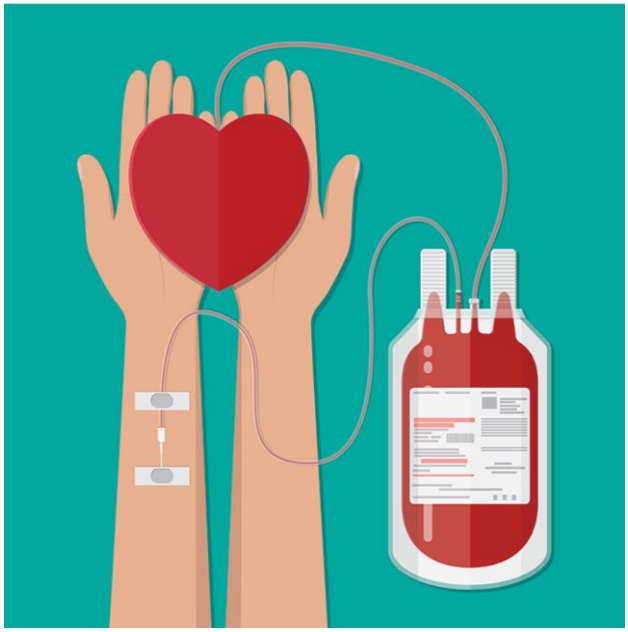
THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)



BLOOD BANK MANAGEMENT SYSTEM

Course: DataBase Management System (UCS310)

Submitted By:

1. Anshita (102003182)

INDEX

PROBLEM STATEMENT	1
ER DIAGRAM	2
INFORMATION OF ENTITIES	3-4
RELATIONSHIP BETWEEN ENTITIES	5-6
ER DIAGRAM TO TABLES	7
RELATIONAL SCHEMAS	8-10
NORMALIZATION	11-13
TABLES AFTER NORMALIZATION	14-18
SQL QUERIES	19-27
DATABASE	28-32

PROBLEM STATEMENT

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.

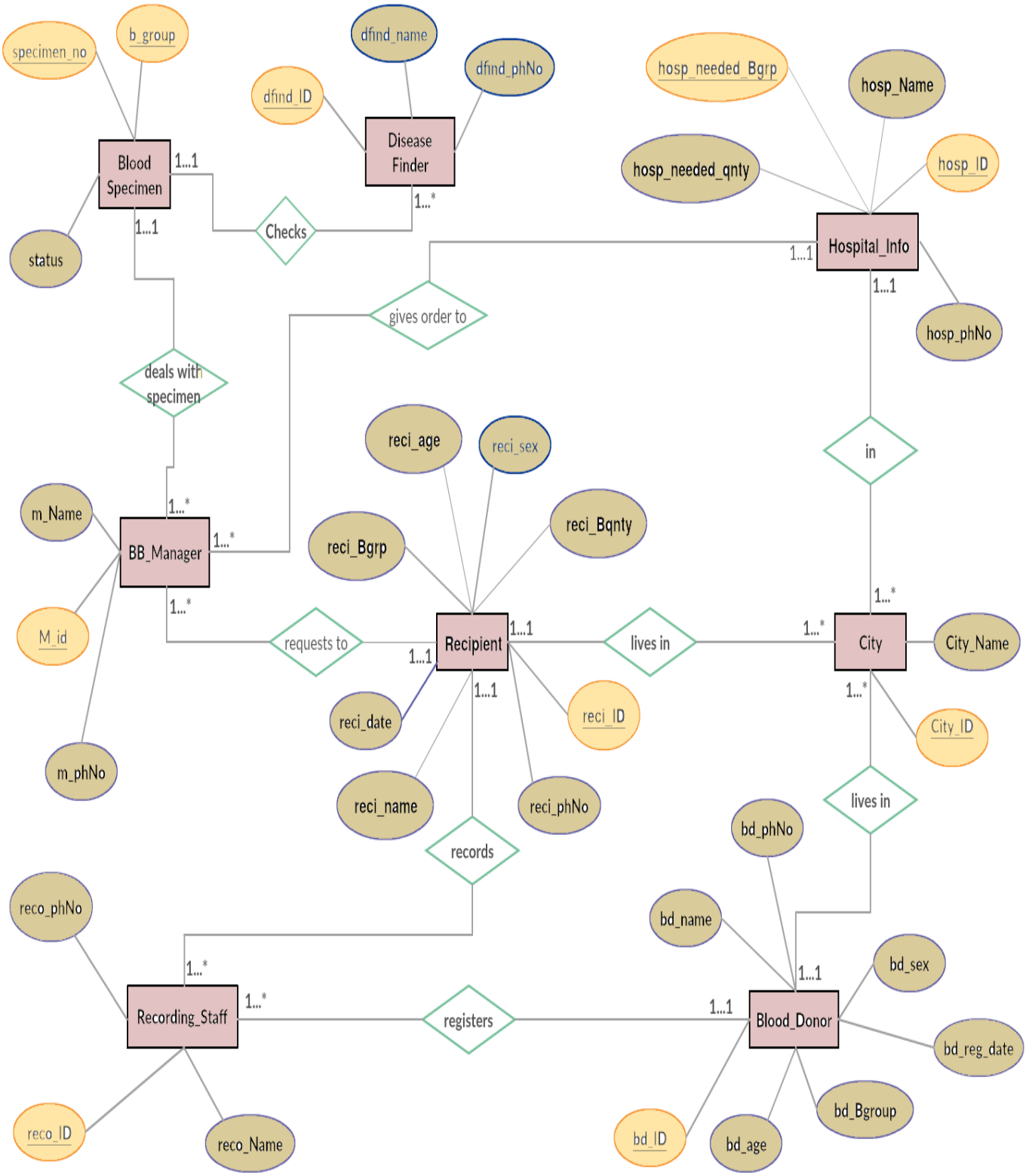
The major concern each blood bank has is to monitor the quality of the blood and monitor the people who donates the blood, that is ‘donors’. But this a tough job. The existing system will not satisfy the need of maintaining quality blood and keep track of donors. To overcome all these limitations we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

