**ACKNOWLEDGMENT:**

We would like to express our special thanks of gratitude to our mentor Dr.Debanjan Sadhya who gave us the golden opportunity to do this wonderful project on the topic Blood Donation DataBase, which also helped me in doing a lot of Research and I came to know about so many new things I am really thankful to them. Secondly we would also like to thank our batchmates and friends who helped me a lot in finalizing this project within the limited time frame. Also the coordination among us helped us to complete this project the way we wished.

**SCOPE OF THE PROJECT :**

Database Management systems are widely used by companies and organizations to maintain and manage their knowledge and information resources. Managing our data in a proper way to use it efficiently later is the major factor for a database. Our blood donation Database helps to store information about the people involved in a particular task called blood donation. It includes nurses, patients , donors , types of donations. It is as simple as having a right detailed page of the transactions that happened and are going to be happening.

**ABOUT :**

A blood donation database plans to implement a database to enhance its data management practice and ultimately advance its business operations.

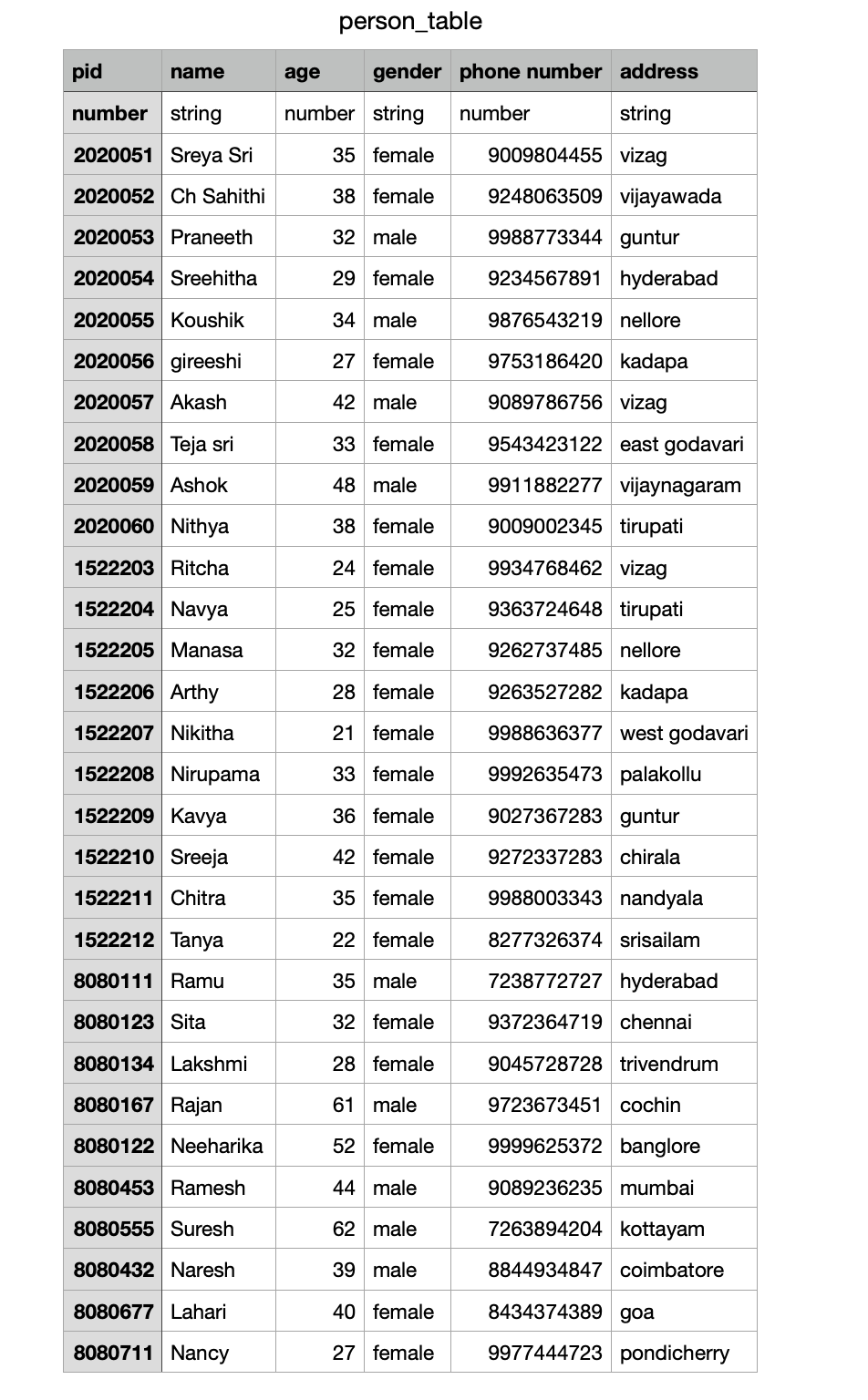
The initial planning analysis phases have revealed the following system requirements:

* Each person has a unique person ID as well as the following attributes:

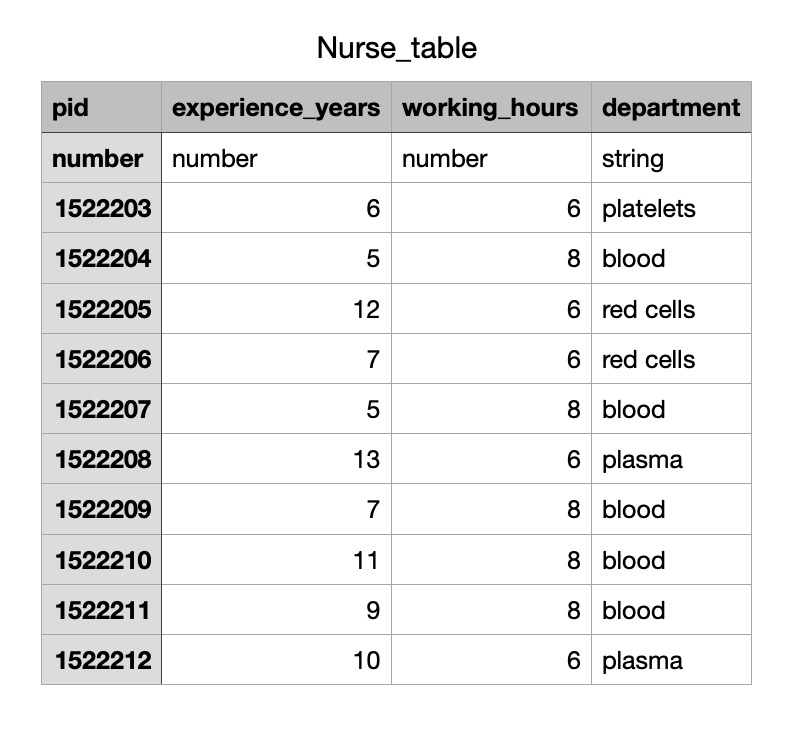
Name, age, gender, phone number, address.

* Donors, Patients, and nurses are identified by their unique IDs.
* Donor's name, age, weight, blood pressure, sugar levels are stored in the Donor entity.
* The patient's entity contains the required type and the requested date for the blood.
* The nurse entity contains their working experience, working hours, and the department in which they work.
* There are four types of donation : Blood, Plasma, Platelets, Red blood cells.
* Donor name, Patient name, amount of blood are the attributes in the Donation entity.
* The quantity of blood available and the blood groups' records are stored in the blood bags table.
* The patient needs to appeal for blood beforehand and provide their ID, location, blood type required, and quantity.
* So, the above information is stored in the request entity.
* All the donation records such as Donor ID, Patient ID, date of donation is also stored.
* The donation record entity contains such information.

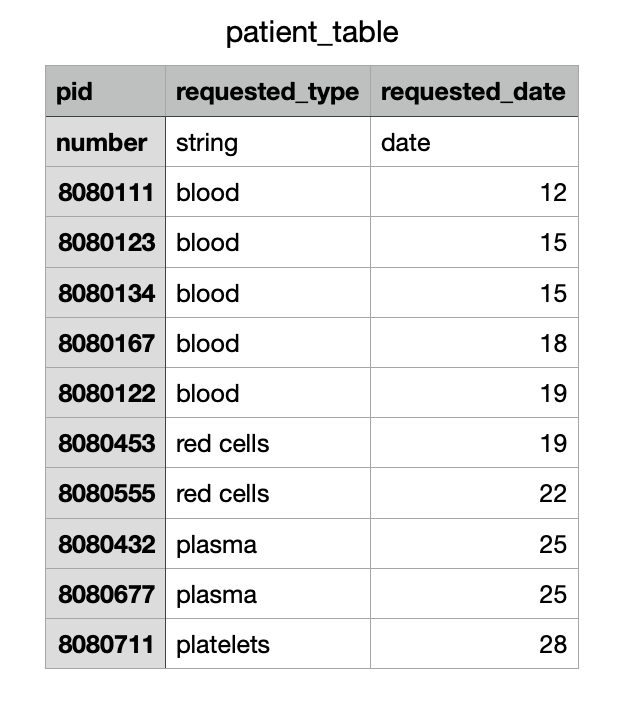
**OUR DATABASE ENTITIES AND TABLES :**



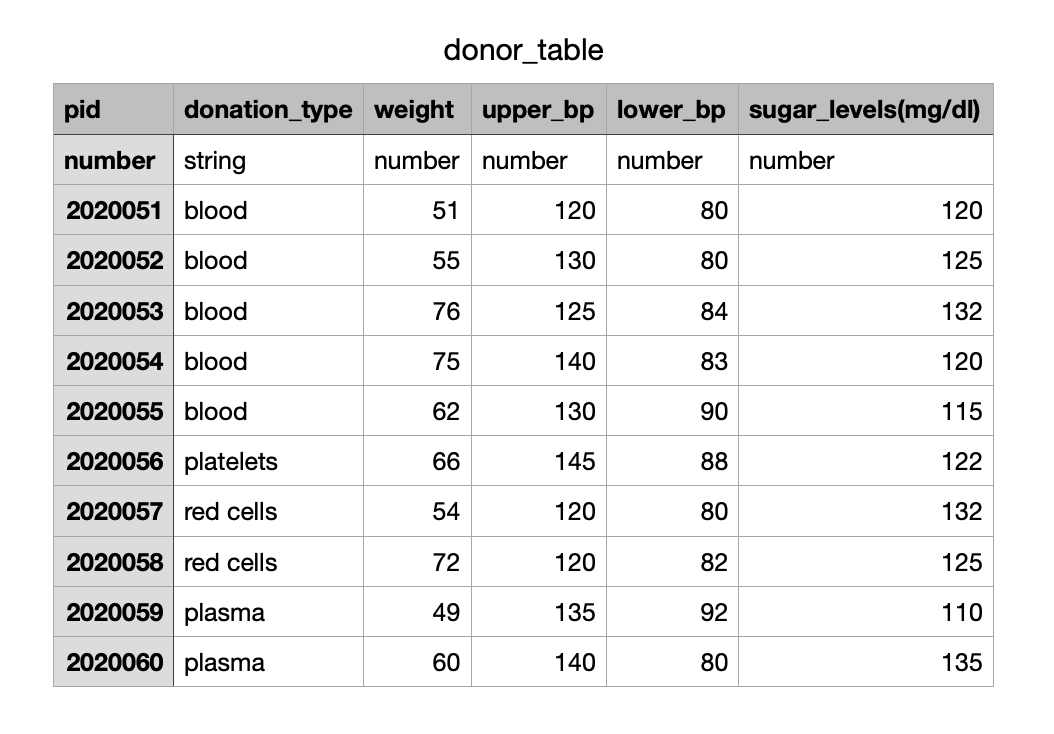
**NURSE TABLE :**



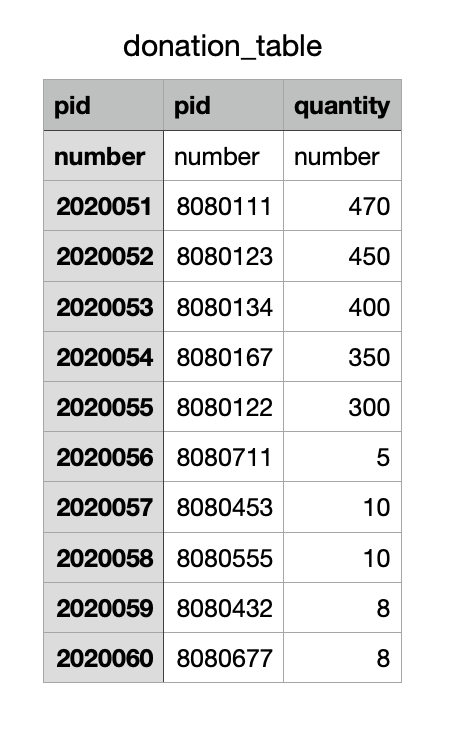
**PATIENT TABLE :**



**DONOR TABLE :**



**DONATION TABLE :**



**THE EXECUTIONS WE USED FOR RA AND SQL** :

group:PROJECT

person\_table = {

pid, name, age, gender, phonenumber, address

2020051, 'Sreya Sri', 35, 'female', 9009804455, 'vizag'

