BLOOD BANK MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

NAME OF THE CANDIDATE

Dimpal Tyagi(24MCA20021)

In partial fulfilment for the award of the degree of

MASTERS OF COMPUTER APPLICATION

Chandigarh University
OCTOBER 2024

INDEX

INTRODUCTION

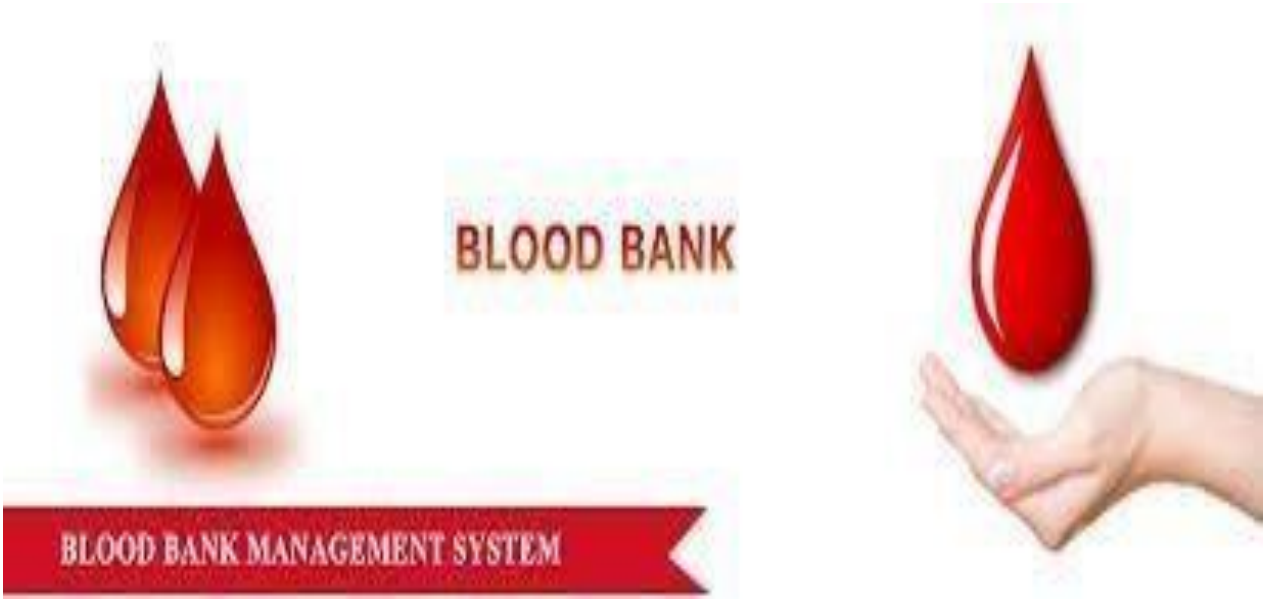
Blood banks collect, store and provide collected blood to the patients who are in need of blood. The people who donate blood are called 'donors'. The banks then group the blood which they receive according to the blood groups. They also make sure that the blood is not contaminated.

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.[4]