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 Oracle 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 Oracle 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.

ER DIAGRAM


specimen_no
b_group
dfind_name
dfind_phNo
dfind_ID
Disease Finder
Blood Specimen
1...1
1...*
Checks
status
hosp_needed_Bgrp
hosp_Name
hosp_needed_qnty
hosp_ID
Hospital_Info
hosp_phNo
gives order to
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
City_ID
reci_date
reci_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
bd_Bgroup
reco_ID
reco_Name
bd_ID
bd_age

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. **BB_Manager: (Attributes – m_ID, m_Name, m_phNo)**

   The blood bank manager is the person who takes care of the avaible blood samples in the blood bank, he is also resposible for handaling blood requests from recipients and hospitals. Blood manager has a unique indentfication 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.

4. **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.

5. **BloodSpecimen : (Attributes – specimen_number, b_group , status)**

   In data base, under BloodSpecimen 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)**

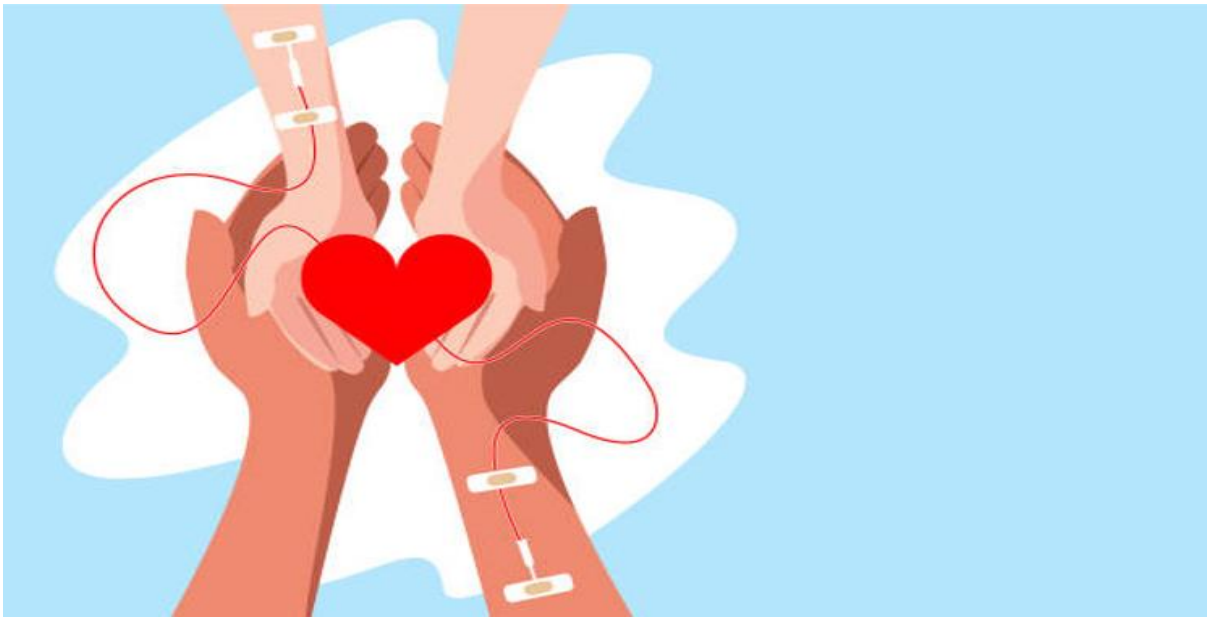
   In data base , under DiseaseFinder entity we will store the information of the doctor who checks the blood for any kind of contaminations. To store that information we have unique identification number (dfind_ID) as primary key. Along with name and phone number of the doctor will also be stored under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

7. **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.

8. **city: (Attributes- city_ID, city_name)**

   This entity will store the information of cities where donors, recipients and hospitals are present. A unique indentification number (City_ID) will be used as primary key to indefiy 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 hospital in it. One hospital will belong in one city.

2. **City and Blood_Donor:**
   Relationship = “lives in”
   Type of relation = 1 to many
   Explanation = In a city, many donor can live. One donor will belong to one city.

3. **City and Recipient:**
   Relationship = “lives in”
   Type of relation = 1 to many
   Explanation = In a city, many recipient can live. One recipient will belong to one city.

4. **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.

5. **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.

6. **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.

7. **BB_Manager and Blood Specimen:**

   Relationship = “deales with specimen”

   Type of relation = 1 to many

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

8. **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.

