CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

PROJECT REPORT
ON
BLOOD BANK MANAGEMENT SYSTEM

Program Name: BCA

Subject Name/Code: Database Management System(23CAT-251)

Submitted by:

Name: Harsh Girdhar

UID: **23BCA10485**

Section: BCA – **4** "A"

Submitted to:

Name: Arvinder Singh

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

ABSTRACT

Introduction:

This SQL script creates a Blood Bank Management System that handles data for donors, hospitals, blood donations, blood inventory, and recipients. It includes table creation with primary and foreign keys to ensure data integrity, followed by sample data insertion. The script also features a variety of SQL queries to retrieve and analyze information, such as total donations by donors, blood inventory levels, recipient details, and average donation quantities. This system provides a structured way to manage and track blood donation and distribution efficiently.

Technique:

🛠 Techniques Used

1. Database Creation

CREATE DATABASE and USE statements to initialize and select the database.

2. Table Creation

CREATE TABLE to define structured tables.

Use of PRIMARY KEY and AUTO_INCREMENT for unique record identification.

FOREIGN KEY constraints to establish relationships between tables (e.g., donor_id, hospital_id).

3. Data Insertion

INSERT INTO to populate tables with sample data for donors, hospitals, donations, inventory, and recipients.

4. Data Retrieval (SELECT Queries)

Basic SELECT * for full table views.

SELECT with specific columns and aliases using AS.

5. Joins

Use of JOIN and table aliases to combine data from multiple tables (e.g., joining Donor with Blood_Donation, or Recipient with Hospital).

6. Aggregation Functions

SUM(), AVG(), COUNT() to calculate totals, averages, and record counts.

GROUP BY to organize aggregated data by fields like donor name or blood group.

7. Filtering

WHERE clause to filter rows based on conditions (e.g., blood group, quantity > 300).

8. Sorting

ORDER BY used to sort results ascending (ASC) or descending (DESC) by quantity.

System Configuration:

💻 1. Software Requirements

Component	Recommendation
DBMS	MySQL (v5.7 or later) / MariaDB
OS	Windows, Linux, or macOS
Interface	MySQL Workbench, phpMyAdmin, or command-line
Optional	XAMPP / WAMP for local server setup

🛠️ 2. Hardware Requirements (Minimum)

Resource	Specification
Processor	Dual-core 1.8 GHz or higher
RAM	4 GB (8 GB recommended for better performance)
Storage	500 MB for MySQL and data files
Disk Type	SSD recommended for faster query execution

🌐 3. Additional Tools (Optional but Helpful)

Text Editor/IDE: VS Code, Sublime Text, or any SQL-friendly editor

ER Diagram Tool: dbdiagram.io, MySQL Workbench for visualizing schema

Backup Tool: MySQL dump for database export/import

🔐 4. Configuration Tips

Set proper user privileges in MySQL (avoid using root for application access).

Enable foreign key checks to maintain relational integrity.

Configure regular backups and logging in a production setup.

SUMMARY

Input:

The input of this SQL code consists of structured data definitions and data entries used to simulate a real-world blood bank scenario. Here's what the input includes:

🗁 1. Database and Table Creation
Creation of a new database: bloodbank

Definition of five main tables:

Donor

Hospital

Blood_Donation

Blood_Inventory

Recipient

Each table includes relevant columns, primary keys, and foreign keys where applicable.

🧪 2. Sample Data Insertion

Donor Table: 3 donors with different ages, blood groups, and contacts.

Hospital Table: 2 hospitals with names, locations, and contact info.

Blood_Donation Table: 3 donation records linked to donors.

Blood_Inventory Table: Blood group-wise inventory levels.

Recipient Table: 2 recipients associated with different hospitals.

📊 3. Query Input

Includes various SQL queries as input to:

Retrieve all donors and their total donation quantities.

Check available blood inventory.