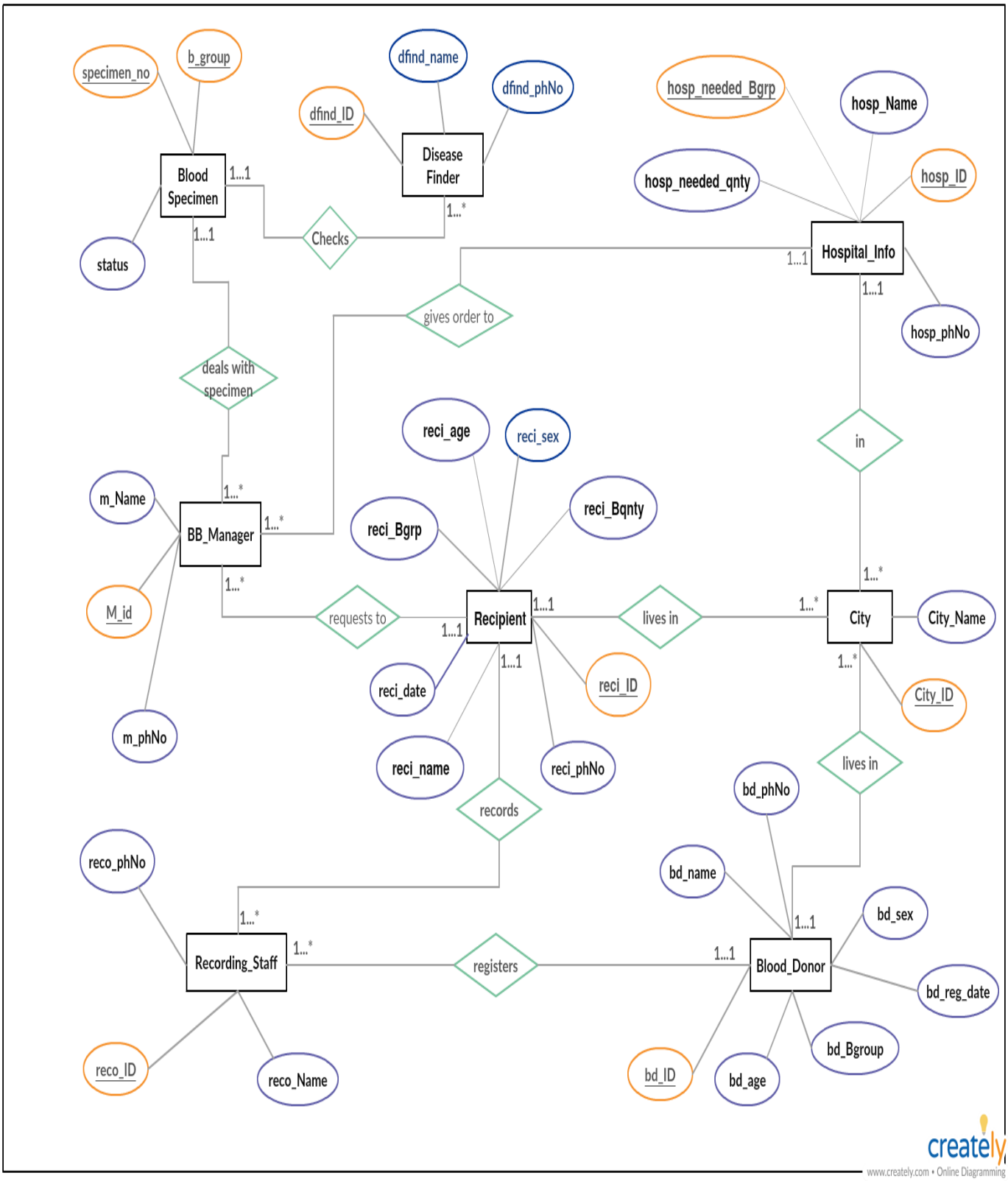
LITERATURE WORK

'Blood Donation Management system' is similar to 'Organ Donation Management System' and 'Charity Management System' databases.

'Organ donation management system' is a database that automates the organ donation to the patients who are in need of it. Each hospital maintains the 'Organ Donation Management System'. This system keeps track of organs available in the hospital to donate them to the patients. This system also contains donors and acceptors. So this system is similar to 'Blood Bank Management System'. [1]

'Charity Management System' keeps track of the donation of money to different organizations which are in need of it. Here in this system, Donor is a person who donates money and acceptor is an organization who requests for money or to whom the money is donated. This database contains thousands of records to whom the money is donated by a person. This system is similar to 'Blood Bank Management System' because the functioning is very similar when compared. [2]

DIAGRAM


specimen_no
b_group
dfind_name
dfind_phNo
dfind_ID
Disease Finder
Blood Specimen
1...1
Checks
1...*
status
hosp_needed_Bgrp
hosp_Name
hosp_needed_qnty
hosp_ID
Hospital_Info
gives order to
hosp_phNo
deals with specimen
reci_age
reci_sex
in
m_Name
BB_Manager
reci_Bgrp
reci_Bqnty
M_id
requests to
Recipient
lives in
City
City_Name
reci_date
reci_ID
City_ID
m_phNo
reci_name
reci_phNo
records
lives in
bd_phNo
reco_phNo
bd_name
bd_sex
Recording_Staff
registers
Blood_Donor
bd_reg_date
reco_ID
reco_Name
bd_ID
bd_age
bd_Bgroup
creately
www.creately.com • Online Diagramming

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.

2. **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 recivies blood from blood bank, when blood is given to a recipient a rericipient ID (reci_ID) is generated and used as primary key for the recipient entity to indentify 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.

3. **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 enitity has reco_ID which is primary key along with recoder's name and recodrer's phone number will also be stored in the data base under Recording_Staff entity.

4. **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 toether makes the primary key. We will store hospital name and the blood quantity reqiured at the hospital.

RELATIONAL SCHEMAS

Donor Table:

Attribute Name	Description	Type
bd_id	Blood Donor's Id	int
bd_Name	Blood Donor's Name	varchar
bd_age	Blood Donor's Age	int
bd_sex	Blood Donor's Sex	char
bd_bgrp	Blood Donor's blood group	varchar
bd_regdate	Registration Date of Donor	date

Recipient Table:

Attributes Name	Description	Type
reci_id	Recipient's Id	int
reci_Name	Recipient's Name	varchar
reci_age	Recipient's age	int
reci_sex	Recipient's sex	char
reci_bgrp	Recipient's blood group	varchar
reci_bqnty	Recipient's blood quantity	int
reci_reg_date	Recipient's registration date	date

Blood Specimen Table:

Attributes Name	Description	Type
Specimen_No	Blood Sample's unique id	int
b_grp	Blood Group	varchar
status	Whether blood is pure or not?	int
M_id	Blood Bank Manager's id	int
Dfind_id	Disease Finder's unique id	int