2020052, 'Ch Sahithi', 38, 'female', 9248063509, 'vijayawada'

2020053, 'karthik V', 32, 'male', 9988773344, 'guntur'

2020054, 'Sreehitha', 29, 'female', 9234567891, 'hyderabad'

2020055, 'Koushik', 34, 'male', 9876543219, 'nellore'

2020056, 'gireeshe', 27, 'female', 9753186420, 'kadapa'

2020057, 'Akash', 42, 'male', 9089786756, 'vizag'

2020058, 'Teja sri', 33, 'female', 9543423122, 'east godavari'

2020059, 'Ashok', 48, 'male', 9911882277, 'vijaynagaram'

2020060, 'Nithya', 38, 'female', 9009002345, 'tirupati'

1522203, 'Ritcha', 24, 'female', 9934768462, 'vizag'

1522204, 'Navya', 25, 'female', 9363724648, 'tirupati'

1522205, 'Manasa', 32, 'female', 9262737485, 'nellore'

1522206, 'Arthy', 28, 'female', 9263527282, 'kadapa'

1522207, 'Nikitha', 21, 'female', 9988636377, 'west godavari'

1522208, 'Nirupama', 33, 'female', 9992635473, 'palakollu'

1522209, 'Kavya', 36, 'female', 9027367283, 'guntur'

1522210, 'Sreeja', 42, 'female', 9272337283, 'chirala'

1522211, 'Chitra', 35, 'female', 9988003343, 'nandyala'

1522212, 'Tanya', 22, 'female', 8277326374, 'srisailam'

8080111, 'Ramu', 35, 'male', 7238772727, 'hyderabad'

8080123, 'Sita', 32, 'female', 9372364719, 'chennai'

8080134, 'Lakshmi', 28, 'female', 9045728728, 'trivendrum'

8080167, 'Rajan', 61, 'male', 9723673451, 'cochin'

8080122, 'Neeharika', 52, 'female', 9999625372, 'banglore'

8080453, 'Ramesh', 44, 'male', 9089236235, 'mumbai'

8080555, 'Suresh', 62, 'male', 7263894204, 'kottayam'

8080432, 'Naresh', 39, 'male', 8844934847, 'coimbatore'

8080677, 'Lahari', 40, 'female', 8434374389, 'goa'

8080711, 'Nancy', 27, 'female', 9977444723, 'pondicherry'

}

Nurse\_table = {

pid, experience\_years, working\_hours, department

1522203, 6, 6, 'platelets'

1522204, 5, 8, 'blood'

1522205, 12, 6, 'red cells'

1522206, 7, 6, 'red cells'

1522207, 5, 8, 'blood'

1522208, 13, 6, 'plasma'

1522209, 7, 8, 'blood'

1522210, 11, 8, 'blood'

1522211, 9, 8, 'blood'

1522212, 10, 6, 'plasma'

}

patient\_table = {

ptid, requested\_type, requested\_date

8080111, 'blood', '12'

8080123, 'blood', '15'

8080134, 'blood', '15'

8080167, 'blood', '18'

8080122, 'blood', '19'

8080453, 'red cells', '19'

8080555, 'red cells', '22'

8080432, 'plasma', '25'

8080677, 'plasma', '25'

8080711, 'platelets', '28'

}

donor\_table = {

pid, donation\_type, weight, upper\_bp, lower\_bp, sugar\_levels

2020051, 'blood', 51, 120, 80, 120

2020052, 'blood', 55, 130, 80, 125

2020053, 'blood', 76, 125, 84, 132

2020054, 'blood', 75, 140, 83, 120

2020055, 'blood', 62, 130, 90, 115

2020056, 'platelets', 66, 145, 88, 122

2020057, 'red cells', 54, 120, 80, 132

2020058, 'red cells', 72, 120, 82, 125

2020059, 'plasma', 49, 135, 92, 110

2020060, 'plasma', 60, 140, 80, 135

}

donation\_table = {

pid, ptid, quantity

2020051, 8080111, 470

2020052, 8080123, 450

2020053, 8080134, 400

2020054, 8080167, 350

2020055, 8080122, 300

2020056, 8080711, 5

2020057, 8080453, 10

2020058, 8080555, 10

2020059, 8080432, 8

2020060, 8080677, 8

}

Donationrecords = {

DonorID, patientID, date\_of\_donation

2020051, 8080432, '26072021'

2020052, 8080167, '20072021'

2020053, 8080134, '16072021'

2020054, 8080111, '14072021'

2020055, 8080123, '15072021'

2020056, 8080677, '26072021'

2020057, 8080711, '28072021'

2020058, 8080122, '20072021'

2020059, 8080453, '19072021'

2020060, 8080555, '22072021'

}

donation\_type = {

Dtype, frequency

'plasma', 28

'platelets', 7

'red cells', 112

'whole blood', 56

}

request\_table = {

ptid, lid, required\_type, date, quantity

8080111, 1050, 'blood', '12072021', 470

8080123, 1060, 'blood', '15072021', 450

8080134, 1043, 'blood', '15072021', 400

8080167, 1023, 'blood', '18072021', 350

8080122, 1044, 'blood', '19072021', 300

8080453, 1098, 'red cells', '19072021', 10

8080555, 1072, 'red cells', '22072021', 10

8080432, 1053, 'plasma', '25072021', 8

8080677, 1051, 'plasma', '25072021', 8

8080711, 1098, 'platelets', '28072021', 5

}

locationtable = {

Lid, citycode, cityname, hospitalname, hospitalcode

1050, 'MBI', 'mumbai', 'sunshine hospital', 'SSH'

1060, 'DEL', 'delhi', 'yashoda hospital', 'YSDH'

1051, 'BGLR', 'banglore', 'Mahatma Gandhi hospital', 'MGH'

1053, 'HYD', 'hyderabad', 'osmania hospitals', 'OH'

1098, 'CHNI', 'chennai', 'KIIMS hospital', 'KIIMSH'

1043, 'KLKTA', 'kolkata', 'Mamatha hospital', 'MH'

1023, 'GWLR', 'gwalior', 'Vasan hospital', 'VH'

1044, 'LCKW', 'lucknow', 'Apollo hospital', 'APLH'

1098, 'CHNDGR', 'chandigarh', 'Care hospital', 'CRH'

1072, 'ALHBD', 'allahabad', 'Rainbow hospital', 'RNBH'

}

bloodbagstable = {

B\_groups, quantityavailable

'A+', 12000

'A-', 11000

'B+', 8000

'B-', 6000

'O+', 18000

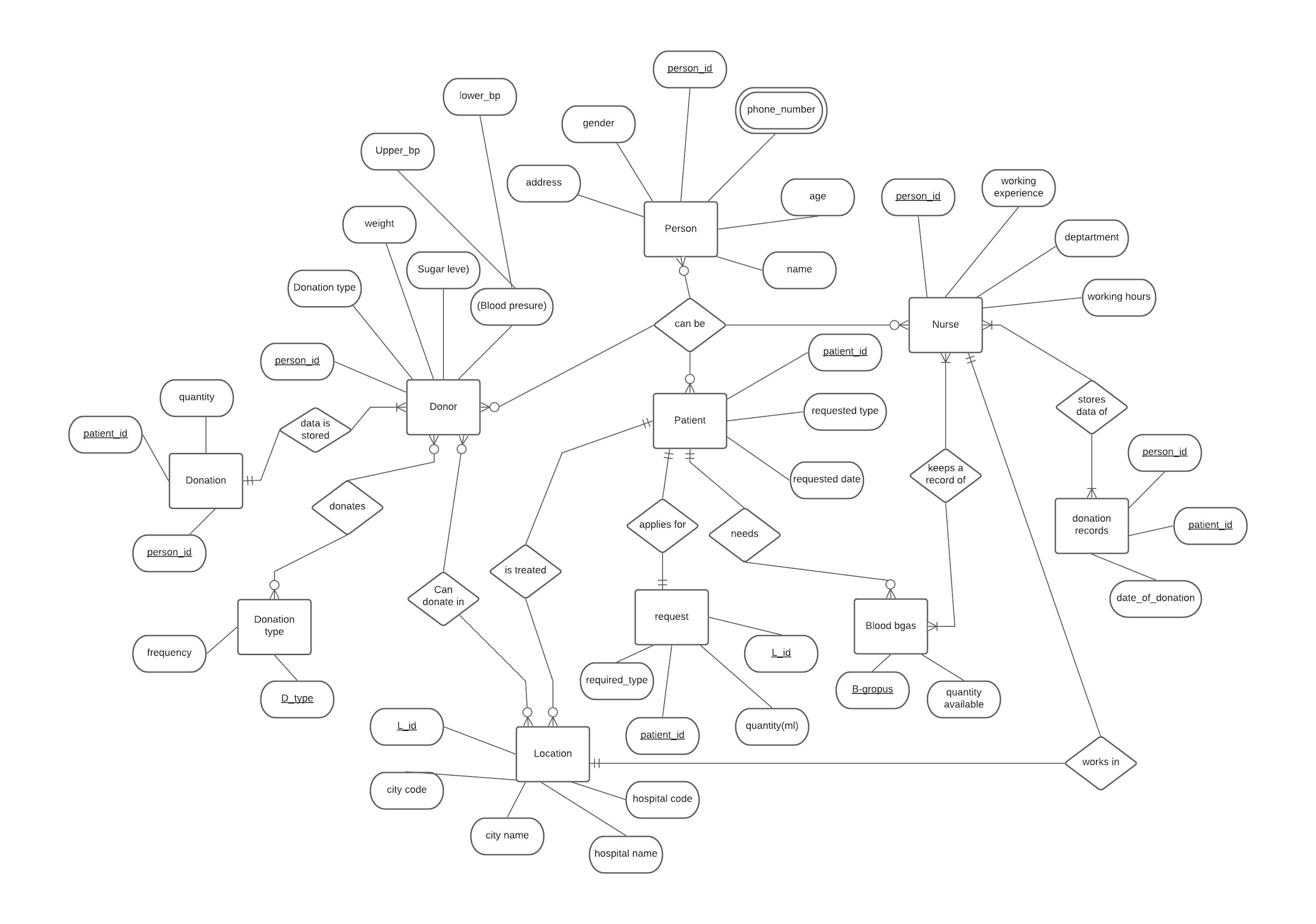
'O-', 9000

'AB+', 11000

'AB-', 5600

}

ENTITY-RELATIONSHIP DIAGRAM



The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At the view level, the ER model is considered a good option for designing databases.

The ER diagram consists of the entities, their attributes, and the relationship between the entities.