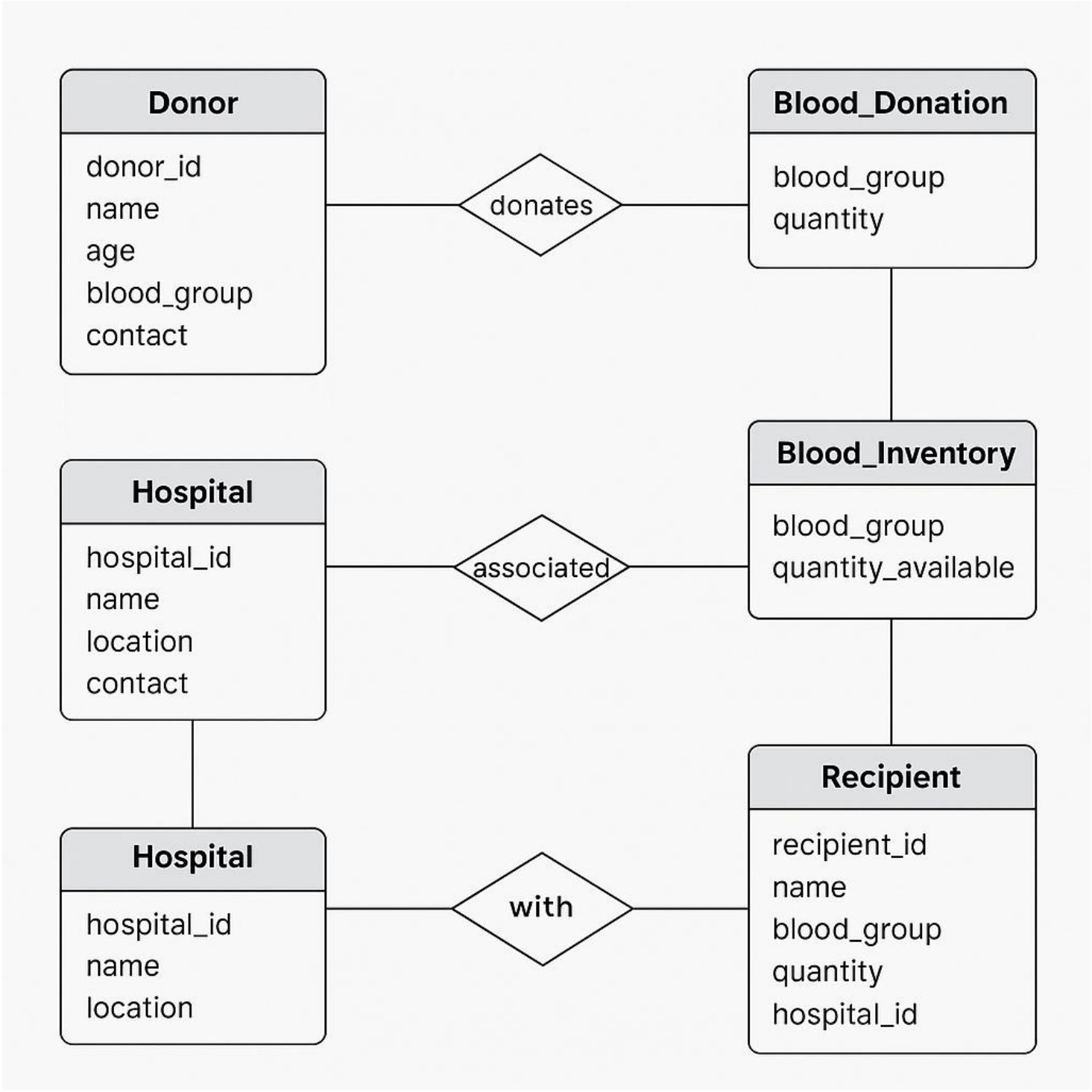
List recipient and hospital information.

Filter and sort based on quantity.

Perform aggregations like SUM(), AVG(), and COUNT().

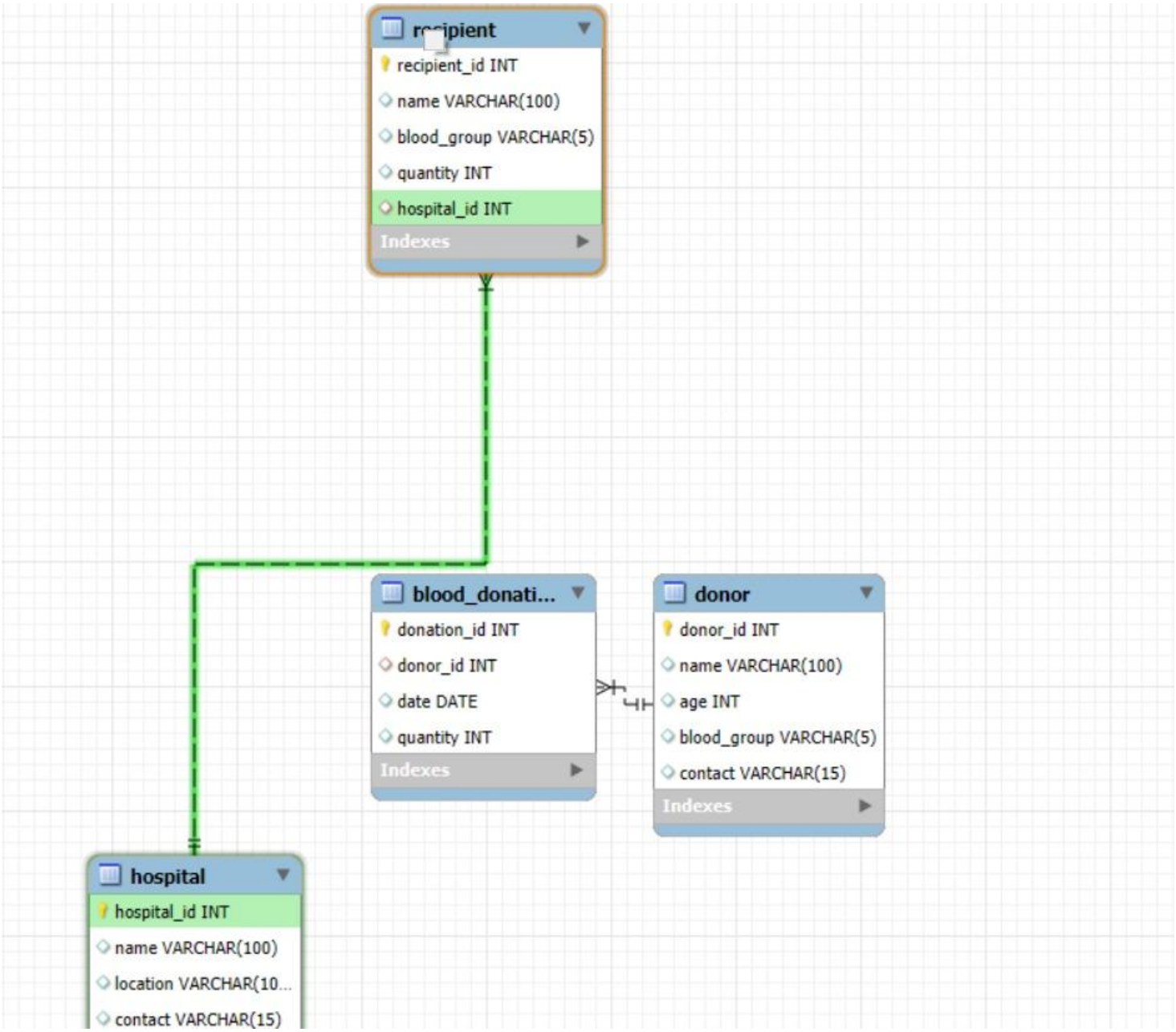
CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

