',53,'B-',1,101412,1400,105,'M','2015-04-17'),

(10006,'Erwin',45,'O+',1.5,101512,1500,105,'M','2015-12-17'),

(10007,'Natasha',22,'AB-',1,101212,1500,101,'M','2015-05-17'),

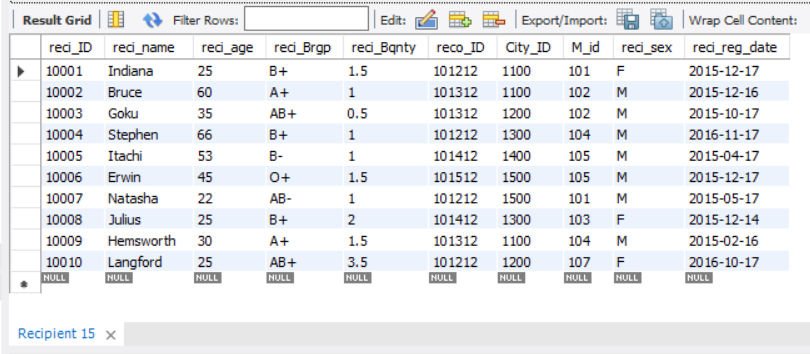
(10008,'Julius',25,'B+',2,101412,1300,103,'F','2015-12-14'),

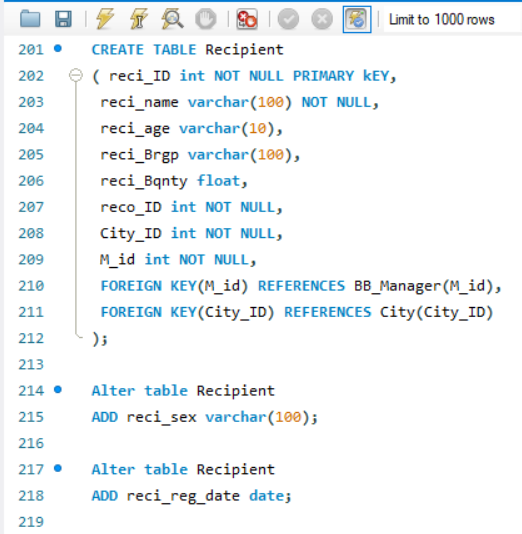
(10009,'Hemsworth',30,'A+',1.5,101312,1100,104,'M','2015-02-16'),

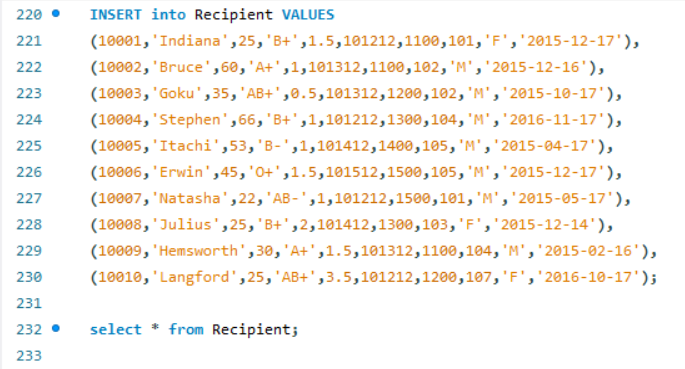
(10010,'Langford',25,'AB+',3.5,101212,1200,107,'F','2016-10-17');

-- Display table

select \* from Recipient;







-- Creation of 'Recording\_Staff' table

-- Creation of 'Recording\_Staff' table

CREATE TABLE Recording\_Staff (

    reco\_ID int NOT NULL PRIMARY KEY,

    reco\_Name varchar(100) NOT NULL,

    reco\_phNo bigint

);

-- Value insertion

INSERT into Recording\_Staff

VALUES (101012, 'Tanjiro', 4044846553),

    (101112, 'Zenitsu', 4045856553),

    (101212, 'Inosuke', 4045806553),

    (101312, 'Mitsuri', 4045806553),

    (101412, 'Nezuko', 4045806553),

    (101512, 'Muzan', 4045806553),

    (101612, 'Akaza', 4045806553),

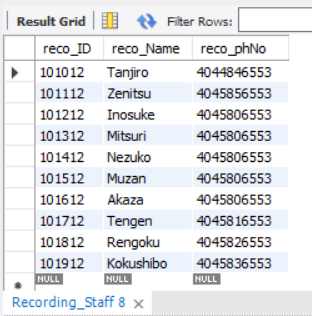
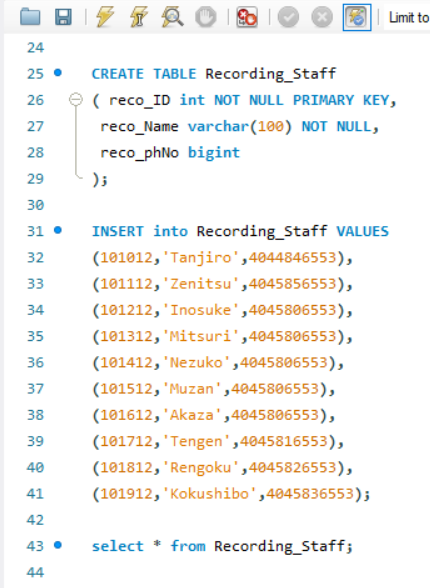
    (101712, 'Tengen', 4045816553),

