**BLOOD BANK MANAGEMENT SYSTEM**

(DBMS PROJECT)

Surender Nadar - 43

**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 types of blood in emergency situations from either blood banks 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 intention of proposing such a system is to abolish the panic caused during an emergency due to unavailability of blood.

**INTRODUCTION**

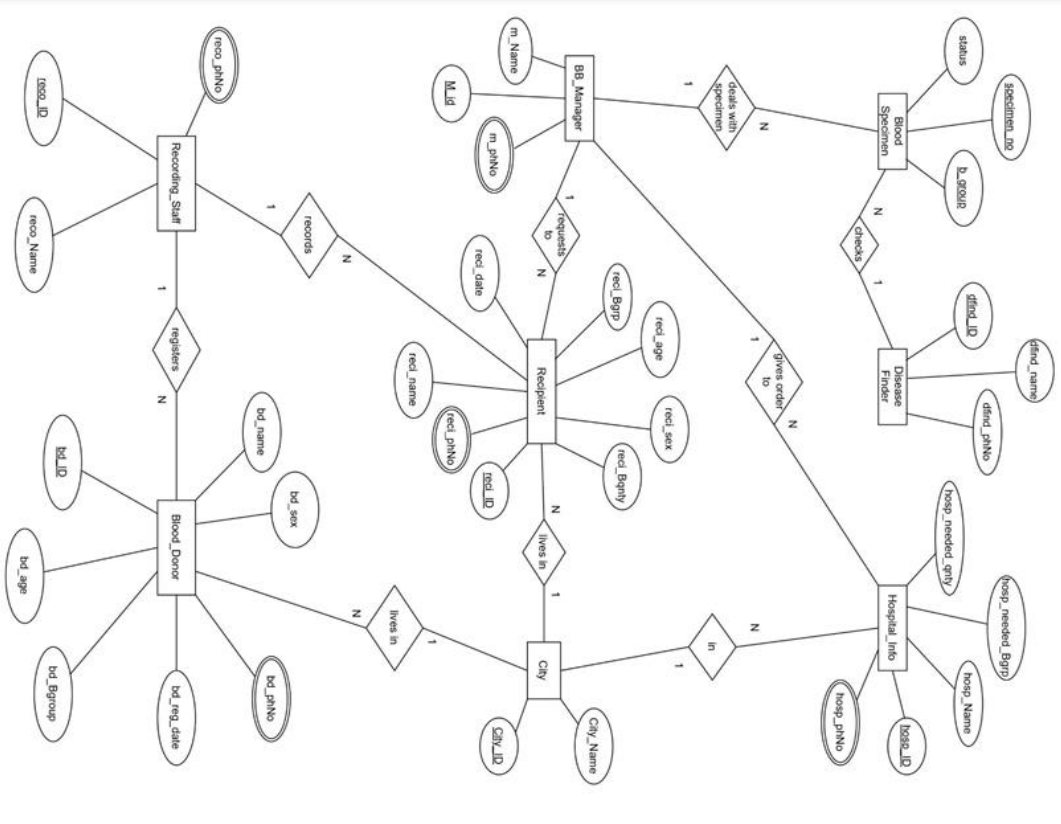
Blood banks collect, store and provide collected blood to the patients who are in need of blood. The people who donate blood are called ‘donors’. The banks then group the blood which they receive according to the blood groups. They also make sure that the blood is not contaminated. The main mission of 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 donate the blood, that is ‘donors’. But this is a tough job. The existing system will not satisfy the need of maintaining quality blood and keeping 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 a 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 manpower 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 the primary key to identify the donor information. Other than that name, age , sex , blood group, phone number and registration dates will be stored in the 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 receives blood from a blood bank, when blood is given to a recipient a recipient ID (reci\_ID) is generated and used as the primary key for the recipient entity to identify blood recipients information. Along with its name ,age, sex, blood group (needed), blood quantity(needed) , phone number, and registration dates are also stored in the database under the recipient entity.

3. **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 the database under BB\_Manager entity.

4. **Recording\_Staff** :

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

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

5. **BloodSpecimen** :

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

In the database, under the 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.

6. **DiseaseFinder** :

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

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

7. **Hospital\_Info** :

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

In the database, under the 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 the hospital name and the blood quantity required at the hospital.

8. **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 the primary key to identify the information about the city. Along with ID city names will also be stored under this entity.