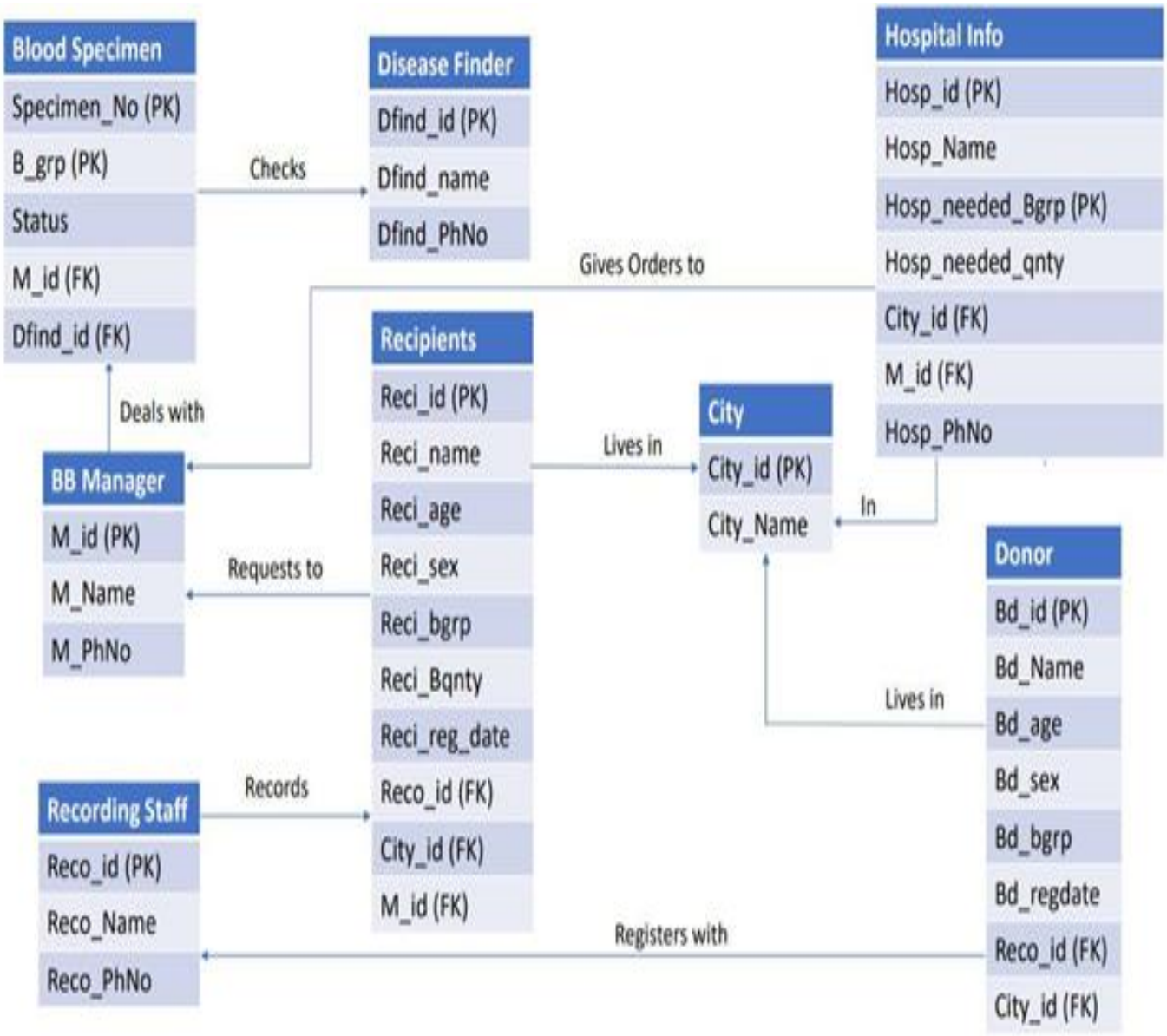
9. **Disease_finder and Blood Specimen:**

   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

ER DIAGRAM TO TABLES



RELATIONAL SCHEMAS

TABLE BB_MANAGER

Column	Null?	Type
M_ID	NOT NULL	NUMBER(10,0)
MNAME	NOT NULL	VARCHAR2(100)
M_PHNO	-	NUMBER(10,0)

TABLE BLOOD_DONOR

Column	Null?	Type
BD_ID	NOT NULL	NUMBER
BD_NAME	NOT NULL	VARCHAR2(100)
BD_AGE	-	NUMBER(2,0)
BD_SEX	-	VARCHAR2(10)
BD_BGROUP	-	VARCHAR2(10)
BD_REG_DATE	-	DATE
RECO_ID	NOT NULL	NUMBER
CITY_ID	NOT NULL	NUMBER

TABLE BLOODSPECIMEN

Column	Null?	Type
SPECIMEN_NUMBER	NOT NULL	NUMBER
B_GROUP	NOT NULL	VARCHAR2(10)
STATUS	-	NUMBER
DFIND_ID	NOT NULL	NUMBER
M_ID	NOT NULL	NUMBER

TABLE CITY

Column	Null?	Type
CITY_ID	NOT NULL	NUMBER
CITY_NAME	NOT NULL	VARCHAR2(50)