- 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:

Attribute Name	Description	Type
dfind_id	Disease Finder's unique id	int
dfind_name	Disease Finder's name	varchar
dfind_phNo	Disease Finder's phone number	bigint

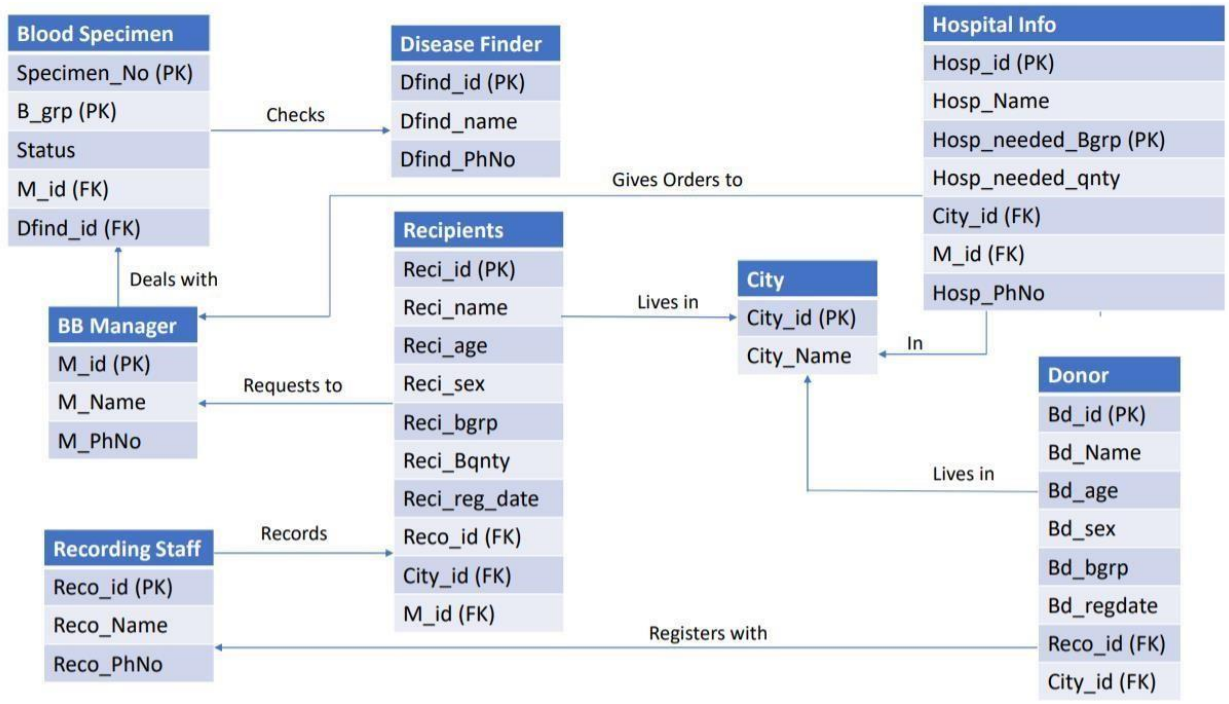
- The relationship with Disease finder and Blood Specimen is of 1 to many. Therefore, the primary key of Disease finder is used as a foreign key in Blood Specimen.

Hospital info Table:

Attributes Name	Description	Type
hosp_id	Hospital's unique id	int
hosp_name	Hospital's name	varchar
hosp_needed_Bgrp	Blood group needed by hospital	varchar
hosp_needed_qnty	Quantity of blood group needed	int
city_id	City's unique id	int

- 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.

DIAGRAM WITH TABLES



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. [3][6]

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.

2. **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.

3. **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.

4. **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.

5. **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.

6. **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.

7. **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.

8. **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.

TABLES AFTER NORMALIZATION

BB_Manager:

Results | Messages

	M_id	mName	m_phNo
1	101	Jack	4693951392
2	102	Mark	4693804553
3	103	Dan	4693804552
4	104	Stacy	4693804551
5	105	Henry	4693804550
6	106	Steve	4694959671
7	107	Jason	4695959671
8	108	Stella	4663959671
9	109	Monika	4673959671
10	110	John	4693859671

Blood_Donor:

Results | Messages

	bd_ID	bd_name	bd_age	bd_sex	bd_Bgroup	bd_reg_date	reco_ID	City_ID	bd_phNo
1	150011	Pat	29	M	O+	2015-07-19	101412	1300	4693951232
2	150021	Shyam	42	F	A-	2015-12-24	101412	1300	4600001232
3	150121	Dan	44	M	AB+	2015-08-28	101212	1200	4611111232
4	150221	Mark	25	M	B+	2015-12-17	101212	1100	4622221232
5	160011	Abdul	35	F	A+	2016-11-22	101212	1100	4633331232
6	160031	Mike	33	F	AB-	2016-02-06	101212	1400	4644441232
7	160091	Carrol	24	M	B-	2016-10-15	101312	1500	4655551232
8	160101	Smith	22	M	O+	2016-01-04	101312	1200	4666661232
9	160301	Elisa	31	F	AB+	2016-09-10	101312	1200	4677771232
10	160401	Mark	29	M	O-	2016-12-17	101212	1200	4688881232