**RELATIONAL SCHEMAS**

**Donor Table:**

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

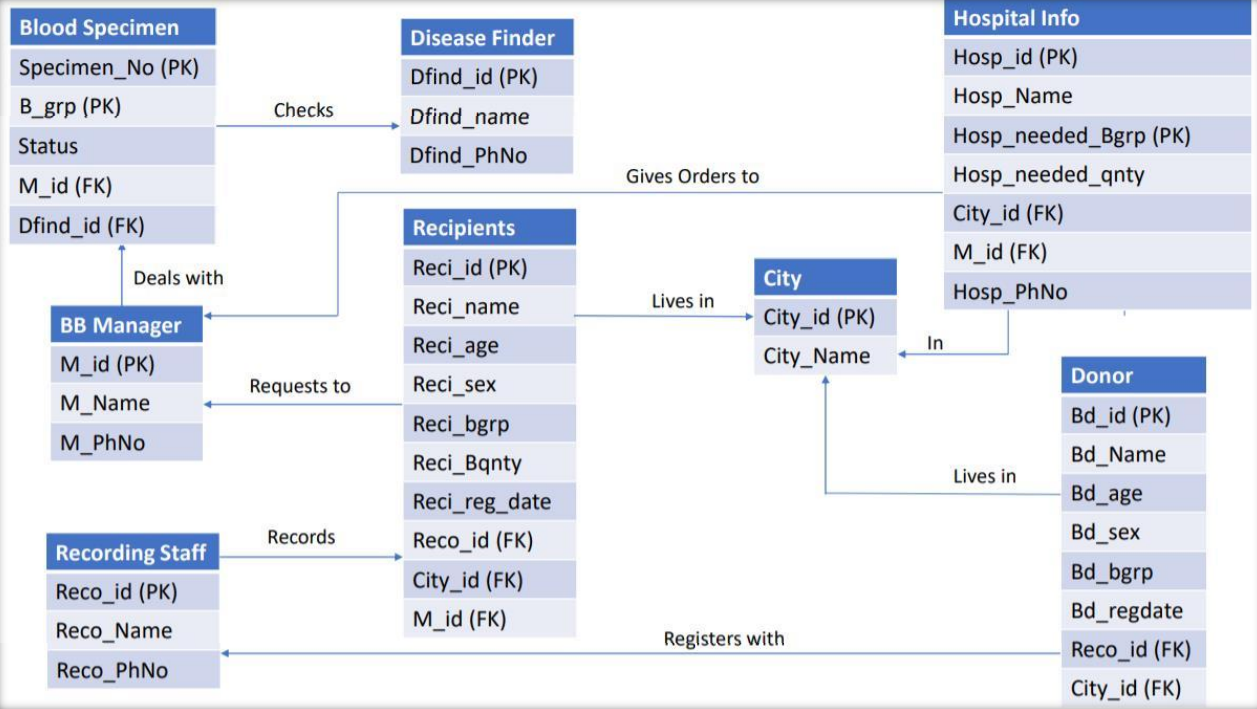
**Blood Bank Manager Table:**

* 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 the 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 the primary key of the Blood Bank manager is used as a foreign key in Hospital info.

**RELATIONAL SCHEMA**



**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:**

* **Blood\_Donor**

(bd\_Id, bd\_name, bd\_phNo bd\_sex, bd\_age, bd\_reg\_date, bd\_Bgroup, reco\_ID, City\_ID)

(Functional Dependency Exists)

1. The table is in its first normal form.
2. The table is in its second normal form.
3. The table is in its third normal form.

* **City**

(city\_id , city\_name)

(Functional Dependency Exists)

1. The table is in its first normal form.
2. The table is in its second normal form.
3. The table is in its third normal form.

* **Recording\_staff**

(reco\_name, reco\_ID, reco\_phNo)

(Functional Dependency Exists)

1. The table is in its first normal form.
2. The table is in its second normal form.
3. The table is in its third normal form.

* **Blood\_recipient**

(reci\_Id, reci\_sex, reci\_phNo, reci\_age, reci\_date, reci\_name, reci\_Bqnty, reci\_Bgrp, reco\_id, city\_id, m\_id)

(Functional Dependency Exists)

1. The table is in its first normal form.
2. The table is in its second normal form.
3. The table is in its third normal form.

* **Blood Specimen**

( b\_group, specimen\_no, status, dfind\_id, m\_id )

(Functional Dependency Exists)

1. The table is in its first normal form.
2. The table is in its second normal form.
3. The table is in its third normal form.

* **Disease\_finder**

( dfind\_id, dfind\_name, dfind\_PhNo)

(Functional Dependency Exists)

1. The table is in its first normal form.
2. The table is in its second normal form.
3. The table is in its third normal form.

* **BB\_manager**

( M\_id, m\_name, m\_phNo)

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

(Functional Dependency Exists)

1. The table is in its first normal form.
2. The table is in its second normal form.
3. The table is in its third normal form.