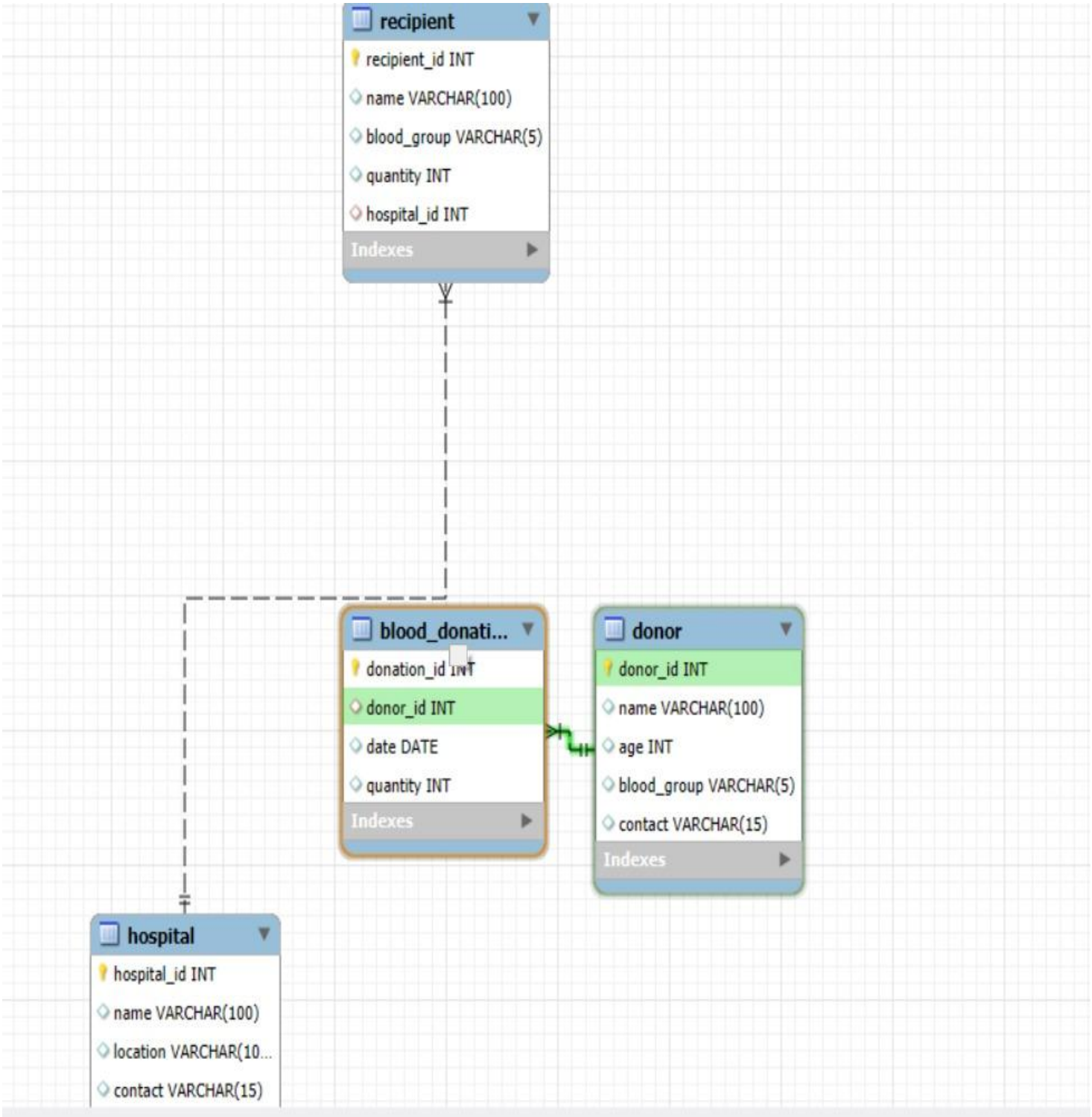
ER DIAGRAM:

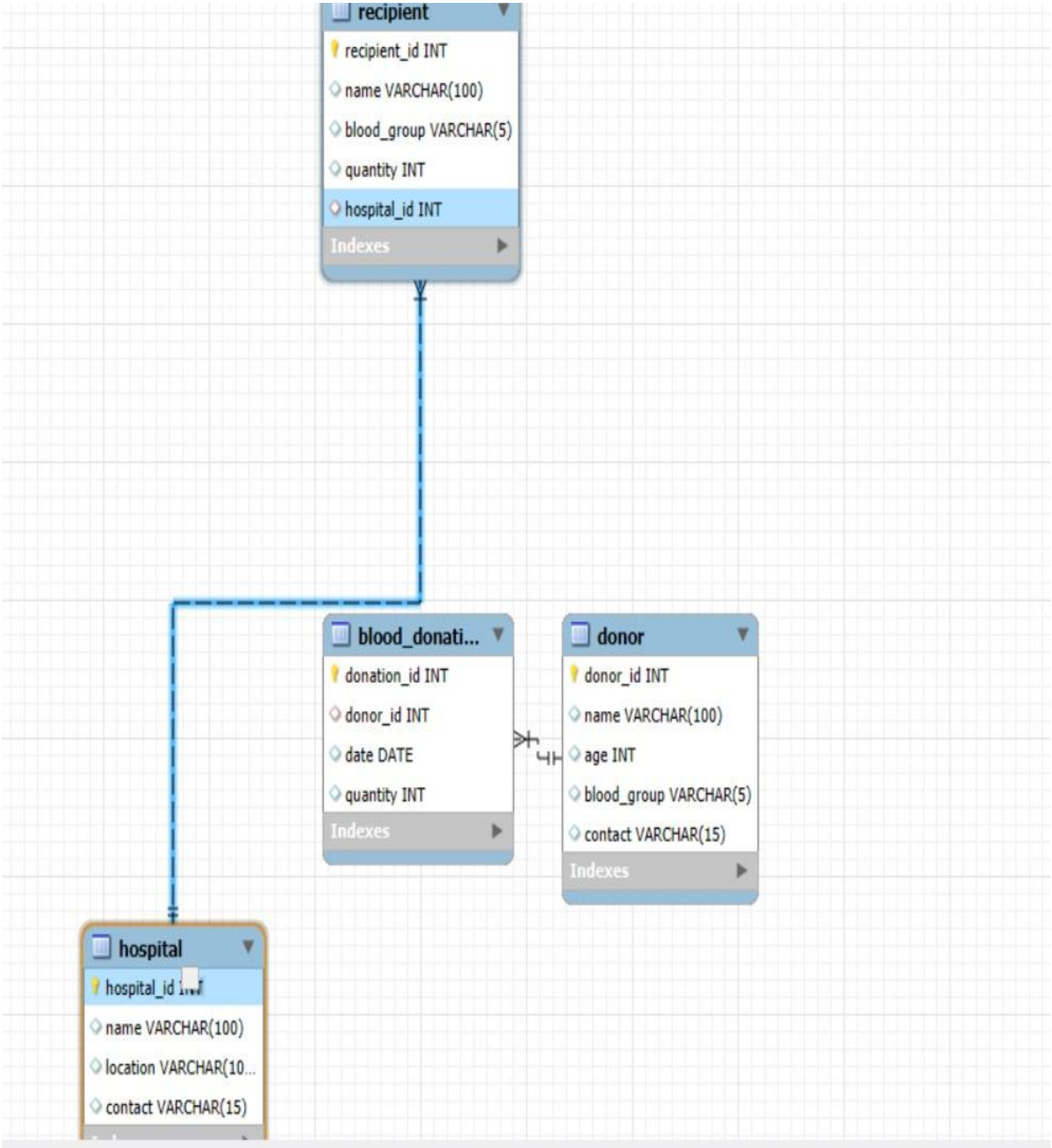


CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

TABLE REALTION:




recipient
recipient_id INT
name VARCHAR(100)
blood_group VARCHAR(5)
quantity INT
hospital_id INT
Indexes
blood_donati...
donation_id INT
donor_id INT
date DATE
quantity INT
Indexes
donor
donor_id INT
name VARCHAR(100)
age INT
blood_group VARCHAR(5)
contact VARCHAR(15)
Indexes
hospital
hospital_id INT
name VARCHAR(100)
location VARCHAR(10...
contact VARCHAR(15)


recipient
recipient_id INT
name VARCHAR(100)
blood_group VARCHAR(5)
quantity INT
hospital_id INT
Indexes
blood_donati...
donation_id INT
donor_id INT
date DATE
quantity INT
Indexes
donor
donor_id INT
name VARCHAR(100)
age INT
blood_group VARCHAR(5)
contact VARCHAR(15)
Indexes
hospital
name VARCHAR(100)
location VARCHAR(10...
contact VARCHAR(15)

TABULAR FORMAT:

1. Donor

Column	Data Type	Description
donor_id	INT	Primary Key, Auto-incremented unique ID
name	VARCHAR(100)	Name of the donor
age	INT	Age of the donor
blood_group	VARCHAR(5)	Blood group of the donor
contact	VARCHAR(15)	Contact number of the donor

2. Hospital

Column	Data Type	Description
hospital_id	INT	Primary Key, Auto-incremented unique ID
name	VARCHAR(100)	Name of the hospital
location	VARCHAR(100)	Location of the hospital
contact	VARCHAR(15)	Contact number of the hospital

3. Blood Donation

Column	Data Type	Description
donation_id	INT	Primary Key, Auto-incremented unique ID
donor_id	INT	Foreign Key referencing the `donor_id` from the `Donor` table
date	DATE	Date of the blood donation
quantity	INT	Quantity of blood donated (in milliliters)

4. Blood Inventory

Column	Data Type	Description
blood_group	VARCHAR(5)	Primary Key, Blood group (e.g., 'A+', 'B+', 'O-')
quantity_available	INT	Quantity of blood available for that blood group

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

5.Reciepent

Column	Data Type	Description
recipient_id	INT	Primary Key, Auto-incremented unique ID
name	VARCHAR(100)	Name of the recipient
blood_group	VARCHAR(5)	Blood group of the recipient (e.g., 'A+', 'O-', etc.)
quantity	INT	Quantity of blood required (in milliliters)
hospital_id	INT	Foreign Key referencing the `hospital_id` from the `Hospital` tabl