BloodSpecimen:

Results | Messages

	specimen_number	b_group	status	dfind_ID	M_id
1	1001	B+	1	11	101
2	1002	O+	1	12	102
3	1003	AB+	1	11	102
4	1004	O-	1	13	103
5	1005	A+	0	14	101
6	1006	A-	1	13	104
7	1007	AB-	1	15	104
8	1008	AB-	0	11	105
9	1009	B+	1	13	105
10	1010	O+	0	12	105
11	1011	O+	1	13	103
12	1012	O-	1	14	102
13	1013	B-	1	14	102
14	1014	AB+	0	15	101

City:

Results | Messages

	City_ID	City_name
1	1100	Dallas
2	1200	Austin
3	1300	Irving
4	1400	Houston
5	1500	Richardson
6	1600	Plano
7	1700	Frisco
8	1800	Arlington
9	1900	San Antonio
10	2000	Allen

DiseaseFinder:

Results | Messages

	dfind_ID	dfind_name	dfind_PhNo
1	11	Peter	4693804223
2	12	Park	4693804223
3	13	Jerry	4693804223
4	14	Mark	4693804223
5	15	Monika	4693804223
6	16	Ram	4693804123
7	17	Swathi	4693804223
8	18	Gautham	4693804323
9	19	Ashwin	4693804423
10	20	Yash	4693804523

Hospital_Info_1:

Results Messages

	hosp_ID	hosp_name	City_ID	M_id	hosp_phNo
1	1	MayoClinic	1100	101	4611001232
2	2	CleavelandClinic	1200	103	4622001232
3	3	NYU	1300	103	4633001232
4	4	Baylor	1400	104	4644001232
5	5	Charlton	1800	103	4655001232
6	6	Greenoaks	1300	106	4666001232
7	7	Forestpark	1300	102	4677001232
8	8	Parkland	1200	106	4688001232
9	9	Pinecreek	1500	109	4699001232
10	10	WalnutHill	1700	105	4691001232

Hospital_Info_2:

Results Messages

	hosp_ID	hosp_name	hosp_needed_Bgrp	hosp_needed_qnty
1	1	MayoClinic	A+	20
2	1	MayoClinic	A-	40
3	1	MayoClinic	AB+	0
4	1	MayoClinic	AB-	20
5	1	MayoClinic	B-	10
6	2	CleavelandClinic	A+	40
7	2	CleavelandClinic	A-	10
8	2	CleavelandClinic	AB+	20
9	2	CleavelandClinic	AB-	10
10	2	CleavelandClinic	B+	0
11	2	CleavelandClinic	B-	30
12	3	NYU	A+	0
13	3	NYU	A-	0
14	3	NYU	AB+	0
15	3	NYU	AB-	0
16	3	NYU	B+	10
17	3	NYU	B-	20
18	4	Baylor	A+	10
19	4	Baylor	A-	40
20	7	Forestpark	B-	40
21	8	Parkland	B+	10
22	9	Pinecreek	AB-	20

Recipient:

Results Messages

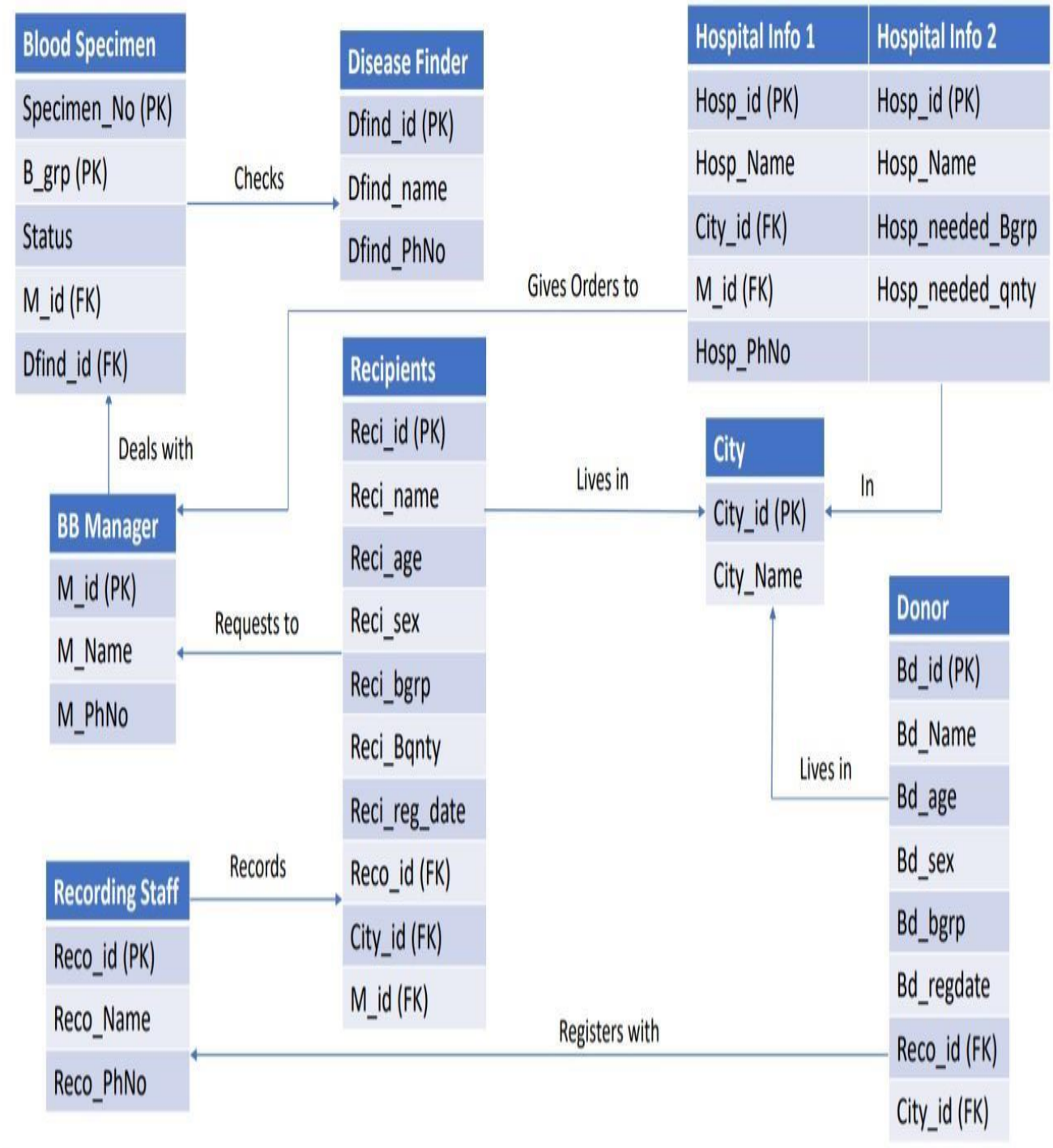
	reci_ID	reci_name	reci_age	reci_Brgp	reci_Bqnty	reco_ID	City_ID	M_id	reci_sex	reci_reg_date
1	10001	Peter	25	B+	1.5	101212	1100	101	M	2015-12-17
2	10002	Dan	60	A+	1	101312	1100	102	M	2015-12-16
3	10003	Steve	35	AB+	0.5	101312	1200	102	M	2015-10-17
4	10004	Parker	66	B+	1	101212	1300	104	M	2016-11-17
5	10005	Jason	53	B-	1	101412	1400	105	M	2015-04-17
6	10006	Preetham	45	O+	1.5	101512	1500	105	M	2015-12-17
7	10007	Swetha	22	AB-	1	101212	1500	101	F	2015-05-17
8	10008	Swathi	25	B+	2	101412	1300	103	F	2015-12-14
9	10009	Lance	30	A+	1.5	101312	1100	104	M	2015-02-16
10	10010	Marsh	25	AB+	3.5	101212	1200	107	M	2016-10-17

Recording_Staff:

Results Messages

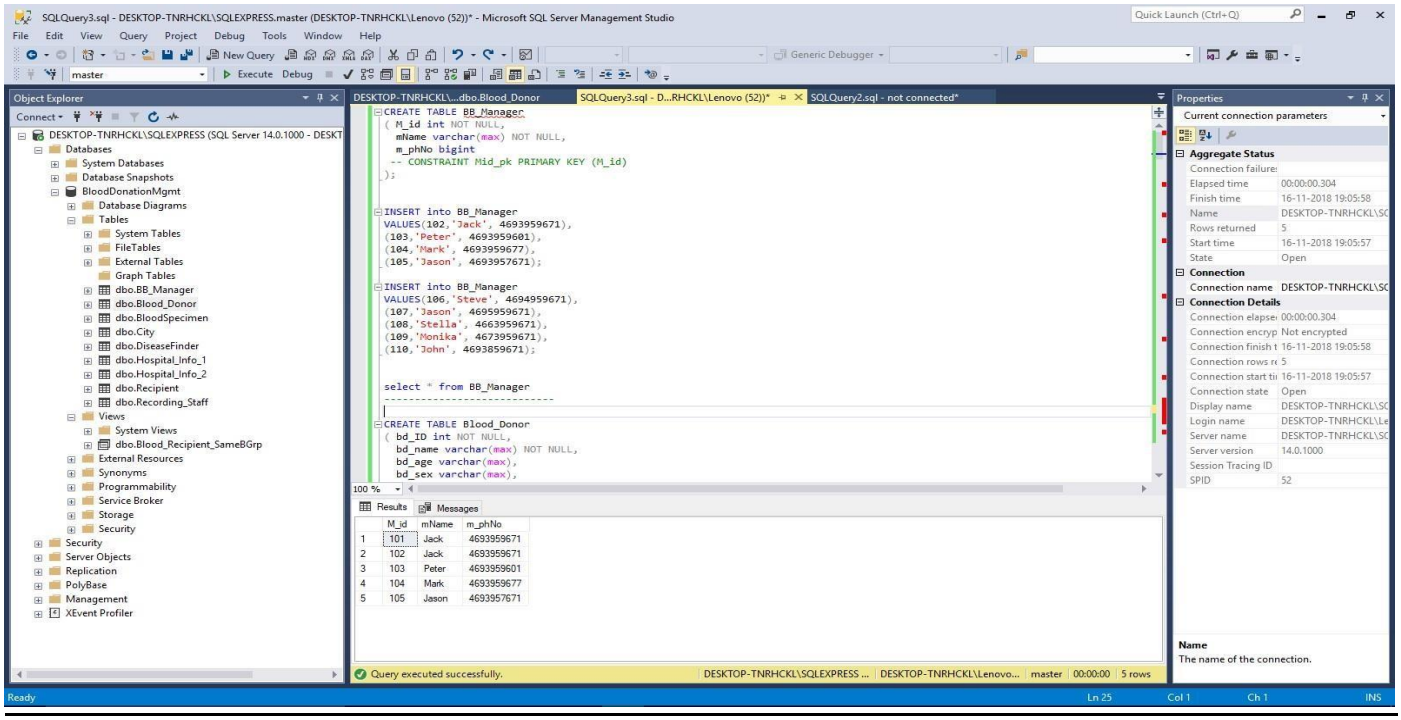
	reco_ID	reco_Name	reco_phNo
1	101012	Lekha	4044846553
2	101112	Mark	4045856553
3	101212	Walcot	4045806553
4	101312	Henry	4045806553
5	101412	Silva	4045806553
6	101512	Adrian	4045806553
7	101612	Mark	4045806553
8	101712	Abdul	4045816553
9	101812	Jerry	4045826553
10	101912	Tim	4045836553