Entity: An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. We took Person, Donor, Patient, Nurse, Donation, Donation type, Blood bags, Location, Request, Donation records as entities in our project.

* They are represented by rectangular blocks.

Attributes: Entities are represented by means of their properties, called attributes.

* All attributes have values.
* There are different types of attributes:

1. Simple attribute
2. Composite attribute
3. Derived attribute
4. Single-value attribute
5. Multi-value attribute

* Some of the attributes in the ER diagram are Person ID, Donation type, Requested type, Address, Amount of blood, etc.
* They are represented by ellipses.

Keys : Key is an attribute or collection of attributes that uniquely identifies an entity among an entity set.

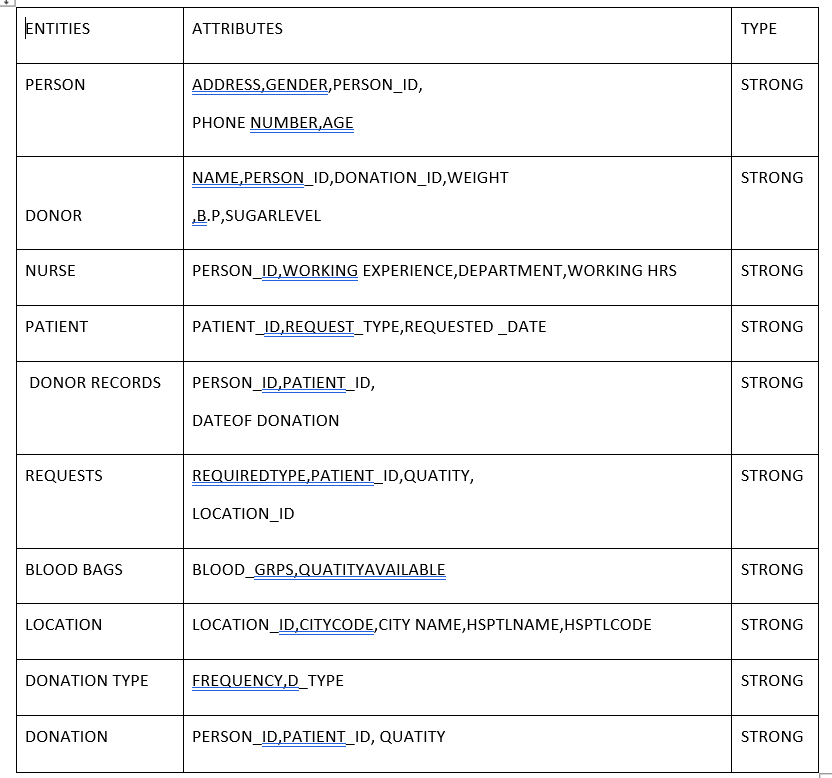
* We have different types of Keys which are:
  1. Super Key
  2. Candidate Key
  3. Primary Key
* In the ER model, the attributes that are Primary keys are underlined.

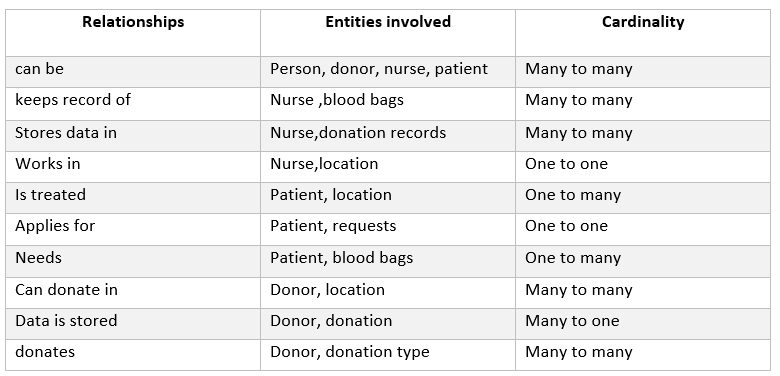
Relationship: The association among entities is called a relationship.

* Relationships can also be categorized based on it’s degree as:

1. Binary = degree 2
2. Ternary = degree 3
3. n-ary = degree n

**ENTITY SETS**



**RELATIONSHIP SETS**

**CONVERSION OF ER DIAGRAM INTO RELATIONAL SCHEMA**

ER Model, when conceptualized into diagrams, gives a good overview of entity-relationship, which is easier to understand.

ER diagrams can be mapped to relational schema, that is, it is possible to create relational schema using ER diagram. We cannot import all the ER constraints into a relational model, but an approximate schema can be generated.

We can easily achieve this by the Mapping process.

1. Mapping Entities:

The algorithm behind this is

• Create tables for each entity.

• Entity's attributes should become fields of tables with their respective data types.

• Declare primary key.

2. Mapping Relationship

It can be done by:

• Create a table for a relationship.

• Add the primary keys of all participating Entities as fields of table with their respective data types.

• If a relationship has any attribute, add each attribute as field of table.

• Declare a primary key composing all the primary keys of participating entities.

• Declare all foreign key constraints.

3. Mapping Weak Entity Sets

The steps are:

• Create table for weak entity set.

• Add all its attributes to table as field.

• Add the primary key of identifying entity set.

• Declare all foreign key constraints.

4. Mapping Hierarchical Entities

This includes:

• Create tables for all higher-level entities.

• Create tables for lower-level entities.

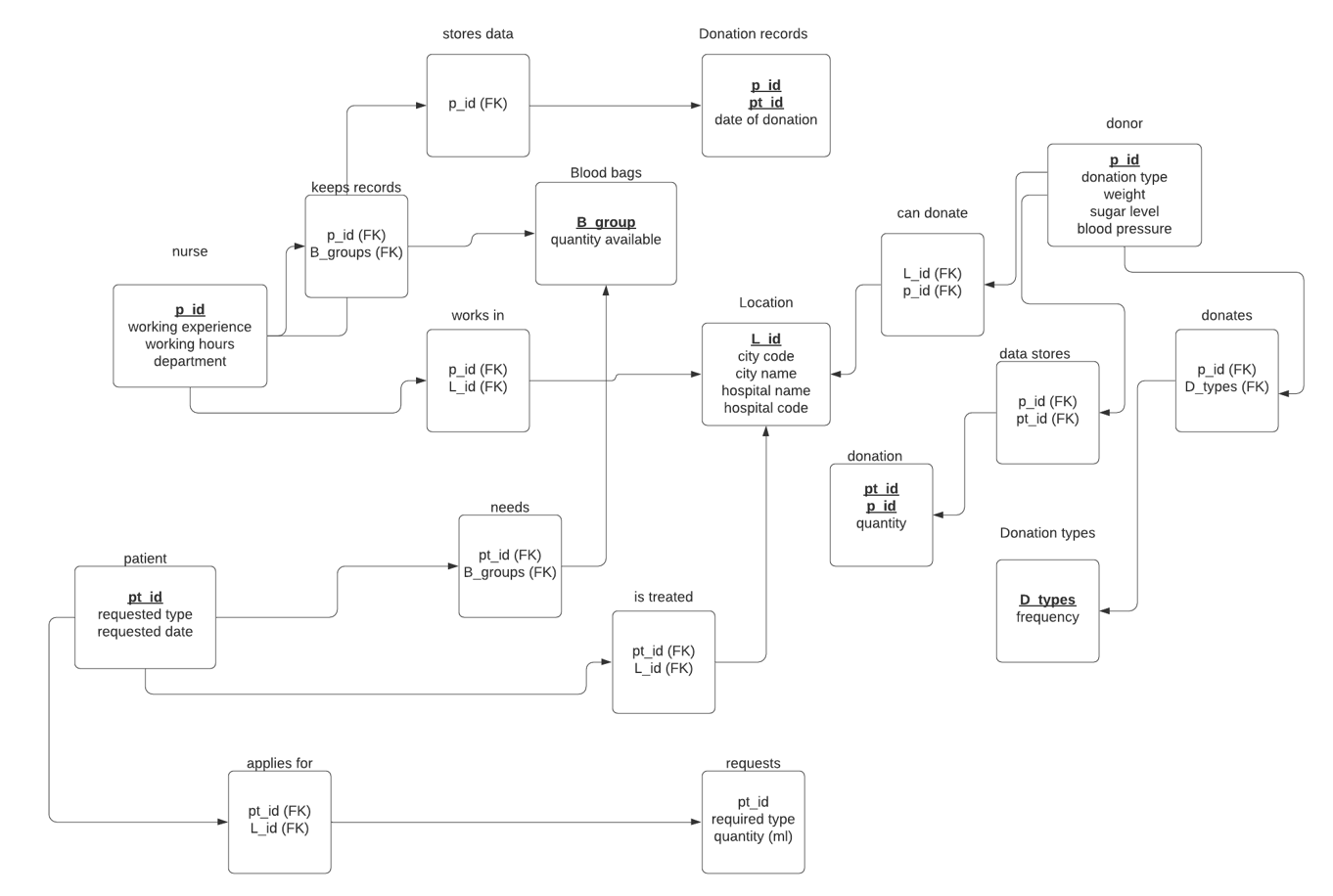
• Add primary keys of higher-level entities in the table of lower-level entities.

• In lower-level tables, add all other attributes of lower-level entities.

• Declare primary key of the higher-level table and the primary key for lower level table.

• Declare foreign key constraints.

After all these steps we get the relational schema.

****

**FUNCTIONAL DEPENDENCIES**

### The functional dependencies determines the relationship that exists between two attributes in Data Base Managment System. This generally exists between a prime key and a non prime key within a table.A functional dependency is denoted by an arrow "→". The functional dependency of A on B is represented by A→ B.

Rules for functional dependencies:

* Reflexive rule :

If A is a set of attributes and B is subset of A, then A holds a value of B.

* Augmentation rule:

When A→B holds, and C is attribute set, then AC → BC also holds. That is adding attributes which do not change the basic dependencies.

* Transitivity rule:

When A→ B holds and B → C holds, then A→ C also holds. A→ B is called as functionally that determines B.

The table representing functional dependencies is:

|  |  |
| --- | --- |
| Relations | Functional dependencies |
|  |  |
| person\_table | person\_table.id→ person\_table.name, person\_table.age, person\_table.gender, person\_table.phone number, person\_table.address, nurse\_table.pid, patient\_table.pid, donor\_table.pid |
| nurse\_table | nurse\_table.pid→ nurse\_table.experience, nurse\_table.working hours, nurse\_table.department, locationtable.Lid |
| patient\_table | patient\_table.ptid→ patient\_table.requested\_type, patient\_table.requested\_date,bloodbagstable.B\_groups,  locationtable.Lid |
| donor\_table | donor\_table.pid→ donor\_table.donation\_type, donor\_table.weight, donor\_table.upper\_bp, donor\_table.lower\_bp, donor\_table.sugar\_levels,  donation\_table.pid, donation\_type.Dtype |
| donation\_table | donation\_table.pid→ donation\_table.ptid, donation\_table.quantity |
| Donationrecords | Donationrecords.DonorID→ Donationrecords.patientID, Donationrecords.date\_of\_donation  Donationrecords.patientID→ Donationrecords.DonorID, Donationrecords.patientID.date\_of\_donation |
| donation\_type | donation\_type.Dtype→ donation\_type.frequency |
| request\_table | request\_table.pid→ request\_table.lid, request\_table.requested\_type, request\_table.date, request\_table.quantity |
| locationtable | locationtable.Lid→ locationtable.citycode, locationtable.cityname, locationtable.hospitalname, locationtable.hospitalcode |
| bloodbagstable | B\_groups→ quantityavailable |

NORMALISATION

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and to ensure that data dependencies make sense.