* **Hospital\_Info**

( hosp\_Id, hosp\_Name, hosp\_phNo, hosp\_needed\_Bgrp, hosp\_needed\_qty, city\_id, m\_id)

1. The table is in its first normal form.
2. 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.

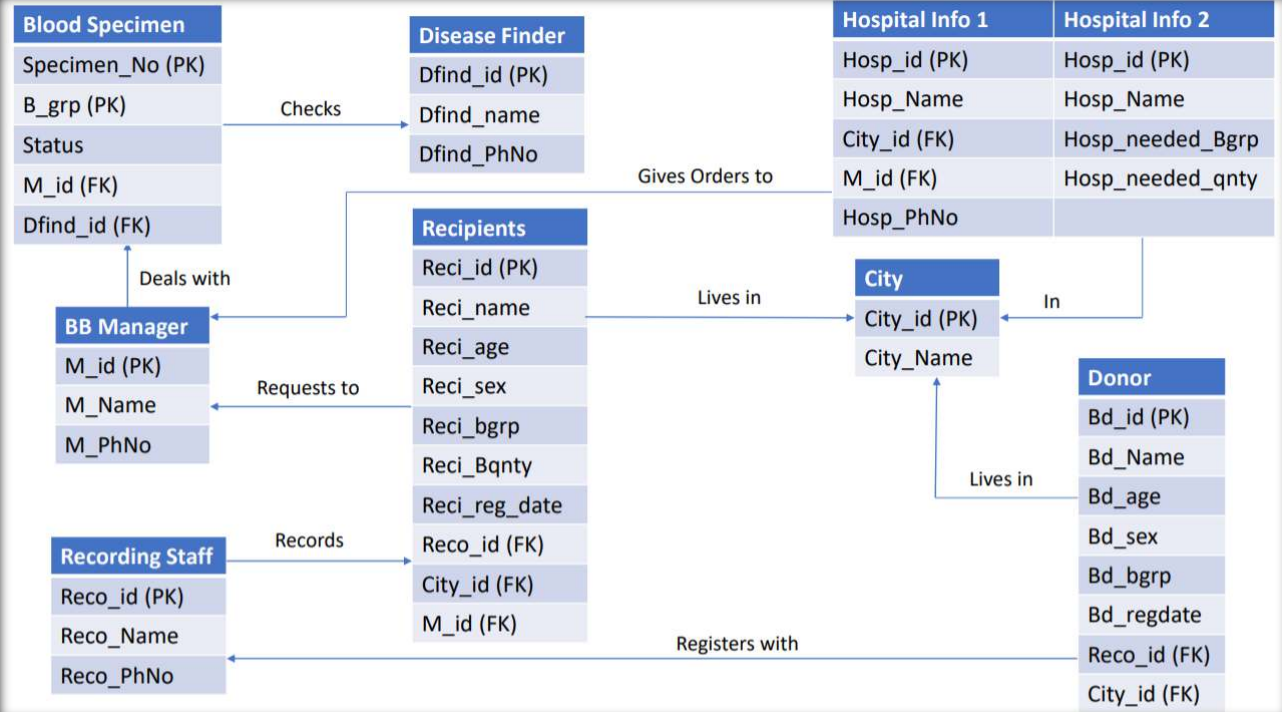
Hospital\_info\_1 (hosp\_Id, hosp\_phNo, hosp\_Name, city\_id, m\_id).

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

Now it is in its second normal form.

1. The table is in its third normal form.

**RELATIONAL SCHEMA AFTER NORMALIZATION**



**SQL IMPLEMENTATION**

**Create BB\_Manager Table:**

CREATE TABLE BB\_Manager

( M\_id int NOT NULL PRIMARY KEY,

mName varchar(20) NOT NULL,

m\_phNo int

);

INSERT ALL

INTO BB\_Manager VALUES(101,'shivank', 9693959671)

INTO BB\_Manager VALUES(102,'shwetanshu', 9693959672)

INTO BB\_Manager VALUES(103,'singh', 9693959673)

INTO BB\_Manager VALUES(104,'yusuf', 9693959674)

INTO BB\_Manager VALUES(105,'jackson', 9693959675)

INTO BB\_Manager VALUES(106,'akhil', 9693959676)

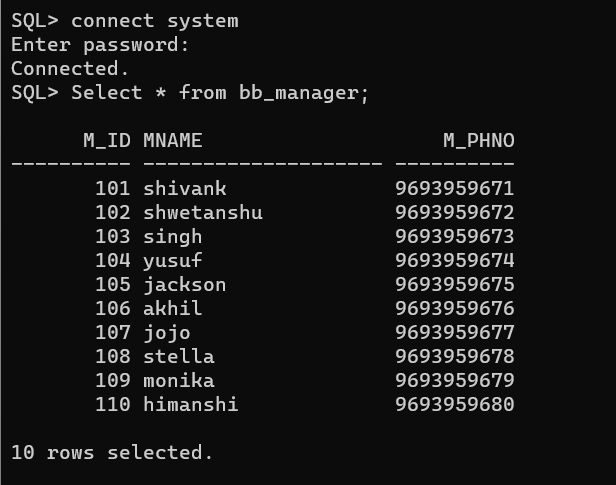
INTO BB\_Manager VALUES(107,'jojo', 9693959677)

INTO BB\_Manager VALUES(108,'stella', 9693959678)

INTO BB\_Manager VALUES(109,'monika', 9693959679)

INTO BB\_Manager VALUES(110,'himanshi', 9693959680)

select \* from dual;