DIAGRAM AFTER NORMALIZATION



SQL IMPLEMENTATION

The implementation on SQL Server is given below [3] [5] [6] [8]:




```
CREATE TABLE BB_Manager ( M_id
int NOT NULL,   mName
varchar(max) NOT NULL,   m_phNo
bigint
 -- CONSTRAINT Mid_pk PRIMARY KEY (M_id)  );


INSERT into BB_Manager
VALUES(102,'Jack', 4693959671),
(103,'Peter', 4693959601), (104,'Mark',
4693959677),
(105,'Jason', 4693957671);

INSERT into BB_Manager
VALUES(106,'Steve', 4694959671),
(107,'Jason', 4695959671),
(108,'Stella', 4663959671),
(109,'Monika', 4673959671),
(110,'John', 4693859671);
  select * from
BB_Manager
----------------------------------------------------------------------------------------
```

```
  Blood_Donor
   bd_ID int NOT NULL, bd_name varchar(max)
NOT NULL, bd_age varchar(max),   bd_sex
varchar(max),   bd_Bgroup varchar(10),
bd_reg_date date,
reco_ID int NOT NULL,
  City_ID int NOT NULL
 -- CONSTRAINT bdID_pk PRIMARY KEY (bd_ID)  );


INSERT into Blood_Donor
VALUES(150221,'Mark',25,'M','B+','2015-12-17',101212,1100),
(160011,'Abdul',35,'F','A+','2016-11-22',101212,1100),
(160101,'Smith',22,'M','O+','2016-01-04',101312,1200),
(150011,'Pat',29,'M','O+','2015-07-19',101412,1300),
(150021,'Shyam',42,'F','A-','2015-12-24',101412,1300), (150121,'Dan',44,'M','AB+','201508-
28',101212,1200),
(160031,'Mike',33,'F','AB-','2016-02-06',101212,1400),
(160301,'Elisa',31,'F','AB+','2016-09-10',101312,1200),
(160091,'Carrol',24,'M','B-','2016-10-15',101312,1500),
(160401,'Mark',29,'M','O-','2016-12-17',101212,1200);
   select * from Blood_Donor
-------------------------------------------------------------------------------------------
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
  CONSTRAINT specimenumber_pk PRIMARY KEY (specimen_number)  );



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);


Select * from BloodSpecimen
-----------------------------------------------------------------------------------------
CREATE TABLE City
( City_ID int NOT NULL,
  City_name varchar(max) NOT NULL,
 -- CONSTRAINT CityID_pk PRIMARY KEY (City_ID)  );

INSERT into City
VALUES(1200,'Austin'),
(1300,'Irving'),
(1400,'Houston'),
(1500,'Richardson');

INSERT into City
VALUES(1600,'Plano'),
(1700,'Frisco'),
(1800,'Arlington'),
(1900,'San Antonio'),
(2000,'Tyler');

select * from City
-----------------------------------------------------------------------------------------




CREATE TABLE DiseaseFinder ( dfind_ID
int NOT NULL,   dfind_name
varchar(max) NOT NULL,   dfind_PhNo
bigint
 -- CONSTRAINT dfindID_pk PRIMARY KEY (dfind_ID)  );



INSERT into DiseaseFinder
VALUES(11,'Peter',4693804223),
(12,'Park',4693804223),
(13,'Jerry',4693804223),
(14,'Mark',4693804223),
(15,'Monika',4693804223);
```

```
INSERT into DiseaseFinder
VALUES(16,'Ram',4693804123),
(17,'Swathi',4693804223),
(18,'Gautham',4693804323),
(19,'Ashwin',4693804423),
(20,'Yash',4693804523);

select * from DiseaseFinder

drop table DiseaseFinder
-----------------------------------------------------------------------------------------
Hospital_Info_1  hosp_ID int NOT
  NULL,  hosp_name varchar(max)
  NOT NULL,
City_ID int NOT NULL,   M_id int
NOT NULL   primary key(hosp_ID)
 -- CONSTRAINT hospID_pk PRIMARY KEY (hosp_ID)  );



INSERT into Hospital_Info_1
VALUES(1,'MayoClinic',1100,101),
(2,'CleavelandClinic',1200,103),
(3,'NYU',1300,103);

INSERT into Hospital_Info_1
VALUES(4,'Baylor',1400,104),
(5,'Charlton',1800,103),
(6,'Greenoaks',1300,106),
(7,'Forestpark',1300,102),
(8,'Parkland',1200,106),
(9,'Pinecreek',1500,109),
(10,'WalnutHill',1700,105);

select * from  Hospital_Info_1
-----------------------------------------------------------------------------------------




CREATE TABLE Hospital_Info_2 ( hosp_ID
int NOT NULL,   hosp_name
varchar(max) NOT NULL,
hosp_needed_Bgrp varchar(10),
hosp_needed_qnty int
  primary key(hosp_ID,hosp_needed_Bgrp)  --
CONSTRAINT hospID_pk PRIMARY KEY (hosp_ID)
);

INSERT into Hospital_Info_2
VALUES(1,'MayoClinic','A+',20),
```

```
(1,'MayoClinic','AB+',0),
(1,'MayoClinic','A-',40),
(1,'MayoClinic','B-',10),
(1,'MayoClinic','AB-',20);

INSERT into Hospital_Info_2
VALUES(2,'CleavelandClinic','A+',40),
(2,'CleavelandClinic','AB+',20),
(2,'CleavelandClinic','A-',10),
(2,'CleavelandClinic','B-',30),
(2,'CleavelandClinic','B+',0),
(2,'CleavelandClinic','AB-',10);

INSERT into Hospital_Info_2
VALUES(3,'NYU','A+',0),
(3,'NYU','AB+',0),  (3,'NYU','A-',0),
(3,'NYU','B-',20),
(3,'NYU','B+',10),
(3,'NYU','AB-',0);

INSERT into Hospital_Info_2
VALUES(4,'Baylor','A+',10),
(5,'Charlton','B+',30),
(4,'Baylor','A-',40),
(7,'Forestpark','B-',40), (8,'Parkland','B+',10),
(9,'Pinecreek','AB-',20);

select * from Hospital_Info_2
-------------------------------------------------------------------------------------------



CREATE TABLE Recipient ( reci_ID
int NOT NULL,   reci_name
varchar(max) NOT NULL,   reci_age
varchar(max),   reci_Brgp
varchar(max),   reci_Bqnty float,
reco_ID int NOT NULL,   City_ID int
NOT NULL,   M_id int NOT NULL,
reci_sex varchar(max),
reci_reg_date date
 -- CONSTRAINT reciid_pk PRIMARY KEY (reci_id)  );

Alter table Recipient
ADD reci_sex varchar(max);

Alter table Recipient
```

```
ADD reci_reg_date date;

INSERT into Recipient
VALUES(10001,'Mark',25,'B+',1.5,101212,1100,101,'M','2015-12-17'),
(10002,'Dan',60,'A+',1,101312,1100,102,'M','2015-12-16'),
(10003,'Steve',35,'AB+',0.5,101312,1200,102,'M','2015-10-17'),
(10004,'Parker',66,'B+',1,101212,1300,104,'M','2016-11-17'),
(10005,'Jason',53,'B-',1,101412,1400,105,'M','2015-04-17'),
(10006,'Preetham',45,'O+',1.5,101512,1500,105,'M','2015-12-17'),
(10007,'Swetha',22,'AB-',1,101212,1500,101,'F','2015-05-17');
INSERT into Recipient
VALUES(10008,'Swathi',25,'B+',2,101412,1300,103,'F','2015-12-14'),
(10009,'Lance',30,'A+',1.5,101312,1100,104,'M','2015-02-16'),
(10010,'Marsh',25,'AB+',3.5,101212,1200,107,'M','2016-10-17');
select * from
Recipient

Drop table Recipient
-----------------------------------------------------------------------------------------
Recording_Staff  reco_ID int NOT
  NULL, reco_Name
  varchar(max) NOT NULL,
  reco_phNo bigint
 -- CONSTRAINT recoID_pk PRIMARY KEY (reco_ID)  );

INSERT into Recording_Staff
VALUES(101212,'Walcot',4045806553),
(101312,'Henry',4045806553),
(101412,'Silva',4045806553),
(101512,'Adrian',4045806553),
(101612,'Mark',4045806553);

INSERT into Recording_Staff
VALUES(101712,'Abdul',4045816553),
(101812,'Jerry',4045826553),  (101912,'Tim',4045836553),
(101012,'Lekha',4044846553),
(101112,'Mark',4045856553);

select * from Recording_Staff

-----------------------------------------------------------------------------------------
  update City set City_name = 'Allen' where City_ID =
2000   delete from Hospital_Info_2 where hosp_name =
'Charlton'
```