Normalization rules are divided into the following normal forms:

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

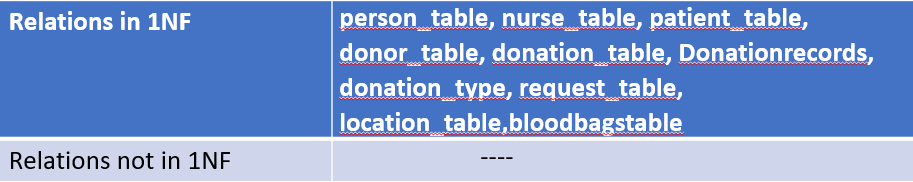
First Normal Form:

A table is in 1NF if:

•There are only Single Valued Attributes (Atomic).

•Values stored in a column should be of the same domain.

•All the columns in a table should have unique names.

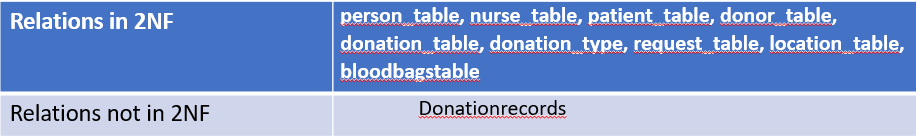


Second Normal Form:

A table is said to be in 2NF if both the following conditions hold:

•Table is in 1NF (First normal form)

•No non-prime attribute should be dependent on the proper subset of any candidate key of table (**No Partial Dependency**).



Reason: In Donationrecords table, donor id and patient id both are partially dependent. Both of them are required for knowing the date of donation; so this relation doesn’t satisfy NO PARTIAL DEPENDENCY and is not in 2NF.

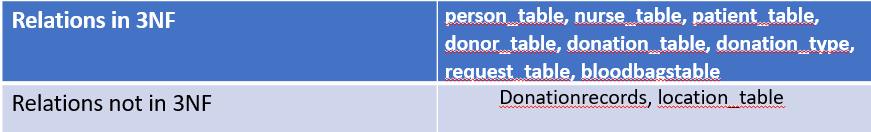
Third Normal Form:

A table is in 3NF if it is in 2NF and for each functional dependency X-> Y, at least one of the following conditions hold:

•X is a super key of table

•Y is a prime attribute of table.

Or in other words, it shouldn’t have **Transition Dependency.**

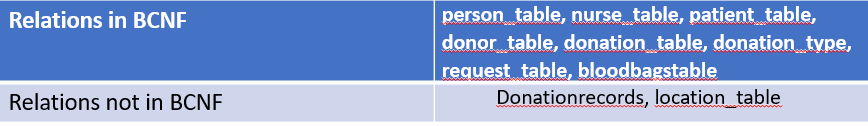
****

**Reason:** In location\_table, city code depends on city name and hospital code depends on hospital name clearly a non-prime is dependent on another non-prime which violates TRANSITION DEPENDENCY; so it is not in 3NF.

Boycc-codd Normal form:

Boyce and Codd Normal Form is a higher version of the Third Normal form. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied:

* R must be in third Normal Form
* and, for each functional dependency ( X → Y ), X should be a super Key.



RELATIONAL ALGEBRA

This is a procedural query language where the relation between the entities will be the output. Here the relations get searched recursively and the final relation will be displayed as the output.

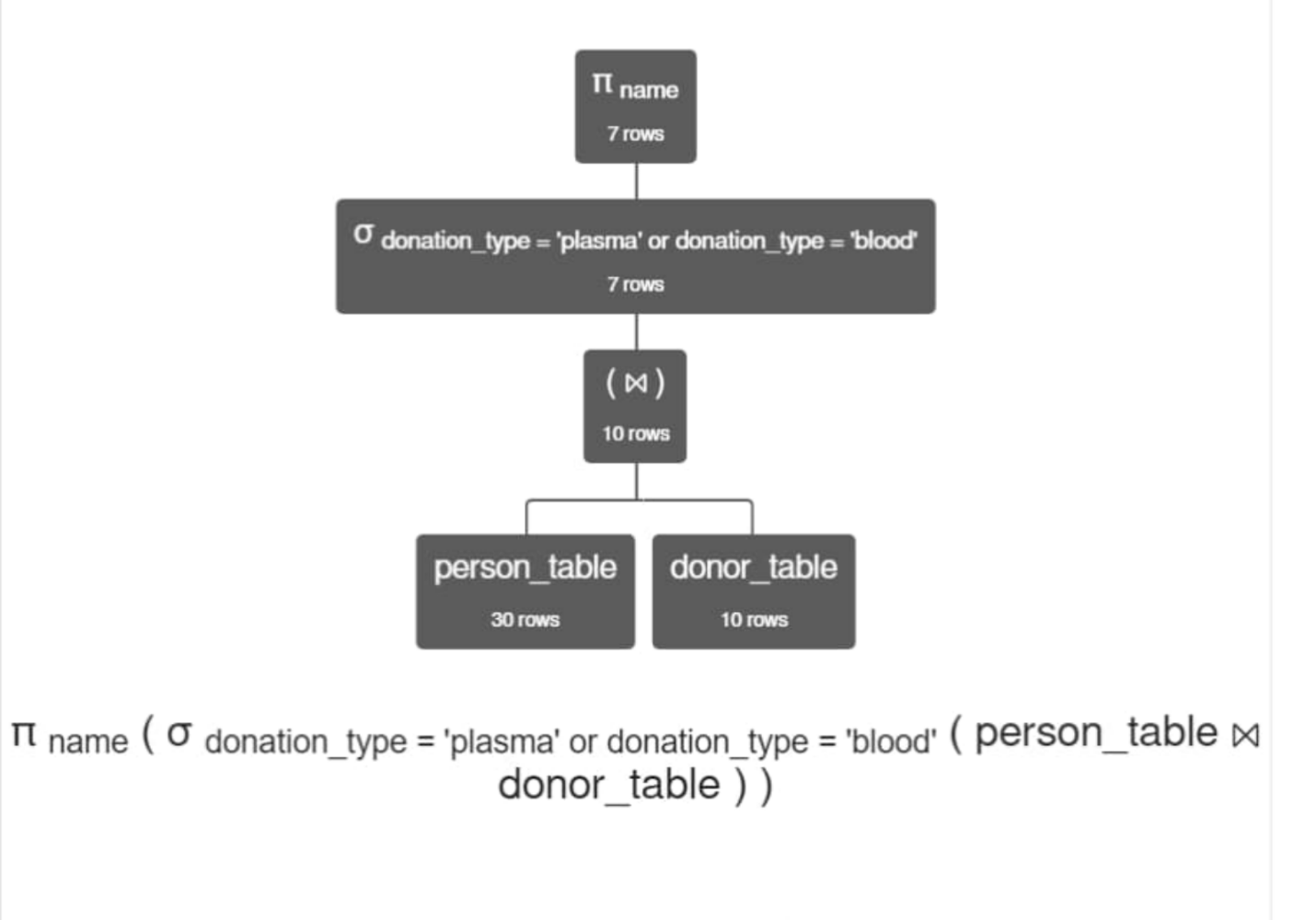
Relational Data Base Examples for proper data search.

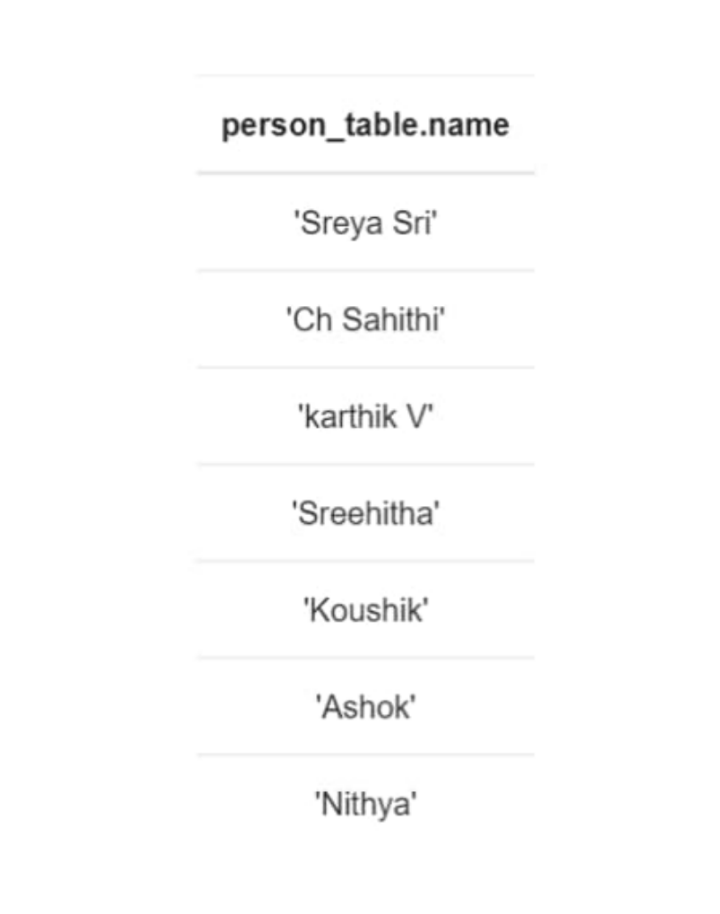
**Question 1 :**

Write the RA expression to find the names of donors who are willing to donate blood or plasma

**Expression :**

π name (σ donation\_type = 'plasma' ∨ donation\_type= 'blood' (person\_table⨝ donor\_table))

**Output 1 :**



**Explanation :**

Here, person(donor) name and the donation\_type are attributes in different relations person\_table and donor\_table respectively.

Hence we will natural join the two relations using the common attribute pid which is person ID.

we will now sort out the rows where the donation\_type is plasma or blood using or operator and selection operator.

Since it is now required to find the donor name, we use the projection operator to project the names of donors.