**Create City Table:**

CREATE TABLE City

( City\_ID int NOT NULL PRIMARY KEY,

City\_name varchar(20)NOT NULL

);

INSERT ALL

into City VALUES(1100,'Dallas')

into City VALUES(1200,'Austin')

into City VALUES(1300,'Irving')

into City VALUES(1400,'Houston')

into City VALUES(1500,'Richardson')

into City VALUES(1600,'Plano')

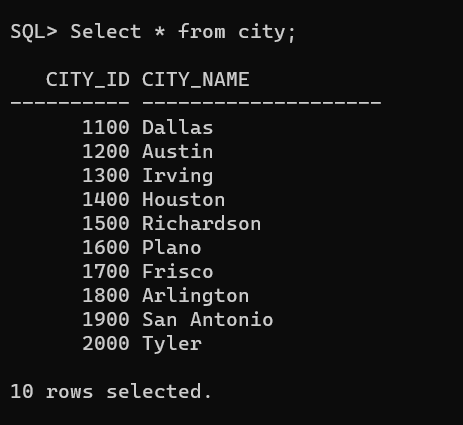
into City VALUES(1700,'Frisco')

into City VALUES(1800,'Arlington')

into City VALUES(1900,'San Antonio')

into City VALUES(2000,'Tyler')

select \* from dual;



**Create Recipient Table:**

CREATE TABLE Recipient

( reci\_ID int NOT NULL PRIMARY kEY,

reci\_name varchar(20)NOT NULL,

reci\_age varchar(5),

reci\_Brgp varchar(5),

reci\_Bqnty float,

reco\_ID int NOT NULL,

City\_ID int NOT NULL,

M\_id int NOT NULL,

reci\_sex varchar(5),

reci\_reg\_date date,

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

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

);

INSERT ALL

into Recipient VALUES(10001,'Peter',25,'B+',1.5,101212,1100,101,'M','17-DEC-2015')

into Recipient VALUES(10002,'shivank',60,'A+',1,101312,1100,102,'M','16-DEC-2015')

into Recipient VALUES(10003,'akhil',35,'AB+',0.5,101312,1200,102,'M','17-OCT-2015')

into Recipient VALUES(10004,'Parker',66,'B+',1,101212,1300,104,'M','17-NOV-2016')

into Recipient VALUES(10005,'jojo',53,'B-',1,101412,1400,105,'M','17-APR-2015')

into Recipient VALUES(10006,'Preetham',45,'O+',1.5,101512,1500,105,'M','17-DEC-2015')

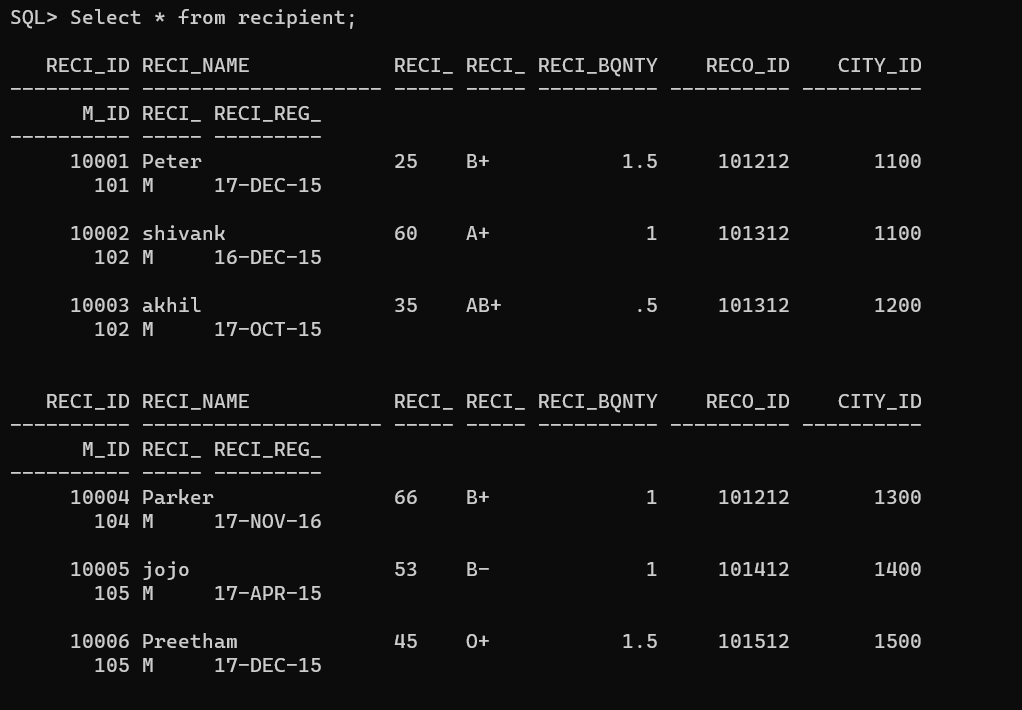
into Recipient VALUES(10007,'Swetha',22,'AB-',1,101212,1500,101,'F','17-MAY-2015')