SAMPLE SQL QUERIES

1. **Create a View of recipients and donors names having the same blood group registered on the same date.**

```
CREATE VIEW Blood_Recipient_SameBGrp; AS  select
Blood_Donor.bd_name,Recipient.reci_name,reco_Name from Recording_Staff
inner join Blood_Donor on Recording_Staff.reco_ID = Blood_Donor.reco_ID
inner join Recipient on Recording_Staff.reco_ID = Recipient.reco_ID
where Blood_Donor.bd_Bgroup = Recipient.reci_Brgp and
Blood_Donor.bd_reg_date = Recipient.reci_reg_date
```

Output:

```
select* from Blood_Recipient_SameBGrp;
```

Results | Messages

	bd_name	reci_name	reco_Name
1	Mark	Peter	Walcot

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

```
Select specimen_number,b_group from BloodSpecimen,DiseaseFinder WHERE
BloodSpecimen.dfind_ID= DiseaseFinder.dfind_ID AND dfind_name='Mark'
AND status=1
```

Output:

Results | Messages

	specimen_number	b_group
1	1012	O-
2	1013	B-

3. **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.**

```
select BB_Manager.M_id,mName,Recipient.reci_name, Recipient.reci_Brgp,b_group
from BB_Manager,Recipient,BloodSpecimen  where Recipient.M_id =
BloodSpecimen.M_id and Recipient.reci_Brgp =
BloodSpecimen.b_group and
status = 1
```

Output:

Results | Messages

	M_id	mName	reci_name	reci_Brgp	b_group
1	101	Jack	Peter	B+	B+
2	102	Mark	Steve	AB+	AB+

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

```
Select bd_ID,bd_name,reci_ID,reci_name FROM Blood_Donor,Recipient
WHERE bd_Bgroup=reci_Brgp AND Blood_Donor.City_ID= Recipient.City_ID
```

Output:

Results | Messages

	bd_ID	bd_name	reci_ID	reci_name
1	150221	Mark	10001	Peter
2	160011	Abdul	10002	Dan
3	150121	Dan	10003	Steve
4	160301	Elisa	10003	Steve
5	160011	Abdul	10009	Lance
6	150121	Dan	10010	Marsh
7	160301	Elisa	10010	Marsh

5. **Display the information of Hospital_Info_1 handled by BB_Manager whose ID is 103:**

```
Select hosp_ID,hosp_name , City_ID, HOspital_Info_1.M_id from Hospital_Info_1,BB_Manager
where BB_Manager.M_id=Hospital_Info_1.M_id and BB_Manager.M_id=103
```

Results | Messages

	hosp_ID	hosp_name	City_ID	M_id
1	2	CleavelandClinic	1200	103
2	3	NYU	1300	103
3	5	Charlton	1800	103

CONCLUSION

Our project well addressed the limitations of the existing system. We designed well organized database management system which is a challenging job in this era. We have built a database for a Blood Bank using Microsoft SQL Server. 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, we have created our ERD and then converted the ERD to relational model and normalized the tables. Using Microsoft SQL Server we have created the tables for our database and inserted some sample values in the tables. Finally, we have executed sample queries on our database to check its performance to retrieve useful information accurately and speedily.

REFERENCES

[1] Organ Donation Management System - Chitra , Ramasamy (2007) - Melaka, Malaysia, UTeM.

[2] Charity Management System - Sara Aljohani (2015)

[3] Blood Donation Management: Recent trends – Sunil K Joseph, Johnson Michael, Prem Jose Vazhacharickal

[4] SQL server tutorial: https://www.tutorialspoint.com/ms_sql_server/

Notes from Prof. Hakki C Cankaya, University of Texas at Dallas, Dallas, Texas