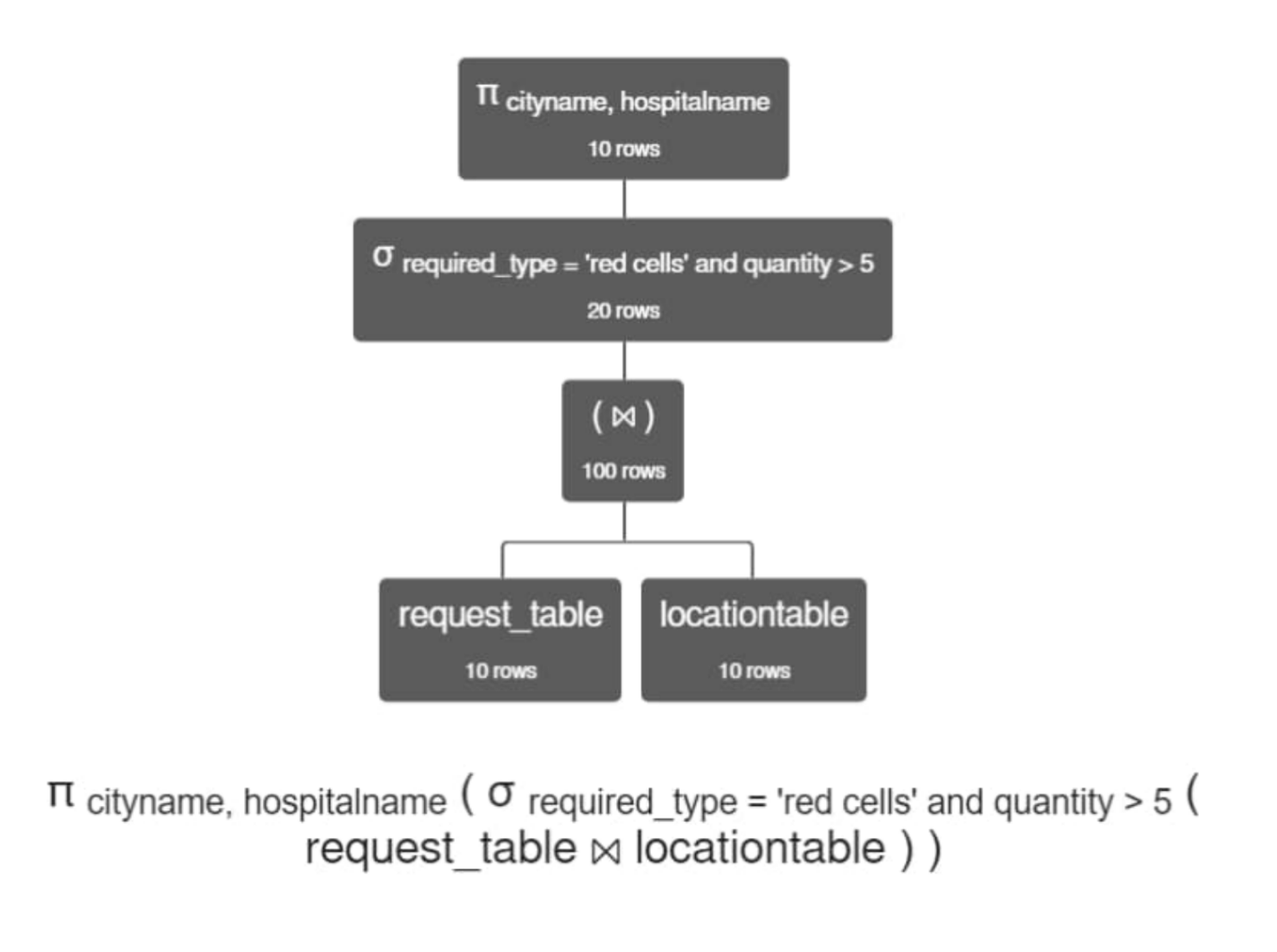
**Question 2 :**

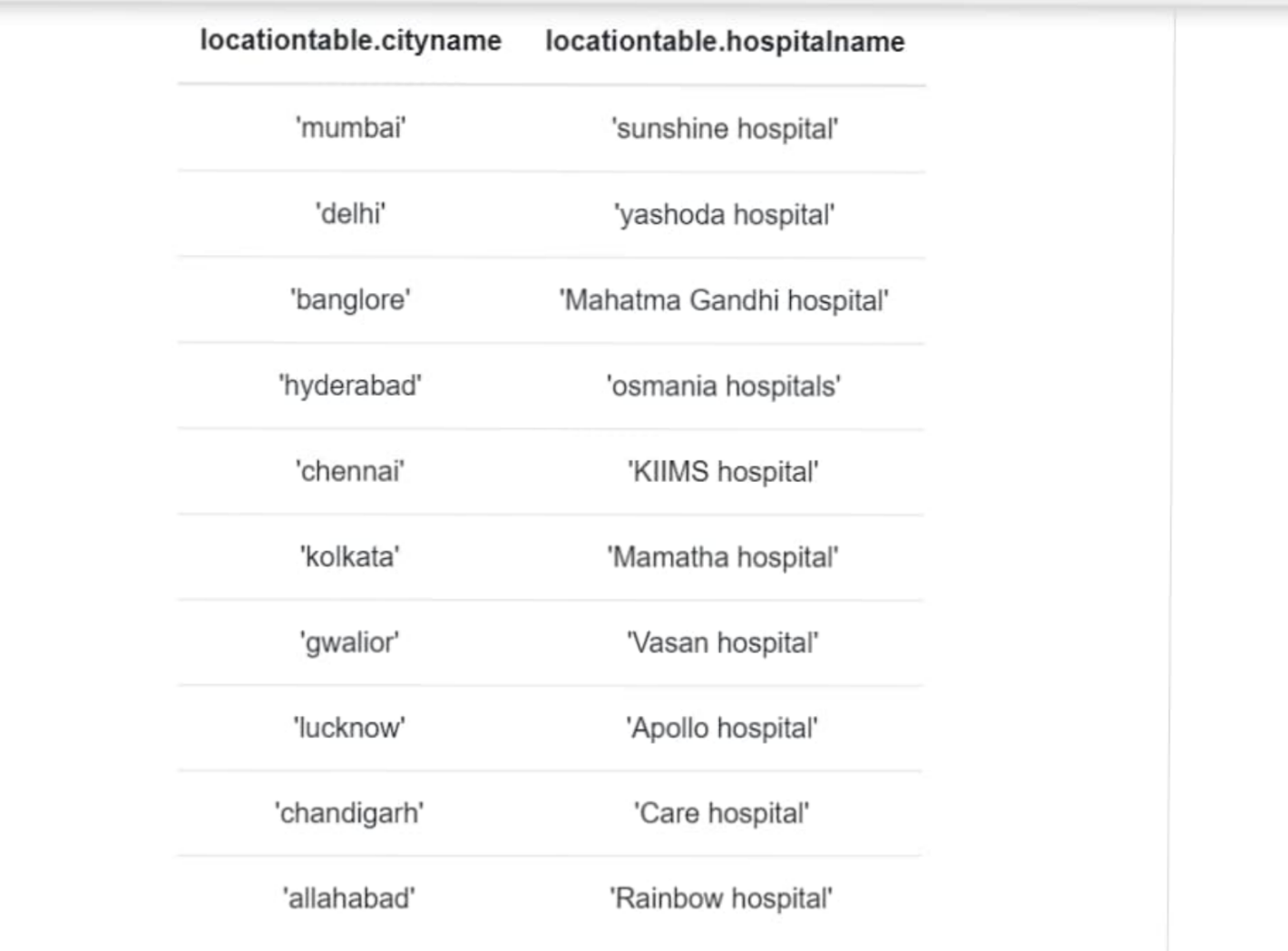
Write the RA expression to find the name of the hospitals and cities where the required red cells quantity is greater than 5.

**Expression :**

π cityname , hospitalname ( σ required\_type = 'red cells' and quantity > 5 ( request\_table ⨝ locationtable ))

**Output 2 :**





**Explanation:**

The hospitalname and cityname are attributes in locationtable relation whereas required\_type and quantity are attributes in reuested\_table. Since we have a common attribute lid in both the relations, we natural join them to get the desired table.

we use selection operator to sort out the tuples where requested type are red cells and the quantity of red cells is greater than 5.

To project the hospital name and city name , projection operator is used.