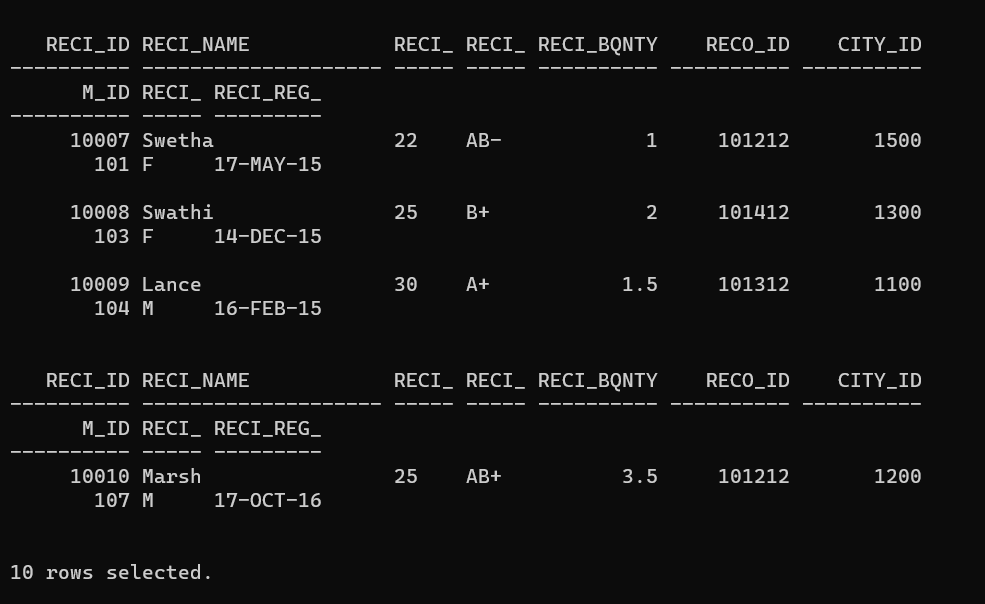
into Recipient VALUES(10008,'Swathi',25,'B+',2,101412,1300,103,'F','14-DEC-2015')

into Recipient VALUES(10009,'Lance',30,'A+',1.5,101312,1100,104,'M','16-FEB-2015')

into Recipient VALUES(10010,'Marsh',25,'AB+',3.5,101212,1200,107,'M','17-OCT-2016')

select \* from dual;





**Create Recording\_Staff Table:**

CREATE TABLE Recording\_Staff

( reco\_ID int NOT NULL PRIMARY KEY,

reco\_Name varchar(20) NOT NULL,

reco\_phNo int

);

INSERT ALL

into Recording\_Staff VALUES(101012,'Lekha',4044846553)

into Recording\_Staff VALUES(101112,'shivam',4045856553)

into Recording\_Staff VALUES(101212,'Walcot',4045806553)

into Recording\_Staff VALUES(101312,'jackson',4045806553)

into Recording\_Staff VALUES(101412,'Silva',4045806553)

into Recording\_Staff VALUES(101512,'Adrian',4045806553)

into Recording\_Staff VALUES(101612,'shivam',4045806553)

into Recording\_Staff VALUES(101712,'shyam',4045816553)

into Recording\_Staff VALUES(101812,'Jerry',4045826553)

into Recording\_Staff VALUES(101912,'Tim',4045836553)

select \* from dual;



**Create Blood\_Donor Table:**

CREATE TABLE Blood\_Donor

( bd\_ID int NOT NULL PRIMARY KEY,

bd\_name varchar(20) NOT NULL,

bd\_age varchar(5),

bd\_sex varchar(5),

bd\_Bgroup varchar(5),

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)

);

INSERT ALL

INTO Blood\_Donor VALUES(150011,'Mark',25,'M','O+','19-JUL-2015',101412,1100)

INTO Blood\_Donor VALUES(150012,'Abdul',35,'M','A-','24-DEC-2015',101412,1100)

INTO Blood\_Donor VALUES(150013,'Shivank',22,'M','AB+','28-AUG-2015',101212,1200)

INTO Blood\_Donor VALUES(150014,'shweta',29,'M','B+','17-DEC-2015',101212,1300)

INTO Blood\_Donor VALUES(150015,'Shyam',42,'M','A+','22-NOV-2016',101212,1300)

INTO Blood\_Donor VALUES(150016,'Dan',44,'F','AB-','06-FEB-2016',101212,1200)