TABLE DISEASEFINDER

Column	Null?	Type
DFIND_ID	NOT NULL	NUMBER
DFIND_NAME	NOT NULL	VARCHAR2(30)
DFIND_PHNO	-	NUMBER(10,0)

TABLE HOSPITAL_INFO_1

Column	Null?	Type
HOSP_ID	NOT NULL	NUMBER
HOSP_NAME	NOT NULL	VARCHAR2(50)
CITY_ID	NOT NULL	NUMBER
M_ID	NOT NULL	NUMBER

TABLE HOSPITAL_INFO_2

Column	Null?	Type
HOSP_ID	NOT NULL	NUMBER
HOSP_NAME	NOT NULL	VARCHAR2(50)
HOSP_NEEDED_BGRP	NOT NULL	VARCHAR2(10)
HOSP_NEEDED_QNTY	-	NUMBER

TABLE RECIPIENT

Column	Null?	Type
RECI_ID	NOT NULL	NUMBER
RECI_NAME	NOT NULL	VARCHAR2(40)
RECI_AGE	-	VARCHAR2(2)
RECI_BRGP	-	VARCHAR2(3)
RECI_BQNTY	-	FLOAT(126)
RECO_ID	NOT NULL	NUMBER
CITY_ID	NOT NULL	NUMBER
M_ID	NOT NULL	NUMBER
RECI_SEX	-	VARCHAR2(10)
RECI_REG_DATE	-	DATE

TABLE RECORDING_STAFF

Column	Null?	Type
RECO_ID	NOT NULL	NUMBER
RECO_NAME	NOT NULL	VARCHAR2(30)
RECO_PHNO	-	NUMBER(10,0)

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.

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

BLOOD BANK MANAGER

M_ID	MNAME	M_PHNO
103	Peter	4693959601
104	Mark	4693959677
105	Jason	4693957671
106	Steve	4694959671
107	Jason	4695959671
108	Stella	4663959671
109	Monika	4673959671
110	John	4693859671
102	Jack	4693959671

BLOOD DONER

BD_ID	BD_NAME	BD_AGE	BD_SEX	BD_BGROUP	BD_REG_DATE	RECO_ID	CITY_ID
150221	Mark	25	M	B+	17-DEC-17	101212	1100
160011	Abdul	35	F	A+	22-NOV-16	101212	1100
160101	Smith	22	M	O+	04-JAN-16	101312	1200
150011	Pat	29	M	O+	19-JUL-15	101412	1300
150021	Shyam	42	F	A-	24-DEC-15	101412	1300
150121	Dan	44	M	AB+	28-AUG-15	101212	1200
160031	Mike	33	F	AB-	06-FEB-16	101212	1400
160301	Elisa	31	F	AB+	10-SEP-16	101312	1200
160091	Carrol	24	M	B-	15-OCT-16	101312	1500
160401	Mark	29	M	O-	17-DEC-16	101212	1200
150221	Mark	25	M	B+	17-DEC-17	101212	1100

BLOOD SPECIMEN

SPECIMEN_NUMBER	B_GROUP	STATUS	DFIND_ID	M_ID
1006	A-	1	13	104
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
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

CITY

CITY_ID	CITY_NAME
1300	Irving
1400	Houston
1500	Richardson
1600	Plano
1700	Frisco
1800	Arlington
1900	San Antonio
2000	Tyler
1200	Austin

DISEASE FINDER

DFIND_ID	DFIND_NAME	DFIND_PHNO
12	Park	4693804223
13	Jerry	4693804223
14	Mark	4693804223
15	Monika	4693804223
16	Ram	4693804123
17	Swathi	4693804223
18	Gautham	4693804323
19	Ashwin	4693804423
20	Yash	4693804523
11	Peter	4693804223

HOSPITAL INFO 1

HOSP_ID	HOSP_NAME	CITY_ID	M_ID
1	MayoClinic	1100	101
2	CleavelandClinic	1200	103
3	NYU	1300	103
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

HOSPITAL INFO 2

HOSP_ID	HOSP_NAME	HOSP_NEEDED_BGRP	HOSP_NEEDED_QNTY
1	MayoClinic	AB+	0
1	MayoClinic	A-	40
1	MayoClinic	B-	10
1	MayoClinic	AB-	20
2	CleavelandClinic	A+	40
2	CleavelandClinic	AB+	20
2	CleavelandClinic	A-	10
2	CleavelandClinic	B-	30
2	CleavelandClinic	B+	0
2	CleavelandClinic	AB-	10
3	NYU	A+	0
3	NYU	AB+	0
3	NYU	A-	0
3	NYU	B-	20
3	NYU	B+	10
3	NYU	AB-	0
4	Baylor	A+	10
5	Charlton	B+	30
4	Baylor	A-	40
7	Forestpark	B-	40
8	Parkland	B+	10
9	Pinecreek	AB-	20
1	MayoClinic	A+	20