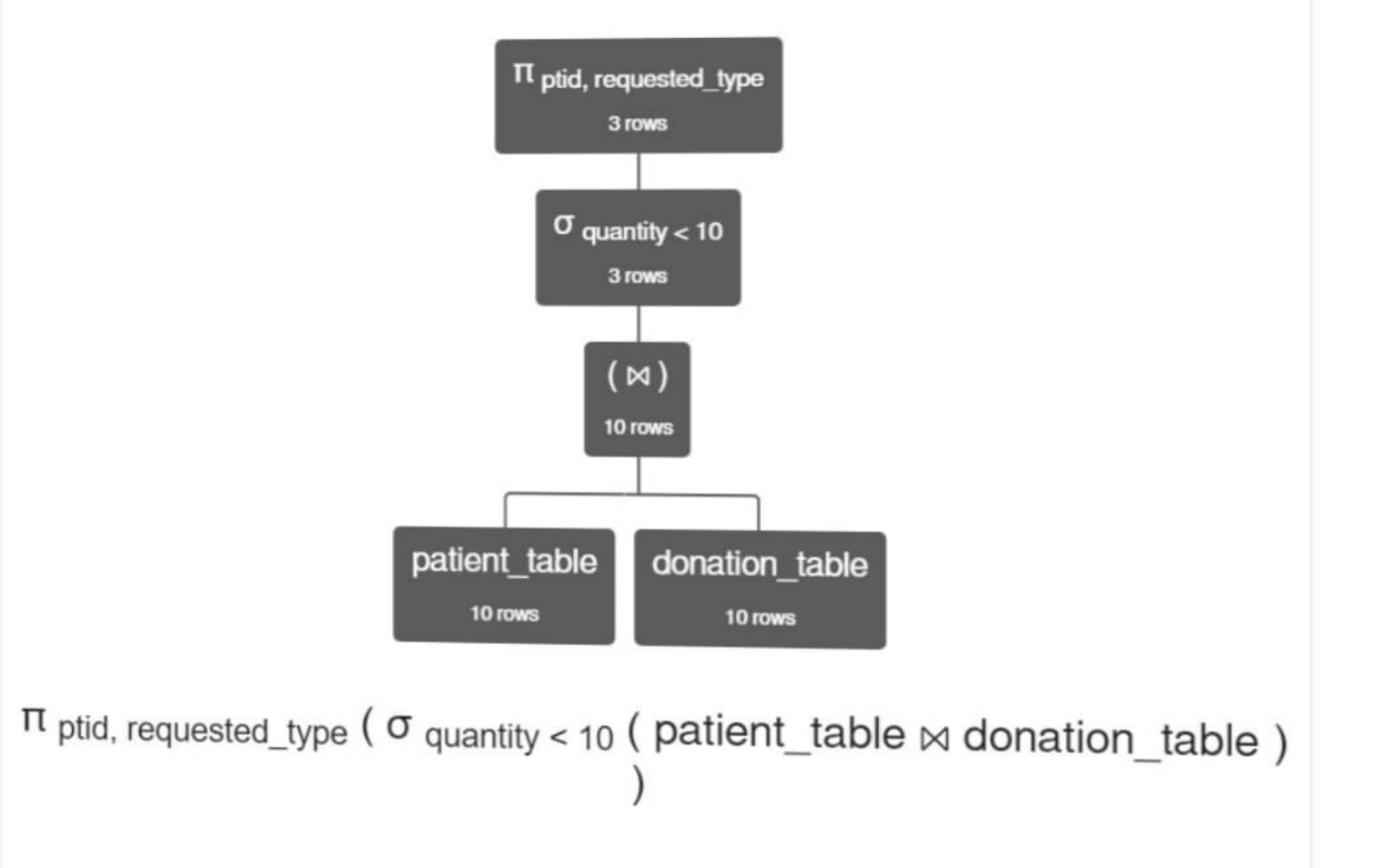
**Question 3 :**

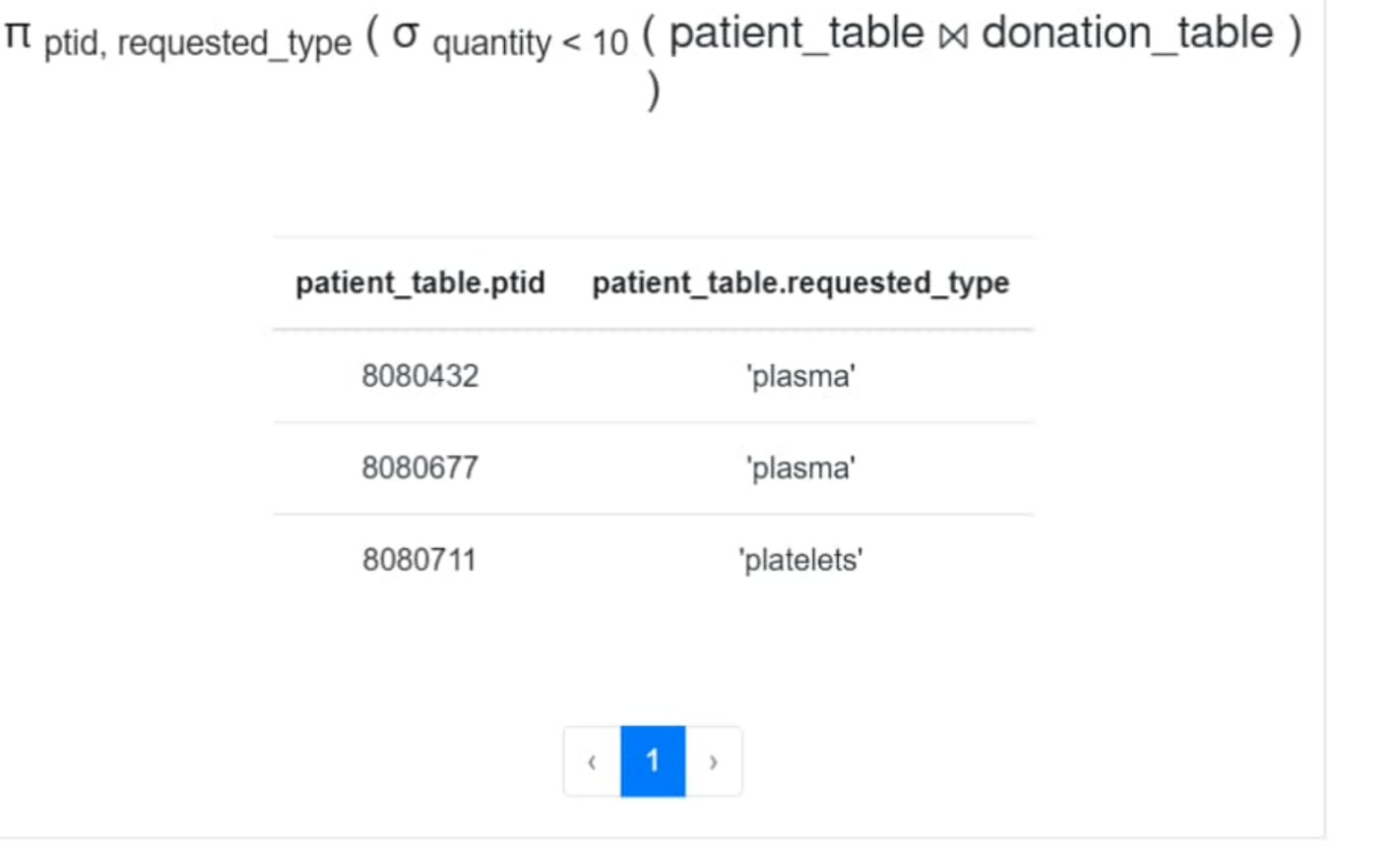
Write the RA expression to find the requested\_type and id of the patients whose required quantity is less than 10.

**Expression :**

π ptid, requested\_type ( σ quantity < 10 ( patient\_table ⨝ donation\_table))

**Output 3 :**





**Explanation:**

Quantity is an attribute in donation\_table whereas the requested\_type is the attribute in patient\_table. The common attribute in both the relations id ptid which is the patient Id.

we natural join the two relations donation\_table and patient\_table using the common attribute ptid.

Now, we sort out the tuples where the required quantity is less than 10 using the selection operator.

Since the final output we wanted is the ptid and requested\_type, we use projection operator over these attributes.

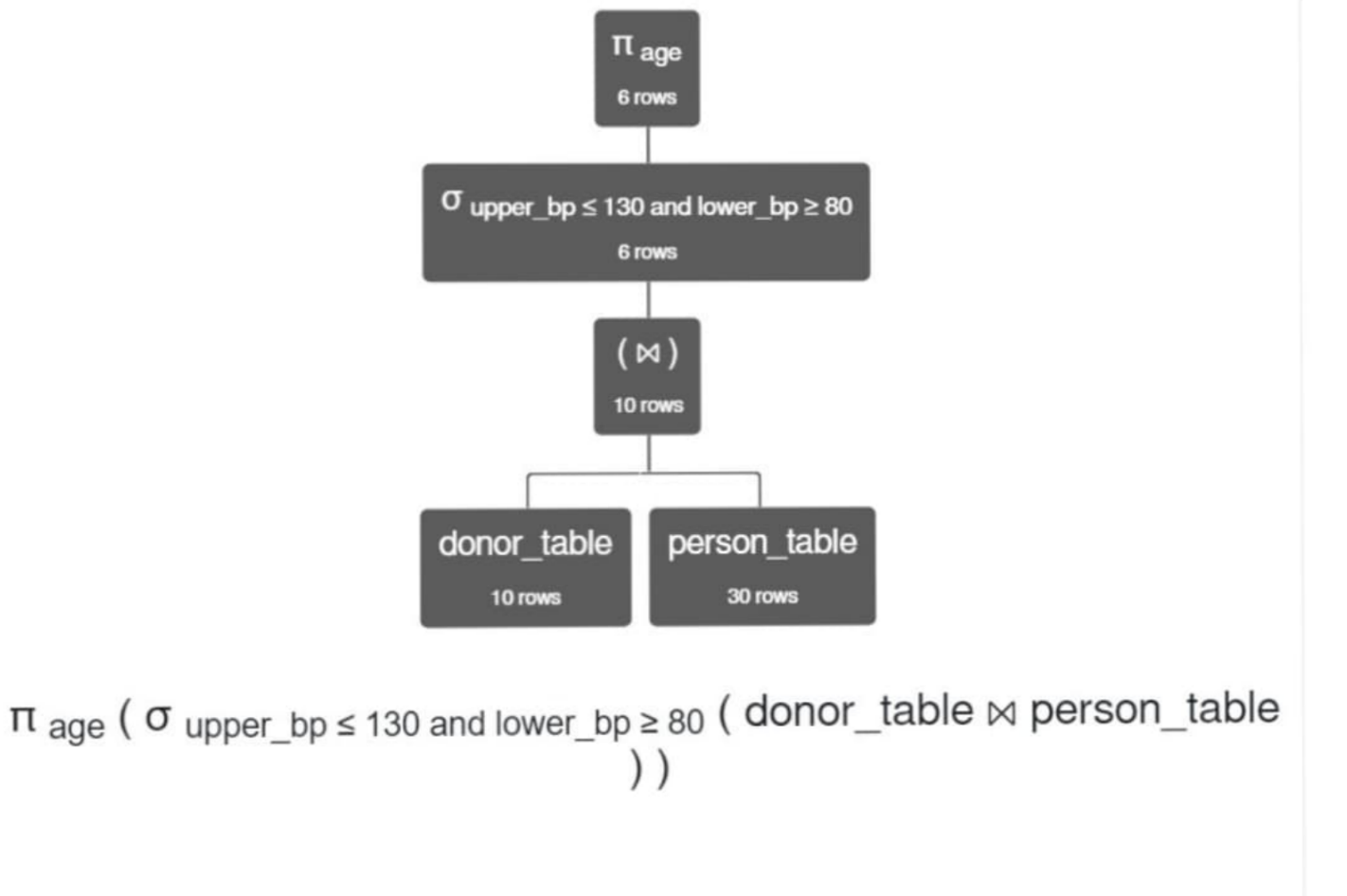
**Question 4 :**

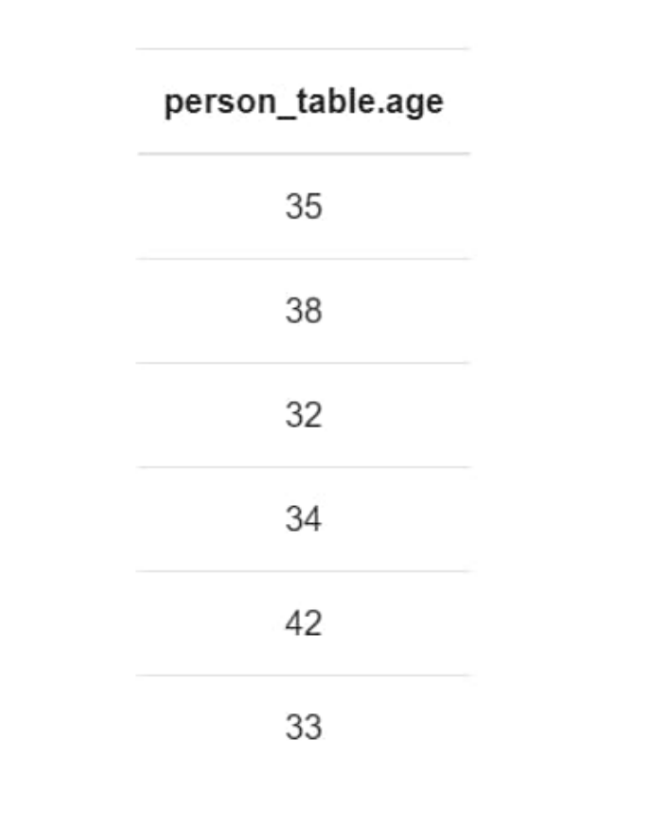
Write the RA expression to find the age of donors whose upper bp is less than or equal to 130 and lower bp is greater than or equal to 80.

**Expression :**

π age ( σ upper\_bp ≤ 130 and lower\_bp ≥ 80 ( donor\_table ⨝ person\_table ))

**Output 4:**





**Explanation:**

Age is attribute in person\_table whereas the upper\_bp and lower\_bp are attributes in donor\_table. To get the desired relaton where all of age, upper\_bp and lower\_bp are attributes, we natural join person\_table and donor\_table. Using greater than and less than or equal to symbols , we find the essential tuples where upper\_bp is less than or equal to 130 and lower\_bp is geater than or equal to 80 . To sort out them , we use selection operator. Finally to get the required output age , we use projection operator over age.