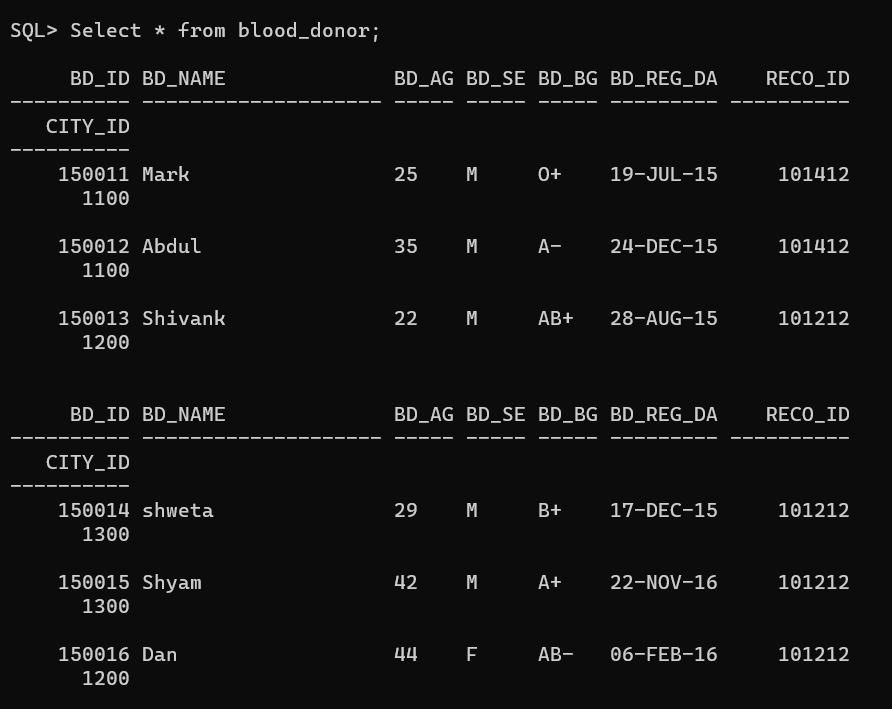
INTO Blood\_Donor VALUES(150017,'Mike',33,'M','B-','15-OCT-2016',101312,1400)

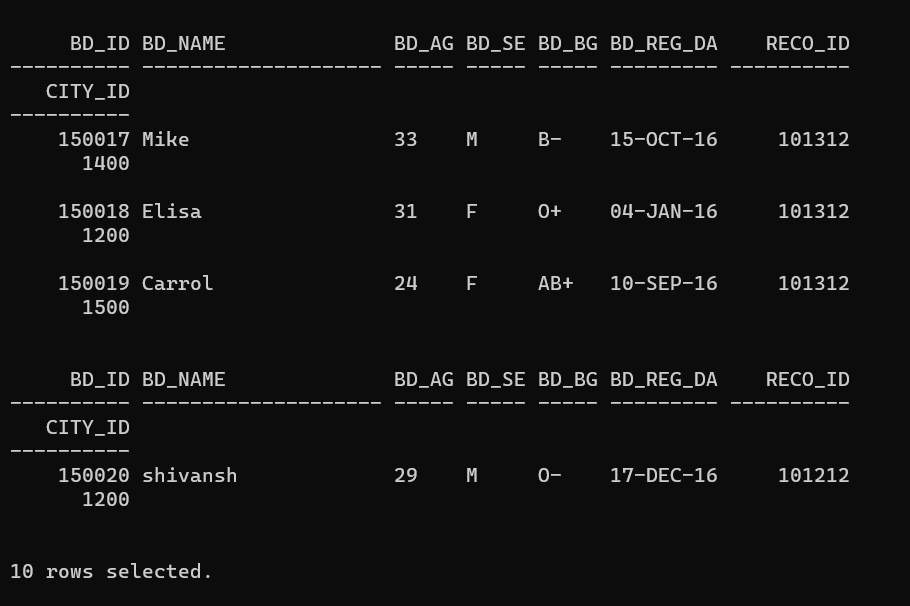
INTO Blood\_Donor VALUES(150018,'Elisa',31,'F','O+','04-JAN-2016',101312,1200)

INTO Blood\_Donor VALUES(150019,'Carrol',24,'F','AB+','10-SEP-2016',101312,1500)

INTO Blood\_Donor VALUES(150020,'shivansh',29,'M','O-','17-DEC-2016',101212,1200)

select \* from dual;





**Create DiseaseFinder Table:**

CREATE TABLE DiseaseFinder

( dfind\_ID int NOT NULL PRIMARY KEY,

dfind\_name varchar(20) NOT NULL,

dfind\_PhNo int

);

INSERT ALL

into DiseaseFinder VALUES(11,'Peter',9693959681)

into DiseaseFinder VALUES(12,'Park',9693959682)

into DiseaseFinder VALUES(13,'Jerry',9693959683)

into DiseaseFinder VALUES(14,'shivam',9693959672)

into DiseaseFinder VALUES(15,'Monika',9693959679)

into DiseaseFinder VALUES(16,'Ram',9693959684)

into DiseaseFinder VALUES(17,'Swathi',9693959685)

into DiseaseFinder VALUES(18,'Gautham',9693959686)

into DiseaseFinder VALUES(19,'Ashwin',9693959687)

into DiseaseFinder VALUES(20,'Yash',9693959688)

select \* from dual;