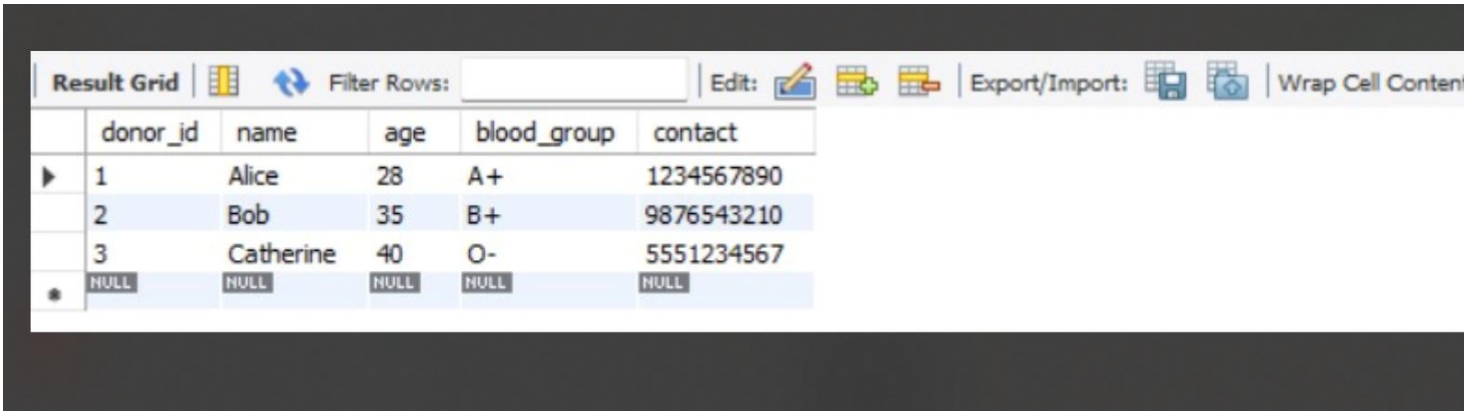
Table created:

#	Time	Action	Message
3	15:04:45	CREATE TABLE Donor (donor_id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(100), ag...	0 row(s) affecte
4	15:05:32	create table hospital(hospital_id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(100), locat...	0 row(s) affecte
5	15:06:11	CREATE TABLE Blood_Donation (donation_id INT PRIMARY KEY AUTO_INCREMENT, donor_id INT, ...	0 row(s) affecte
6	15:06:20	CREATE TABLE Blood_Inventory (blood_group VARCHAR(5) PRIMARY KEY, quantity_available INT)	0 row(s) affecte
7	15:06:29	CREATE TABLE Recipient (recipient_id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(100)...	0 row(s) affecte

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

IMPLEMENTATION:
SQL Queries Performed with Output:

1. SELECT * FROM Donor;



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content

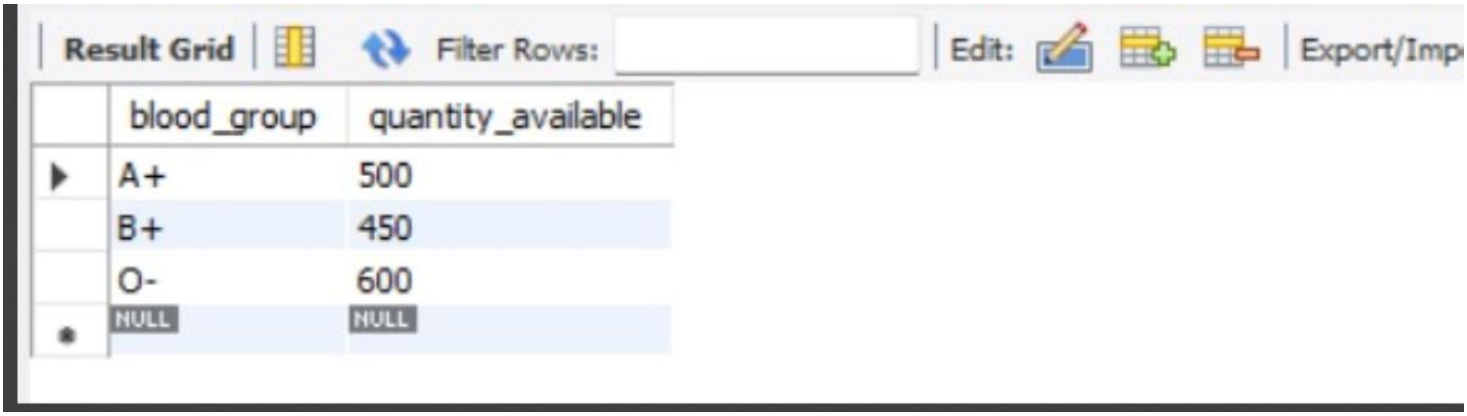
donor_id	name	age	blood_group	contact
1	Alice	28	A+	1234567890
2	Bob	35	B+	9876543210
3	Catherine	40	O-	5551234567
NULL	NULL	NULL	NULL	NULL