RECIPIENT

RECI_ID	RECI_NAME	RECI_AGE	RECI_BRGP	RECI_BQNTY	RECO_ID	CITY_ID	M_ID	RECI_SEX	RECI_REG_DATE
10001	Mark	25	B+	1.5	101212	1100	101	M	17-DEC-15
10002	Dan	60	A+	1	101312	1100	102	M	16-DEC-15
10003	Steve	35	AB+	.5	101312	1200	102	M	17-OCT-15
10004	Parker	66	B+	1	101212	1300	104	M	17-NOV-16
10005	Jason	53	B-	1	101412	1400	105	M	17-APR-15
10006	Preetham	45	O+	1.5	101512	1500	105	M	17-DEC-15
10007	Swetha	22	AB-	1	101212	1500	101	F	17-MAY-15
10008	Swathi	25	B+	2	101412	1300	103	F	14-DEC-15
10009	Lance	30	A+	1.5	101312	1100	104	M	16-FEB-15
10010	Marsh	25	AB+	3.5	101212	1200	107	M	17-OCT-16

RECORDING STAFF

RECO_ID	RECO_NAME	RECO_PHNO
101212	Walcot	4045806553
101312	Henry	4045806553
101412	Silva	4045806553
101512	Adrian	4045806553
101612	Mark	4045806553
101712	Abdul	4045816553
101812	Jerry	4045826553
101912	Tim	4045836553
101012	Lekha	4044846553
101112	Mark	4045856553

SQL QUERIES

1. **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;

SPECIMEN_NUMBER	B_GROUP
1012	O-
1013	B-

2. **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 ;

M_ID	MNAME	RECI_NAME	RECI_BRGP	B_GROUP
103	Peter	Mark	B+	B+
104	Mark	Mark	B+	B+
105	Jason	Mark	B+	B+
106	Steve	Mark	B+	B+
107	Jason	Mark	B+	B+
108	Stella	Mark	B+	B+
109	Monika	Mark	B+	B+
110	John	Mark	B+	B+
102	Jack	Mark	B+	B+
103	Peter	Steve	AB+	AB+
104	Mark	Steve	AB+	AB+
105	Jason	Steve	AB+	AB+
106	Steve	Steve	AB+	AB+
107	Jason	Steve	AB+	AB+
108	Stella	Steve	AB+	AB+
109	Monika	Steve	AB+	AB+
110	John	Steve	AB+	AB+
102	Jack	Steve	AB+	AB+

3. **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 ;

BD_ID	BD_NAME	RECI_ID	RECI_NAME
150221	Mark	10001	Mark
160011	Abdul	10002	Dan
160011	Abdul	10009	Lance
150121	Dan	10003	Steve
150121	Dan	10010	Marsh
160301	Elisa	10003	Steve
160301	Elisa	10010	Marsh
150221	Mark	10001	Mark

4. **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;

HOSP_ID	HOSP_NAME	CITY_ID	M_ID
2	CleavelandClinic	1200	103
3	NYU	1300	103
5	Charlton	1800	103

5. **Display the names of hospitals managed by Peter**

Select hosp_name from Hospital_Info_1,BB_Manager where BB_Manager.M_id=Hospital_Info_1.M_id and BB_Manager.Mname='Peter';

3	NYU	1300
5	Charlton	1800

6. **List all donars who are females and are above 30 years of age**

Select bd_id,bd_name from blood_donor

WHERE bd_sex='F' and bd_age > 30;

BD_ID	BD_NAME
160011	Abdul
150021	Shyam
160031	Mike
160301	Elisa

7. **Find all donars and recipients who belong to Houston**

SELECT bd_name as donors,reci_name as recipients,city_name from

blood_donor d,recipient r,city c

WHERE d.city_id = c.city_id and c.city=r.city and city_name='Houston';

DONORS	RECIPIENTS	CITY_NAME
Mike	Jason	Houston

8. **Find all donars and recipients that come under recording staff member Henry**

```
SELECT bd_name as donors,reci_name as recipients,reco_name
from blood_donor d,recipient r,Recording_Staff s
WHERE d.reco_id=s.reco_id and s.reco_id=r.reco_id and reco_name='Henry';
```

DONORS	RECIPIENTS	RECO_NAME
Smith	Dan	Henry
Elisa	Dan	Henry
Carrol	Dan	Henry
Smith	Steve	Henry
Elisa	Steve	Henry
Carrol	Steve	Henry
Smith	Lance	Henry
Elisa	Lance	Henry
Carrol	Lance	Henry

9. **Find the hospital that requires maximum amout of blood**

```
select max (hosp_needed_qnty),hosp_id,hosp_name
from Hospital_Info_2
group by hosp_id,hosp_name;
```

MAX(HOSP_NEEDED_QNTY)	HOSP_ID	HOSP_NAME
40	2	CleavelandClinic

10. **Find name of universal donars**

```
SELECT COUNT(*),bd_name,bd_bgroup FROM blood_donor
where bd_bgroup='O+'
group by (bd_name,bd_bgroup);
```

COUNT(*)	BD_NAME	BD_BGROUP
1	Pat	O+
1	Smith	O+

PL/SQL QUERIES

1. **Display the total number of donors using concept of stored functions**

```
CREATE OR REPLACE FUNCTION totalDonar
RETURN number IS
 total integer := 0;
BEGIN
 SELECT count(*) into total
 FROM Blood_Donor;
RETURN total;
END;
```

```
DECLARE
total_no NUMBER;
BEGIN
total_no :=totalDonar;
dbms_output.put_line('Total no. of donars : ' ||total_no);
END;
```

```
Function created.
```

```
Statement processed.
Total no. of donars : 10
```