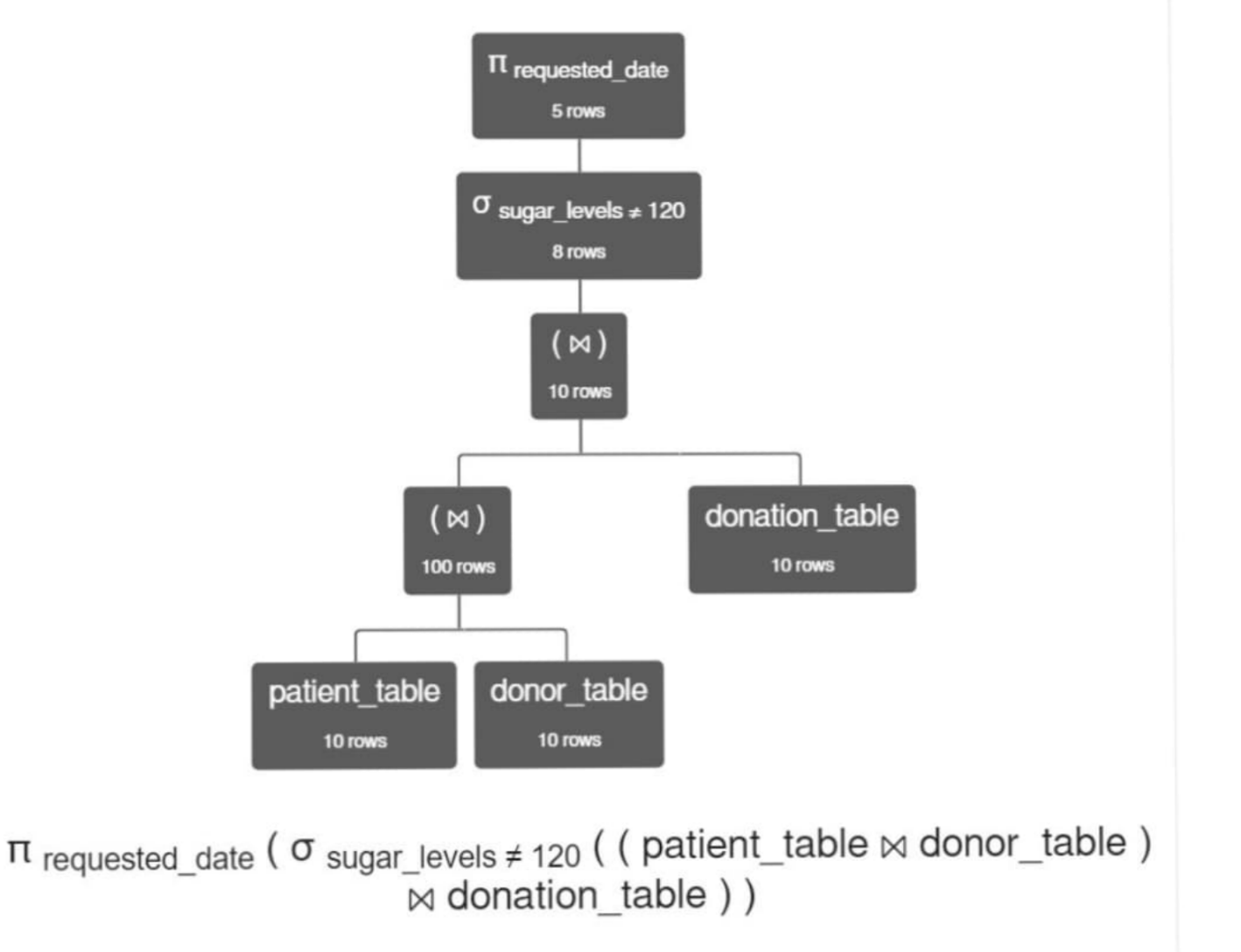
**Question 5:**

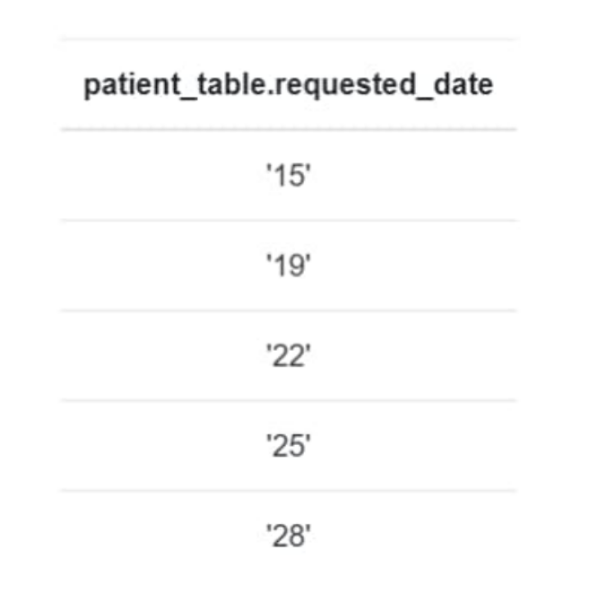
Write the RA expression to find the requested date of the patients who are donated by persons whose sugar levels are not equal to 120.

**Expression :**

π requested\_date ( σ sugar\_levels ≠ 120 (( patient\_table ⨝ donation\_table ) ⨝ donor\_table ))

**Output 5 :**





Explanation:

Requested\_date is the attribute from patient\_table and sugar\_levels is the attribute in donor\_table. We have no common attribute between these relations so we added new relation donation\_table which has common attributes with both of them. pid is the common attribute between donation\_table and and donor\_table. ptid is the common attribute between donation\_table and patient\_table. Hence we natural join these 3 relations to get all the attributes. Now, we sort out the tupples where sugar\_levels are not equal to 130 using selection operator. To get the desired output which is required\_date, we use projection operator.

It is to be noted that the final output has no redundancy.

**SQL**

SQL stands for structured query language,it lets us manipulate and access the data and used to perform operations on the records stored in the database such as updating records, deleting records, creating and modifying tables, views, etc.

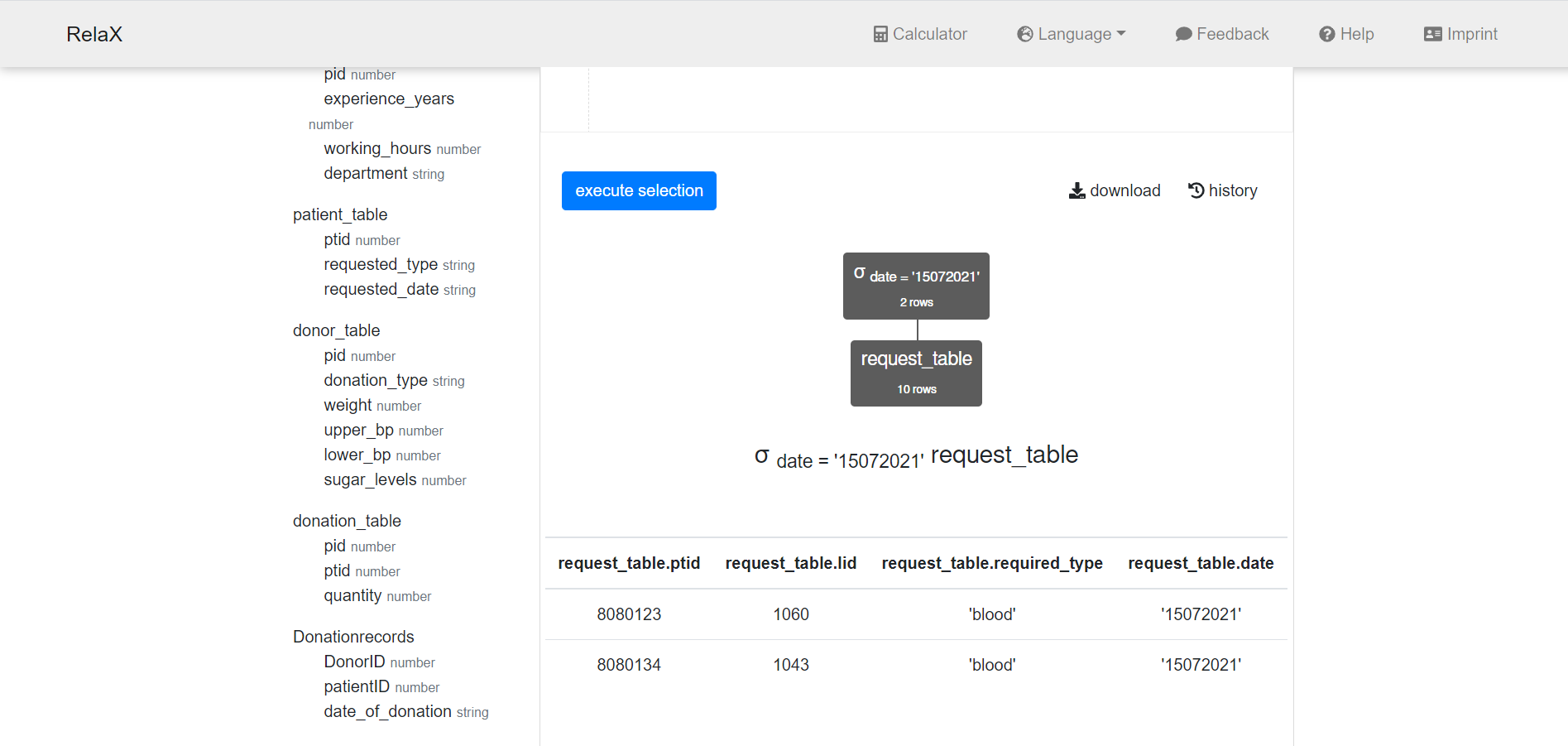
**QUESTION1:**

Write an SQL query to get the list of requests on the date 15-07-2021?

**ANSWER:**

'SELECT\* FROM request\_table

where date='SELECT 15072021';



**Reason:**

To get the list of requests on date 15-07-2021 we need to use the condition statement **where** and display it using **select** command from the request table.

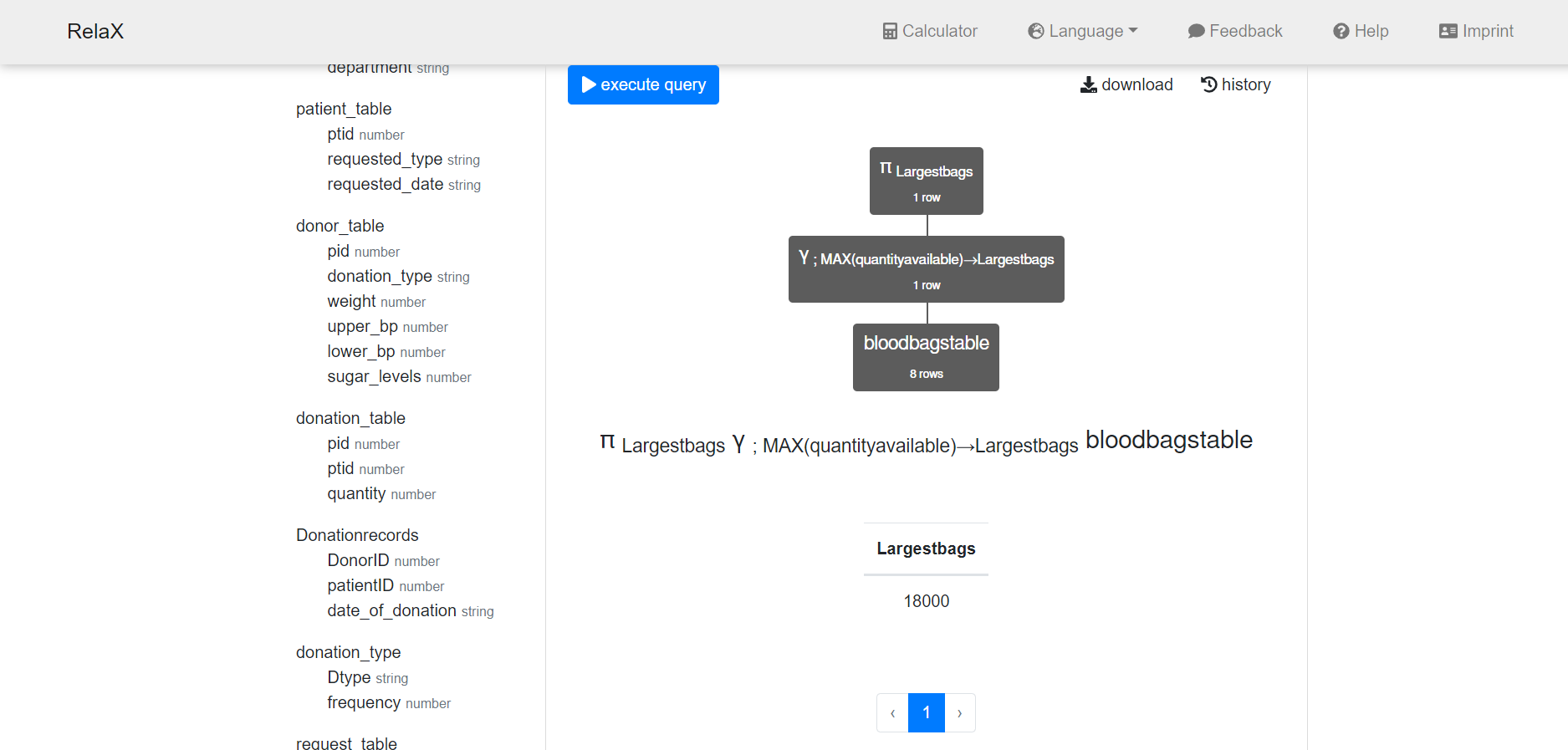
**QUESTION2:**

Write an SQL query to find which blood group has the highest blood bags?

**ANSWER:**

SELECT MAX(quantityavailable) AS Largestbags

FROM bloodbagstable;



**Reason:**

To get the maximum number of blood bags we use **max** function and aggregate it with the largest bags.