2. SELECT Donor.name, SUM(Blood_Donation.quantity) AS total_donated

FROM Donor

JOIN Blood_Donation ON Donor.donor_id = Blood_Donation.donor_id

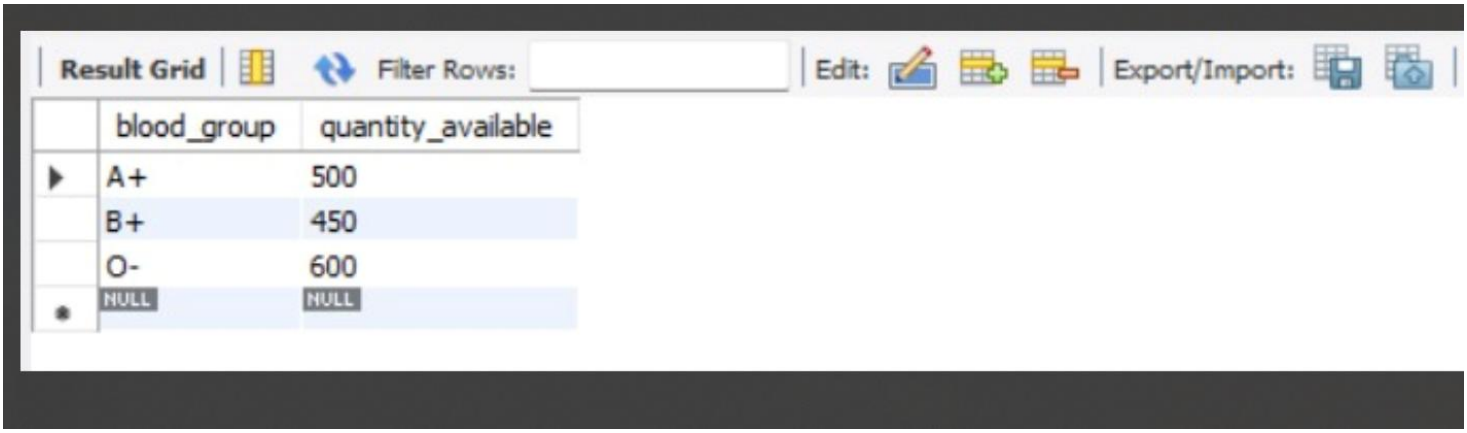
GROUP BY Donor.name;



Result Grid | Filter Rows: | Edit: | Export/Imp

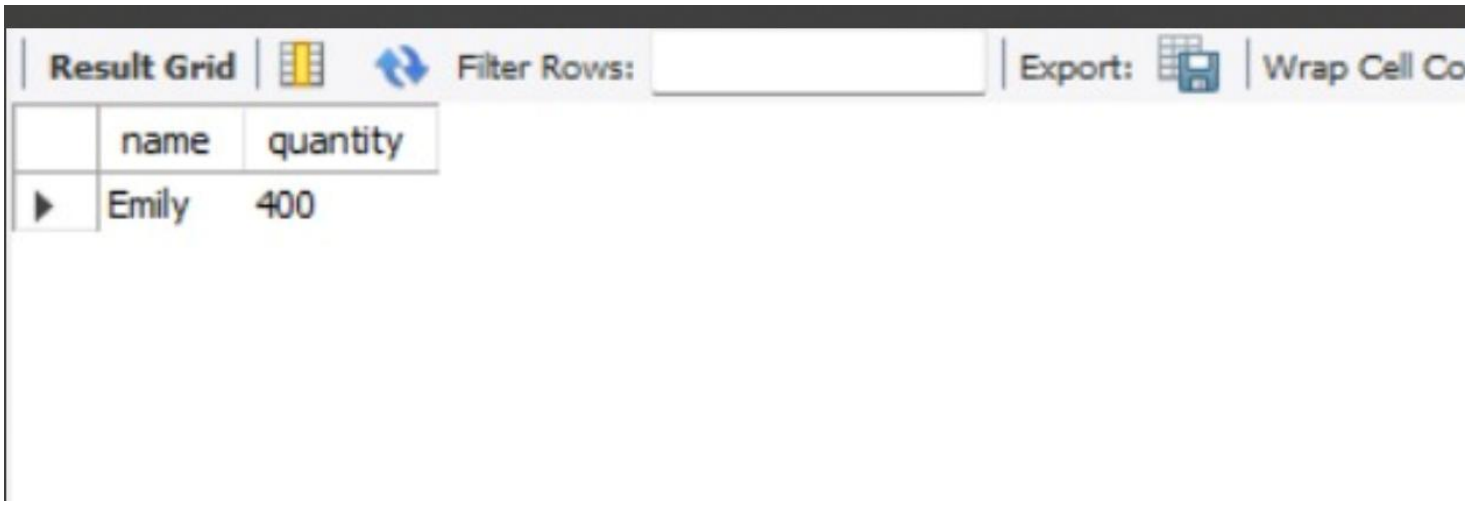
blood_group	quantity_available
A+	500
B+	450
O-	600
NULL	NULL

3. SELECT * FROM Blood_Inventory;


Result Grid | Filter Rows: | Edit: | Export/Import:

blood_group	quantity_available
A+	500
B+	450
O-	600
NULL	NULL

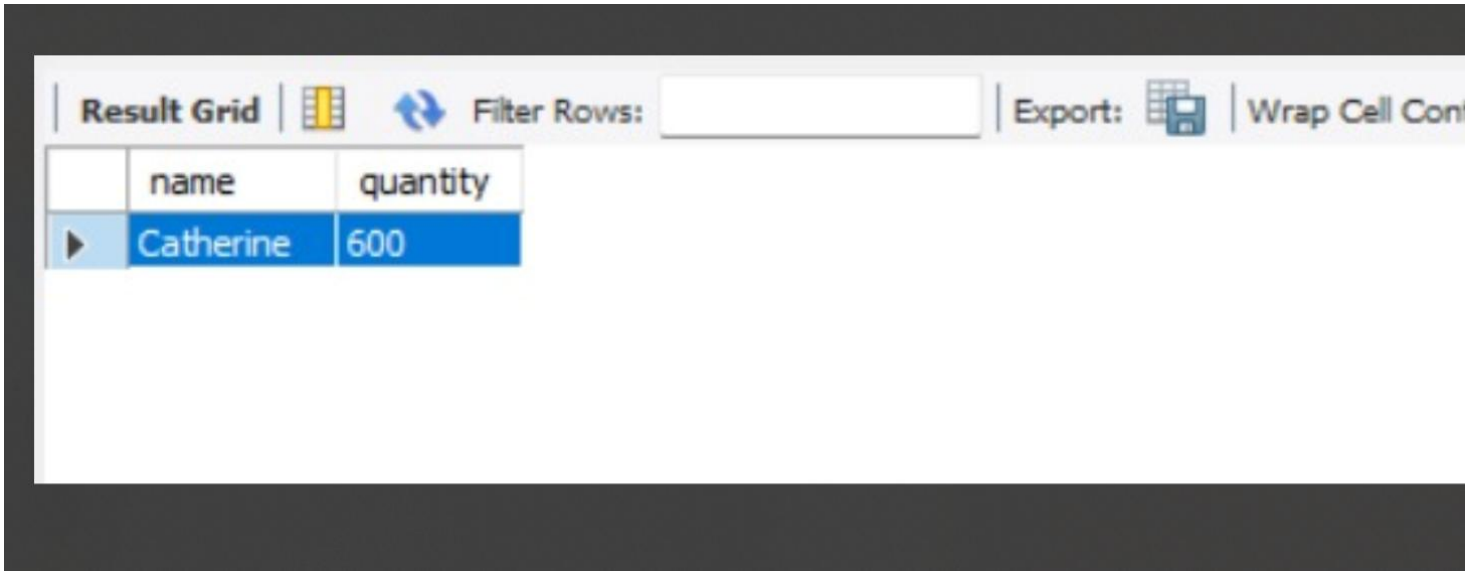
4. SELECT name, quantity FROM Recipient WHERE quantity > 300;


Result Grid | Filter Rows: | Export: | Wrap Cell Co

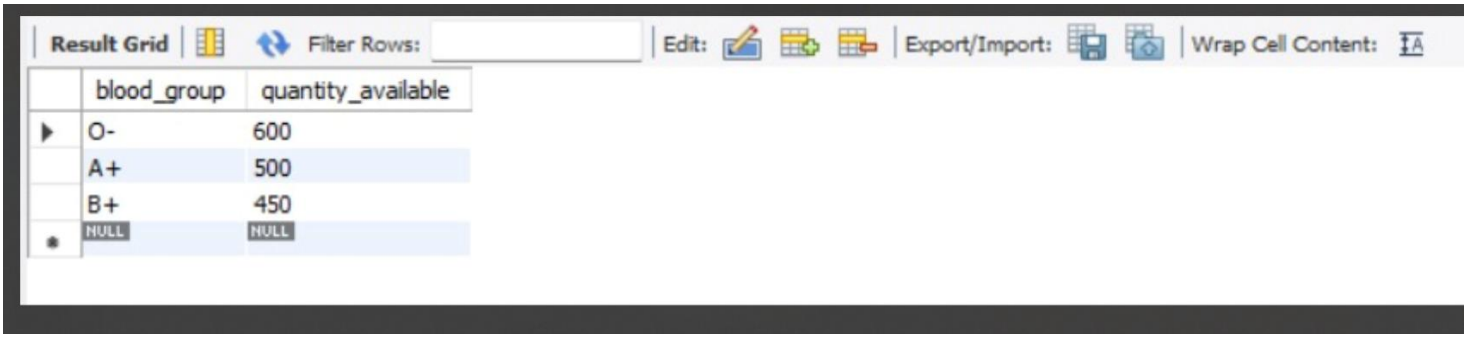
name	quantity
Emily	400

5. SELECT d.name, bd.quantity

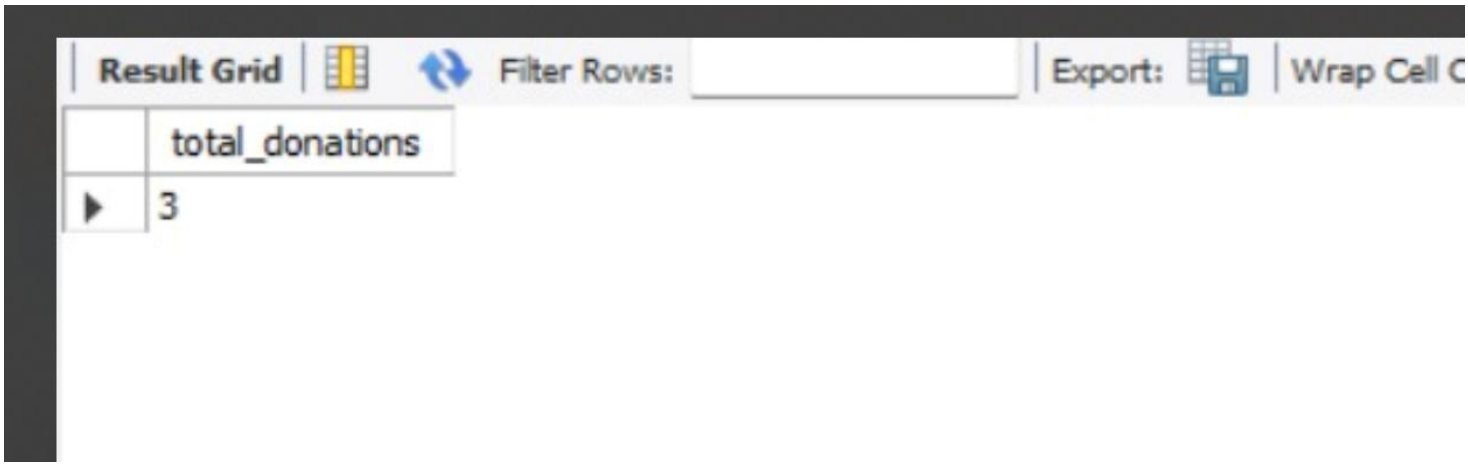
FROM Donor d JOIN Blood_Donation bd ON d.donor_id = bd.donor_idWHERE bd.quantity > 500;