2. **Display any update in donor's city using trigger**

```
create or replace trigger city_change
before update on Blood_Donor
for each row
when(NEW.city_id!=OLD.city_id)
declare

begin
 dbms_output.put_line('Old location : '|| :OLD.city_id);
 dbms_output.put_line('New location: '|| :NEW.city_id);
end;
UPDATE Blood_Donor
SET city_id=1300 where bd_name='Smith';
```

```
Trigger created.
```

```
1 row(s) updated.
Old location : 1200
New location : 1300
```

3.**Write a PL/SQL code to display the name of top 5 hospitals in decreasing order of their quantity of blood needed using cursor**

```
DECLARE
 CURSOR C1 IS SELECT * FROM Hospital_Info_2 ORDER BY hosp_needed_qnty DESC;
 rec C1%ROWTYPE;

BEGIN
 OPEN C1;
 LOOP
  FETCH C1 INTO rec;
  EXIT WHEN C1%ROWCOUNT > 5 OR C1%NOTFOUND;
  dbms_output.put_line('Hospital name : '||rec.hosp_name|| ' Quantity needed : '||rec.hosp_needed_qnty ||' units');
 END LOOP;
 CLOSE C1;
END;
```

```
Statement processed.
Hospital name : CleavelandClinic    Quantity needed : 40 units
Hospital name : Forestpark    Quantity needed : 40 units
Hospital name : Baylor    Quantity needed : 40 units
Hospital name : MayoClinic    Quantity needed : 40 units
Hospital name : Charlton    Quantity needed : 30 units
```

4**. Display details of all recipients whose blood group is "B+" and its value is passed as a parameter using cursor**

```
DECLARE

 CURSOR C1(b varchar)IS SELECT * FROM recipient WHERE reci_Brgp =b;
```

```
BEGIN
 FOR rec IN C1('B+') LOOP
 dbms_output.put_line('Recipient id : '||rec.reci_ID||'  Recipient name : '||rec.reci_name||
 '  Age : '||rec.reci_age||'  Blood group : ' ||rec.reci_Brgp||'  Registration date : '||rec.reci_reg_date);
 END LOOP;

END;
```

```
Statement processed.
Recipient id : 10001  Recipient name : Mark  Age : 25  Blood group : B+  Registration date : 17-DEC-15
Recipient id : 10004  Recipient name : Parker  Age : 66  Blood group : B+  Registration date : 17-NOV-16
Recipient id : 10008  Recipient name : Swathi  Age : 25  Blood group : B+  Registration date : 14-DEC-15
```

5.**Create a stored procedure to insert a record in DiseaseFinder table.Pass the values of columns to this procedure and it should insert a record with passed parameter values**

```
CREATE OR REPLACE PROCEDURE insert_diseaseFinder(dfind_ID IN NUMBER,dfind_name in CHAR,dfind_PhNo NUMBER)IS
BEGIN
INSERT INTO DiseaseFinder VALUES(dfind_ID,dfind_name,dfind_PhNo);
END;

BEGIN
insert_diseaseFinder(21,'Samisha',9124311837);
END;
SELECT * FROM DiseaseFinder ;
```

```
Procedure created.

Statement processed.
```

DFIND_ID	DFIND_NAME	DFIND_PHNO
11	Peter	4693804223
12	Park	4693804223
13	Jerry	4693804223
14	Mark	4693804223
15	Monika	4693804223
16	Ram	4693804123
17	Swathi	4693804223
18	Gautham	4693804323
19	Ashwin	4693804423
20	Yash	4693804523
21	Samisha	9124311837

6.**Create a trigger so that no operation should be performed on Donor table on Friday**

```
CREATE OR REPLACE TRIGGER holiday
BEFORE
INSERT OR DELETE OR UPDATE
ON blood_donor
DECLARE
 DAY char(10);
BEGIN
 DAY := TRIM(to_char(SYSDATE,'Day'));
 dbms_output.put_line('Today is : '||DAY);
 IF DAY = 'Friday' THEN
 RAISE_APPLICATION_ERROR(-20022,'Sorry ! No operation can be performed on donor table on Friday');
END IF;
END;
INSERT into Blood_Donor VALUES(1701,'Joey',55,'M','B+','17-DEC-2017',101212,1100);
```

```
Trigger created.
```

```
ORA-20022: Sorry ! No operation can be performed on donor table on Friday ORA-06512: at "SQL_CRUWZIJYNOAVWFNCZTMNYBYUS.HOLIDAY", line 7
ORA-06512: at "SYS.DBMS_SQL", line 1721
```