**QUESTION3:**

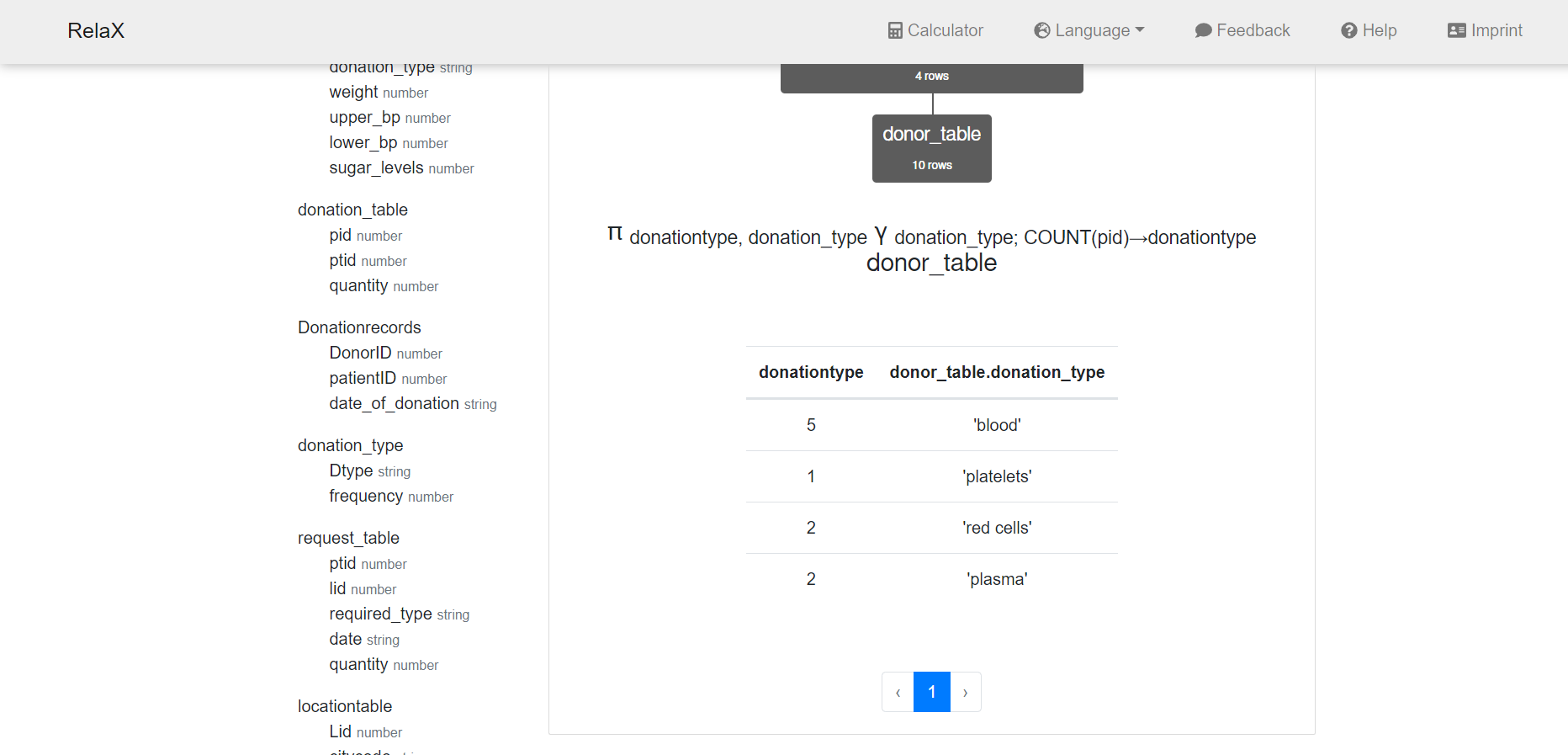
Write an SQL query to count the type of donations done by donors?

**ANSWER:**

SELECT COUNT(pid) AS donationtype, donation\_type

FROM donor\_table

GROUP BY donation\_type;



**Reason:** To get the number of types of donations done by donors we use function **count**.

**QUESTION4:**

Write the sql query to get the required type donation in Hyderabad hsptl?

**ANSWER:**

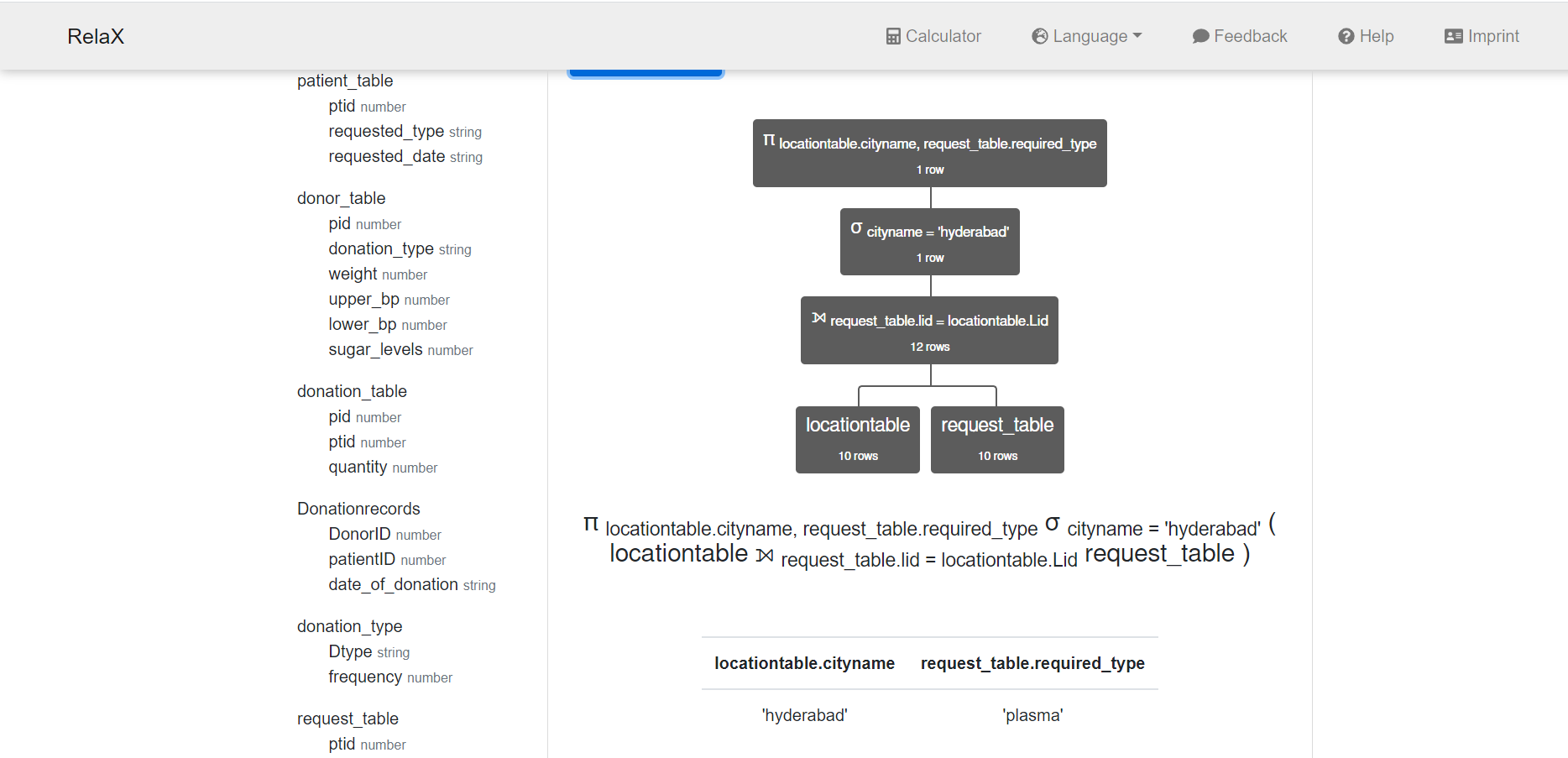
SELECT locationtable.cityname,request\_table.required\_type

FROM locationtable

LEFT JOIN request\_table

ON request\_table.lid=locationtable.Lid

WHERE cityname='hyderabad';



**Reason:**

To get the required type of donation in hyderabad hospital we join the tables requestable and location table by **left join** by using primary key lid and apply the condition of city name **Is equal** to Hyderabad

**QUESTION5:**

Write an sql query to get the names of the nurses who have experience for more than 5years?

**ANSWER:**

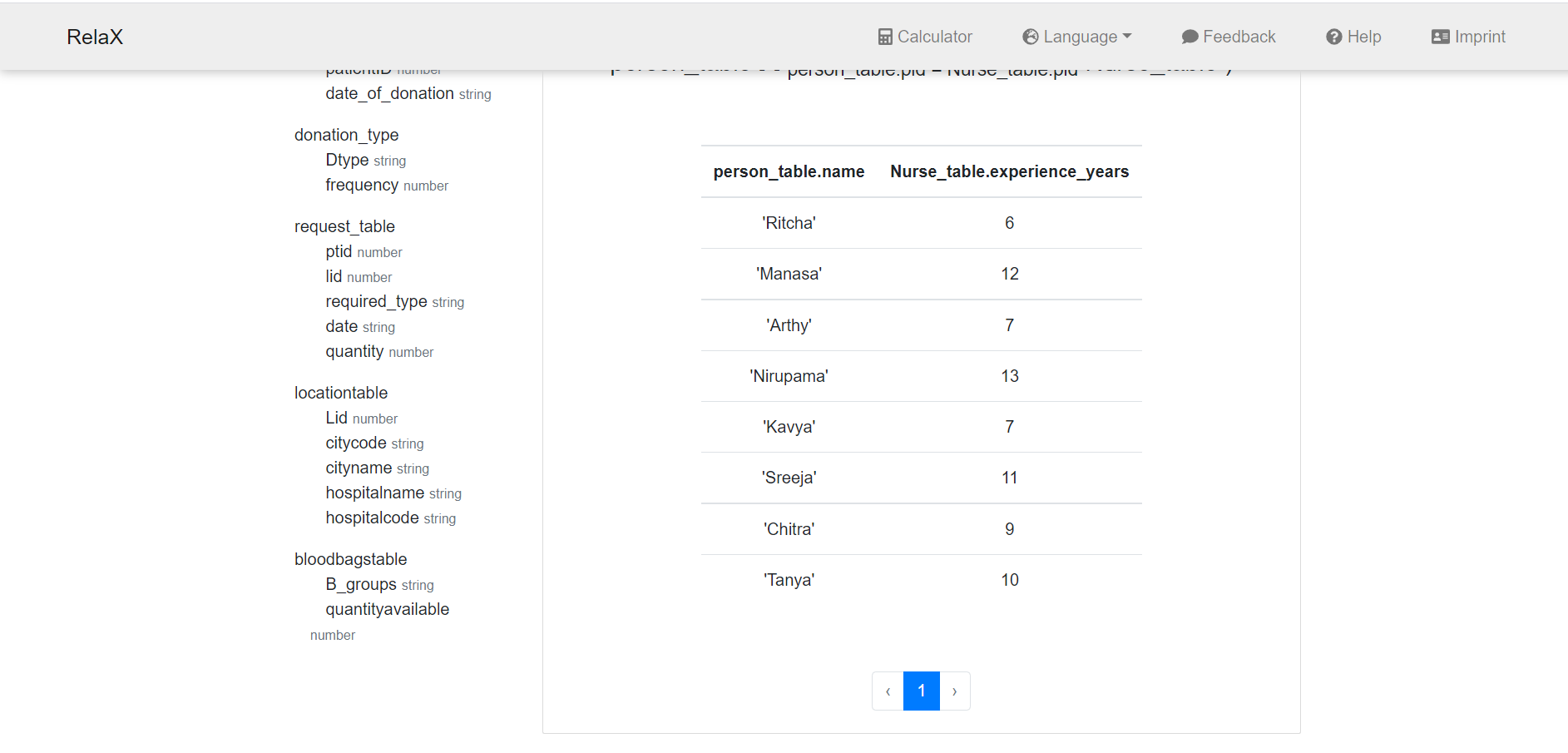
SELECT person\_table.name,Nurse\_table.experience\_years

FROM person\_table

LEFT JOIN Nurse\_table

ON person\_table.pid=Nurse\_table.pid

WHERE experience\_years>5;



**Reason:** To get the names of the nurses who has experienced more than 5 years

We first have to join nurse\_table and person\_table as the primary key is pid by using **left join** and apply the condition of **experience years>5**

**THANK YOU**