**Create 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)

);

INSERT ALL

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

into BloodSpecimen VALUES(1002, 'O+', 1,12,102)

into BloodSpecimen VALUES(1003, 'AB+', 1,11,102)

into BloodSpecimen VALUES(1004, 'O-', 1,13,103)

into BloodSpecimen VALUES(1005, 'A+', 0,14,101)

into BloodSpecimen VALUES(1006, 'A-', 1,13,104)

into BloodSpecimen VALUES(1007, 'AB-', 1,15,104)

into BloodSpecimen VALUES(1008, 'AB-', 0,11,105)

into BloodSpecimen VALUES(1009, 'B+', 1,13,105)

into BloodSpecimen VALUES(1010, 'O+', 0,12,105)

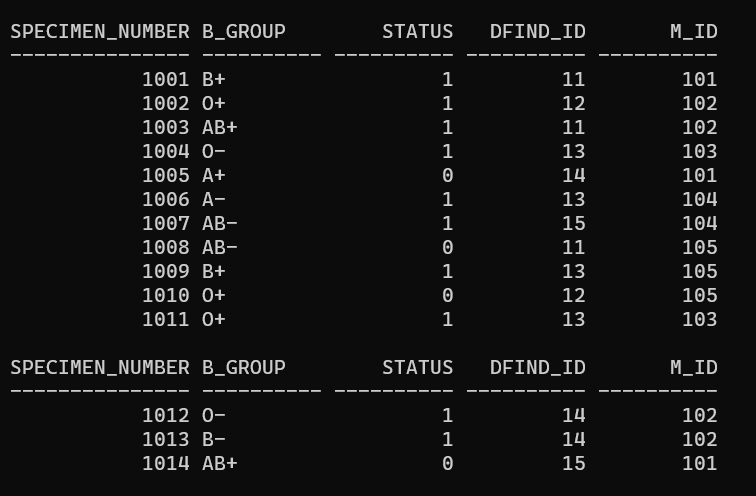
into BloodSpecimen VALUES(1011, 'O+', 1,13,103)

into BloodSpecimen VALUES(1012, 'O-', 1,14,102)

into BloodSpecimen VALUES(1013, 'B-', 1,14,102)

into BloodSpecimen VALUES(1014, 'AB+', 0,15,101)

select \* from dual;



**Create Hospital\_info\_1 Table:**

CREATE TABLE Hospital\_Info\_1

( hosp\_ID int NOT NULL,

hosp\_name varchar(20) 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)

);

INSERT ALL

into Hospital\_Info\_1 VALUES(1,'MayoClinic',1100,101)

into Hospital\_Info\_1 VALUES(2,'CleavelandClinic',1200,103)

into Hospital\_Info\_1 VALUES(3,'NYU',1300,103)

into Hospital\_Info\_1 VALUES(4,'Baylor',1400,104)

into Hospital\_Info\_1 VALUES(5,'Charlton',1800,103)

into Hospital\_Info\_1 VALUES(6,'Greenoaks',1300,106)

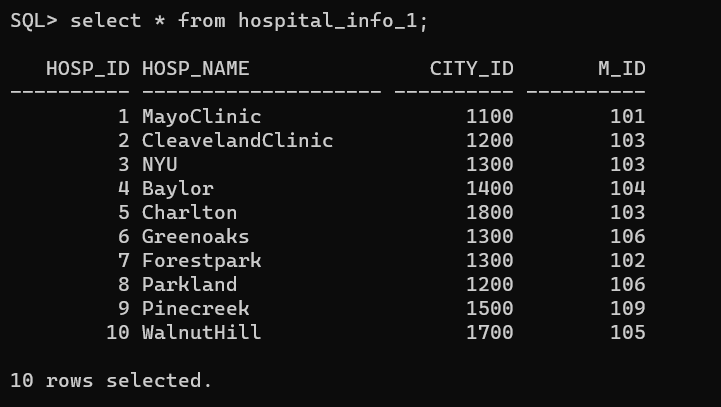
into Hospital\_Info\_1 VALUES(7,'Forestpark',1300,102)

into Hospital\_Info\_1 VALUES(8,'Parkland',1200,106)

into Hospital\_Info\_1 VALUES(9,'Pinecreek',1500,109)

into Hospital\_Info\_1 VALUES(10,'WalnutHill',1700,105)

select \* from dual;



**Create Hospital\_info\_2 Table:**

CREATE TABLE Hospital\_Info\_2

( hosp\_ID int NOT NULL,

hosp\_name varchar(20) NOT NULL,

hosp\_needed\_Bgrp varchar(10),

hosp\_needed\_qnty int,

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

);