DATABASE

```
CREATE TABLE BB_Manager ( M_id NUMBER(10) NOT NULL,mName varchar2(100) NOT NULL,m_phNo NUMBER(10)
);INSERT into BB_Manager VALUES(102,'Jack', 4693959671);

INSERT into BB_Manager VALUES(103,'Peter', 4693959601);

INSERT into BB_Manager VALUES(104,'Mark', 4693959677);

INSERT into BB_Manager VALUES(105,'Jason', 4693957671);

INSERT into BB_Manager VALUES(106,'Steve', 4694959671);

INSERT into BB_Manager VALUES(107,'Jason', 4695959671);

INSERT into BB_Manager VALUES(108,'Stella', 4663959671);

INSERT into BB_Manager VALUES(109,'Monika', 4673959671);

INSERT into BB_Manager VALUES(110,'John', 4693859671);

select * from BB_Manager;
```

```
Create table Blood_Donor

(bd_ID int NOT NULL, bd_name varchar(100) NOT NULL, bd_age number(2),bd_sex varchar(10),bd_Bgroup varchar(10),

bd_reg_date date,reco_ID int NOT NULL,City_ID int NOT NULL );

INSERT into Blood_Donor VALUES(150221,'Mark',25,'M','B+','17-DEC-2017',101212,1100);

INSERT into Blood_Donor VALUES(160011,'Abdul',35,'F','A+','22-NOV-2016',101212,1100);

INSERT into Blood_Donor VALUES(160101,'Smith',22,'M','O+','04-JAN-2016',101312,1200);

INSERT into Blood_Donor VALUES(150011,'Pat',29,'M','O+','19-JUL-2015',101412,1300);

INSERT into Blood_Donor VALUES(150021,'Shyam',42,'F','A-','24-DEC-2015',101412,1300);

INSERT into Blood_Donor VALUES(150121,'Dan',44,'M','AB+','28-AUG-2015',101212,1200);

INSERT into Blood_Donor VALUES(160031,'Mike',33,'F','AB-','06-FEB-2016',101212,1400);

INSERT into Blood_Donor VALUES(160301,'Elisa',31,'F','AB+','10-SEP-2016',101312,1200);

INSERT into Blood_Donor VALUES(160091,'Carrol',24,'M','B-','15 OCT-2016',101312,1500);

INSERT into Blood_Donor VALUES(160401,'Mark',29,'M','O-','17-DEC-2016',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);
INSERT into BloodSpecimen VALUES(1002, 'O+', 1,12,102);
INSERT into BloodSpecimen VALUES(1003, 'AB+', 1,11,102);
INSERT into BloodSpecimen VALUES(1004, 'O-', 1,13,103);
INSERT into BloodSpecimen VALUES(1005, 'A+', 0,14,101);
INSERT into BloodSpecimen VALUES(1006, 'A-', 1,13,104);
INSERT into BloodSpecimen VALUES(1007, 'AB-', 1,15,104);
INSERT into BloodSpecimen VALUES(1008, 'AB-', 0,11,105);
INSERT into BloodSpecimen VALUES(1009, 'B+', 1,13,105);
INSERT into BloodSpecimen VALUES(1010, 'O+', 0,12,105);
INSERT into BloodSpecimen VALUES(1011, 'O+', 1,13,103);
INSERT into BloodSpecimen VALUES(1012, 'O-', 1,14,102);
INSERT into BloodSpecimen VALUES(1013, 'B-', 1,14,102);
INSERT into BloodSpecimen VALUES(1014, 'AB+', 0,15,101);
Select * from BloodSpecimen


CREATE TABLE City (City_ID int NOT NULL,City_name varchar(50) NOT NULL);
INSERT into City VALUES(1200,'Austin');
INSERT into City VALUES(1300,'Irving');
INSERT into City VALUES(1400,'Houston');
INSERT into City VALUES(1500,'Richardson');
INSERT into City VALUES(1600,'Plano');
INSERT into City VALUES(1700,'Frisco');
INSERT into City VALUES(1800,'Arlington');
INSERT into City VALUES(1900,'San Antonio');
INSERT into City VALUES(2000,'Tyler');
select * from City;
```

```
CREATE TABLE DiseaseFinder(dfind_ID int NOT NULL,dfind_name varchar(30) NOT NULL,dfind_PhNo NUMBER(10));

INSERT into DiseaseFinder VALUES(11,'Peter',4693804223);

INSERT into DiseaseFinder VALUES(12,'Park',4693804223);

INSERT into DiseaseFinder VALUES(13,'Jerry',4693804223);

INSERT into DiseaseFinder VALUES(14,'Mark',4693804223);

INSERT into DiseaseFinder VALUES(15,'Monika',4693804223);

INSERT into DiseaseFinder VALUES(16,'Ram',4693804123);

INSERT into DiseaseFinder VALUES(17,'Swathi',4693804223);

INSERT into DiseaseFinder VALUES(18,'Gautham',4693804323);

INSERT into DiseaseFinder VALUES(19,'Ashwin',4693804423);

INSERT into DiseaseFinder VALUES(20,'Yash',4693804523);

select * from DiseaseFinder;

drop table DiseaseFinder
```

```
CREATE TABLE Hospital_Info_1  (hosp_ID int NOT NULL,hosp_name varchar(50) NOT NULL, City_ID int NOT NULL,M_id
int NOT NULL,

primary key(hosp_ID));

INSERT into Hospital_Info_1 VALUES(1,'MayoClinic',1100,101);