    (101812, 'Rengoku', 4045826553),

    (101912, 'Kokushibo', 4045836553);

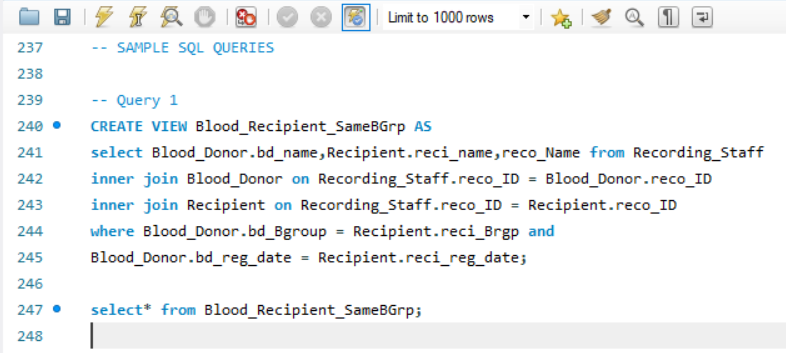
-- Display table

select \* from Recording\_Staff;

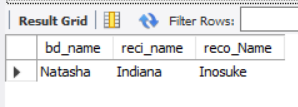


# SAMPLE SQL QUERIES

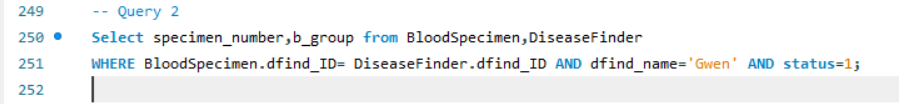
### Create a View of recipients and donors’ names having the same blood group registered on the same date and the name of recording staff name.



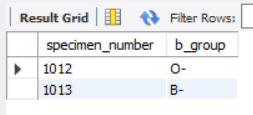
**Output:**

****

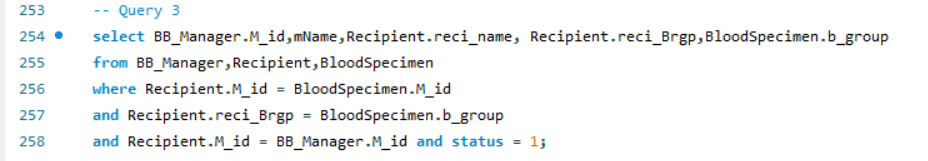
* 1. **Show the blood specimen verified by disease finder Gwen which are pure (status=1).**



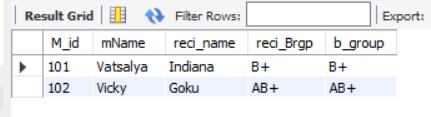
**Output:**

****

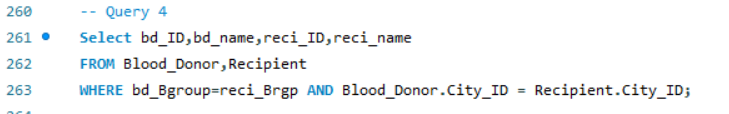
* 1. **Show the pure blood specimen handled by BB\_Manager who also handles a recipient needing the same blood group along with the details of the BB\_Manager and Recipient.**



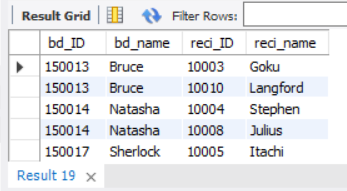
**Output:**

****

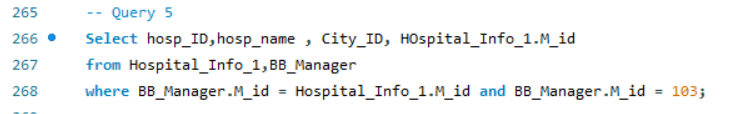
* 1. **Show the donors having the same blood groups required by the recipient staying in the same city along with recipient details.**



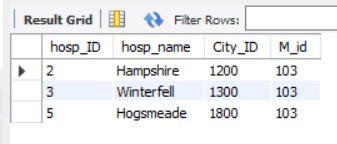
**Output:**

****

* 1. **Display the information of Hospital\_Info\_1 handled by BB\_Manager whose ID is 103:**



**Output:**

****

# CONCLUSION

Prior to this project, a general study of blood bank management system was conducted from recent researches of various authors and facts were gathered in which helped to uncover the misfits that the system was facing.

After proper analyzation of these problems, a solution was then developed in order to meet up the needs of a more advanced system. This system is known as the centralized blood bank repository which helped in eliminating all the problems that the previous systems were facing. With this system, Blood banks/ Centers, Hospitals, Patients and Blood donors will be brought together to enjoy a large number of functionalities and access a vast amount of information, thereby making blood donation and reception a lot easier and faster.

Before implementing the database, in the design phase, we have explored various features, operations of a blood bank to figure out required entities, attributes and the relationship among entities to make an efficient Entity Relationship Diagram (ERD). After analyzing all the requirements, I have created

our ERD and then converted the ERD to relational model and normalized the tables.

Using SQL Server, I have created the tables for my database and inserted some sample values in the tables. Finally, I have executed sample queries on the database to check its performance to retrieve useful information accurately and speedily.