INSERT ALL

into Hospital\_Info\_2 VALUES(1,'MayoClinic','A+',20)

into Hospital\_Info\_2 VALUES(1,'MayoClinic','A-',0)

into Hospital\_Info\_2 VALUES(1,'MayoClinic','AB+',40)

into Hospital\_Info\_2 VALUES(1,'MayoClinic','AB-',10)

into Hospital\_Info\_2 VALUES(1,'MayoClinic','B-',20)

into Hospital\_Info\_2 VALUES(2,'CleavelandClinic','A+',40)

into Hospital\_Info\_2 VALUES(2,'CleavelandClinic','AB+',20)

into Hospital\_Info\_2 VALUES(2,'CleavelandClinic','A-',10)

into Hospital\_Info\_2 VALUES(2,'CleavelandClinic','B-',30)

into Hospital\_Info\_2 VALUES(2,'CleavelandClinic','B+',0)

into Hospital\_Info\_2 VALUES(2,'CleavelandClinic','AB-',10)

into Hospital\_Info\_2 VALUES(3,'NYU','A+',0)

into Hospital\_Info\_2 VALUES(3,'NYU','AB+',0)

into Hospital\_Info\_2 VALUES(3,'NYU','A-',0)

into Hospital\_Info\_2 VALUES(3,'NYU','B-',20)

into Hospital\_Info\_2 VALUES(3,'NYU','B+',10)

into Hospital\_Info\_2 VALUES(3,'NYU','AB-',0)

into Hospital\_Info\_2 VALUES(4,'Baylor','A+',10)

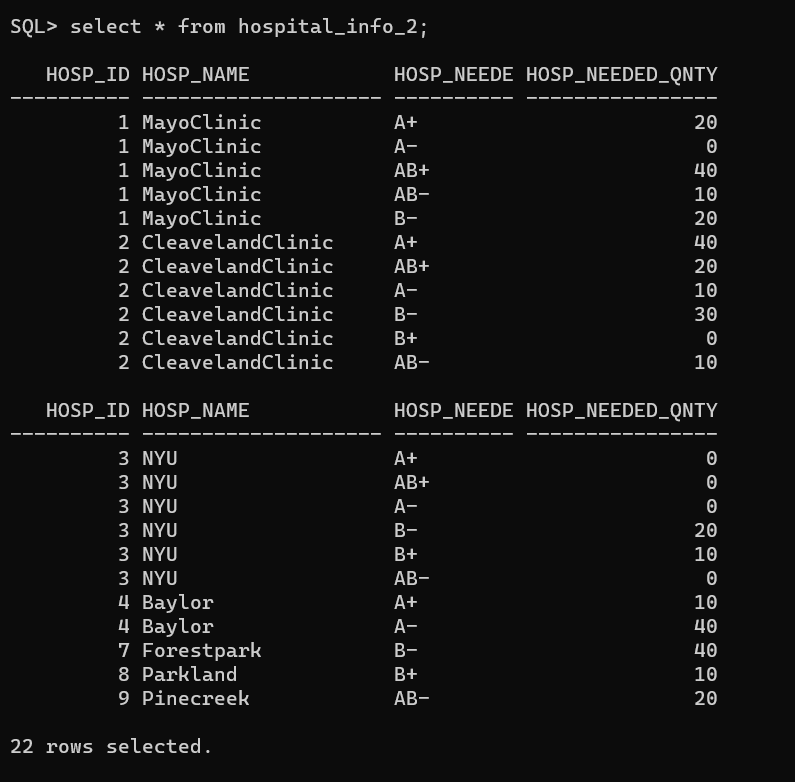
into Hospital\_Info\_2 VALUES(4,'Baylor','A-',40)

into Hospital\_Info\_2 VALUES(7,'Forestpark','B-',40)

into Hospital\_Info\_2 VALUES(8,'Parkland','B+',10)

into Hospital\_Info\_2 VALUES(9,'Pinecreek','AB-',20)

select \* from dual;



**SAMPLE SQL QUERY**

1.Write an Sql Query to print Blood donors from blood\_donor order by

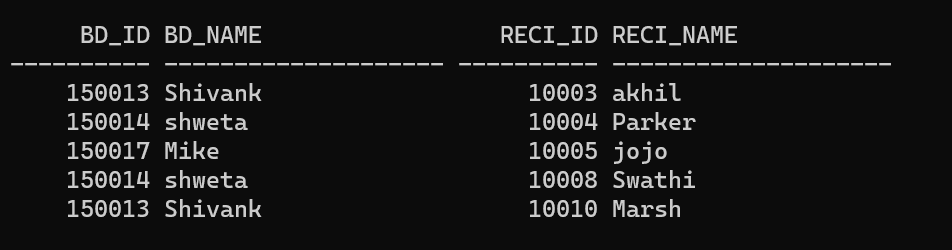
bd\_name Ascending

**Query :** *Select \* from Blood\_donor order by bd\_name asc;*



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

**Query**: *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;*



3. Write an SQL query to print details of Blood\_donor with Bd\_bgroup as “AB+”.

**Query**: *Select \* from blood\_donor where bd\_bgroup like 'AB+%';*