INSERT into Hospital_Info_1 VALUES(2,'CleavelandClinic',1200,103);

INSERT into Hospital_Info_1 VALUES(3,'NYU',1300,103);

INSERT into Hospital_Info_1 VALUES(4,'Baylor',1400,104);

INSERT into Hospital_Info_1 VALUES(5,'Charlton',1800,103);

INSERT into Hospital_Info_1 VALUES(6,'Greenoaks',1300,106);

INSERT into Hospital_Info_1 VALUES(7,'Forestpark',1300,102);

INSERT into Hospital_Info_1 VALUES(8,'Parkland',1200,106);

INSERT into Hospital_Info_1 VALUES(9,'Pinecreek',1500,109);

INSERT into Hospital_Info_1 VALUES(10,'WalnutHill',1700,105);

select * from  Hospital_Info_1;
```

```
CREATE TABLE Hospital_Info_2 ( hosp_ID int NOT NULL,   hosp_name varchar(50) 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);
INSERT into Hospital_Info_2 VALUES(1,'MayoClinic','AB+',0);
INSERT into Hospital_Info_2 VALUES(1,'MayoClinic','A-',40);
INSERT into Hospital_Info_2 VALUES(1,'MayoClinic','B-',10);
INSERT into Hospital_Info_2 VALUES(1,'MayoClinic','AB-',20);
INSERT into Hospital_Info_2 VALUES(2,'CleavelandClinic','A+',40);
INSERT into Hospital_Info_2 VALUES(2,'CleavelandClinic','AB+',20);
INSERT into Hospital_Info_2 VALUES(2,'CleavelandClinic','A-',10);
INSERT into Hospital_Info_2 VALUES(2,'CleavelandClinic','B-',30);
INSERT into Hospital_Info_2 VALUES(2,'CleavelandClinic','B+',0);
INSERT into Hospital_Info_2 VALUES(2,'CleavelandClinic','AB-',10);
INSERT into Hospital_Info_2 VALUES(3,'NYU','A+',0);
INSERT into Hospital_Info_2 VALUES(3,'NYU','AB+',0);
INSERT into Hospital_Info_2 VALUES(3,'NYU','A-',0);
INSERT into Hospital_Info_2 VALUES(3,'NYU','B-',20);
INSERT into Hospital_Info_2 VALUES(3,'NYU','B+',10);
INSERT into Hospital_Info_2 VALUES(3,'NYU','AB-',0);
INSERT into Hospital_Info_2 VALUES(4,'Baylor','A+',10);
INSERT into Hospital_Info_2 VALUES(5,'Charlton','B+',30);
INSERT into Hospital_Info_2 VALUES(4,'Baylor','A-',40);
INSERT into Hospital_Info_2 VALUES(7,'Forestpark','B-',40);
INSERT into Hospital_Info_2 VALUES(8,'Parkland','B+',10);
INSERT into Hospital_Info_2 VALUES(9,'Pinecreek','AB-',20);
select * from Hospital_Info_2 ;
```

```
CREATE TABLE Recipient ( reci_ID int NOT NULL,reci_name varchar(40) NOT NULL,reci_age varchar(2),reci_Brgp varchar(3),
 reci_Bqnty float,reco_ID int NOT NULL,City_ID int NOT NULL,M_id int NOT NULL,reci_sex varchar(10),reci_reg_date date);
INSERT into Recipient VALUES(10001,'Mark',25,'B+',1.5,101212,1100,101,'M','17-DEC-2015');
INSERT into Recipient VALUES(10002,'Dan',60,'A+',1,101312,1100,102,'M','16-DEC-2015');
INSERT into Recipient VALUES(10003,'Steve',35,'AB+',0.5,101312,1200,102,'M','17-OCT-2015');
INSERT into Recipient VALUES(10004,'Parker',66,'B+',1,101212,1300,104,'M','17-NOV-2016');
INSERT into Recipient VALUES(10005,'Jason',53,'B-',1,101412,1400,105,'M','17-APR-2015');
INSERT into Recipient VALUES(10006,'Preetham',45,'O+',1.5,101512,1500,105,'M','17-DEC-2015');
INSERT into Recipient VALUES(10007,'Swetha',22,'AB-',1,101212,1500,101,'F','17-MAY-2015');
INSERT into Recipient VALUES(10008,'Swathi',25,'B+',2,101412,1300,103,'F','14-DEC-2015');
INSERT into Recipient VALUES(10009,'Lance',30,'A+',1.5,101312,1100,104,'M','16-FEB-2015');
INSERT into Recipient VALUES(10010,'Marsh',25,'AB+',3.5,101212,1200,107,'M','17-OCT-2016');
select * from Recipient;



CREATE TABLE Recording_Staff(reco_ID int NOT NULL, reco_Name varchar(30) NOT NULL, reco_phNo NUMBER(10));
INSERT into Recording_Staff VALUES(101212,'Walcot',4045806553);
INSERT into Recording_Staff VALUES(101312,'Henry',4045806553);
INSERT into Recording_Staff VALUES(101412,'Silva',4045806553);
INSERT into Recording_Staff VALUES(101512,'Adrian',4045806553);
INSERT into Recording_Staff VALUES(101612,'Mark',4045806553);
INSERT into Recording_Staff VALUES(101712,'Abdul',4045816553);
INSERT into Recording_Staff VALUES(101812,'Jerry',4045826553);
INSERT into Recording_Staff VALUES(101912,'Tim',4045836553);
INSERT into Recording_Staff VALUES(101012,'Lekha',4044846553);
INSERT into Recording_Staff VALUES(101112,'Mark',4045856553);
select * from Recording_Staff;
```

THANK YOU