6. SELECT blood_group, quantity_available FROM Blood_Inventory ORDER BY quantity_available DESC;



7. SELECT COUNT(*) AS total_donations FROM Blood_Donation;

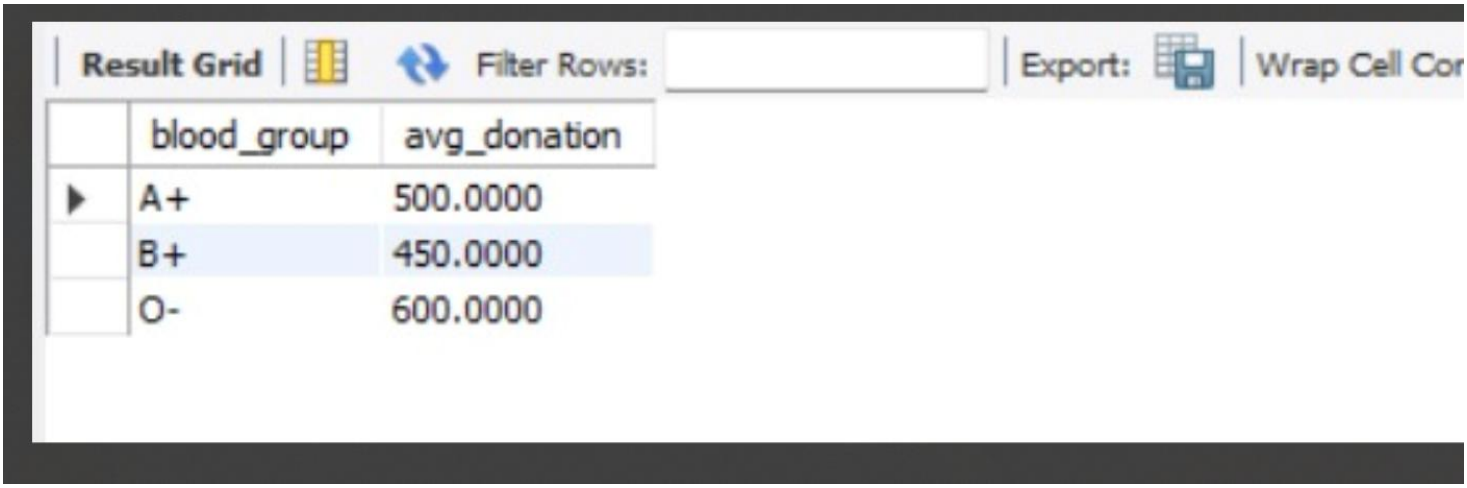


8. SELECT d.blood_group, AVG(bd.quantity) AS avg_donation

FROM Donor d

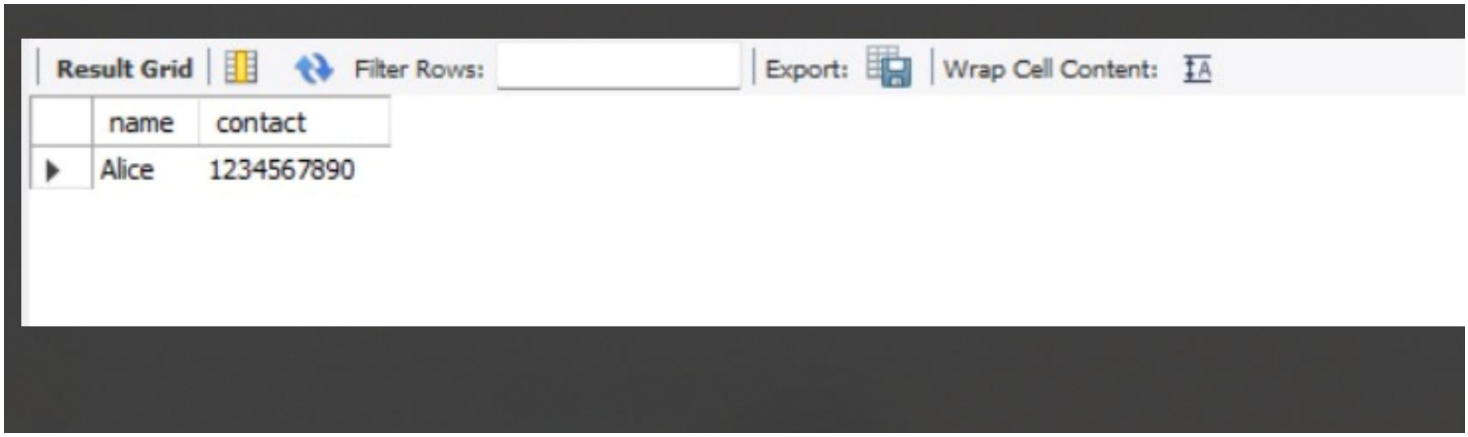
JOIN Blood_Donation bd ON d.donor_id = bd.donor_id

GROUP BY d.blood_group;



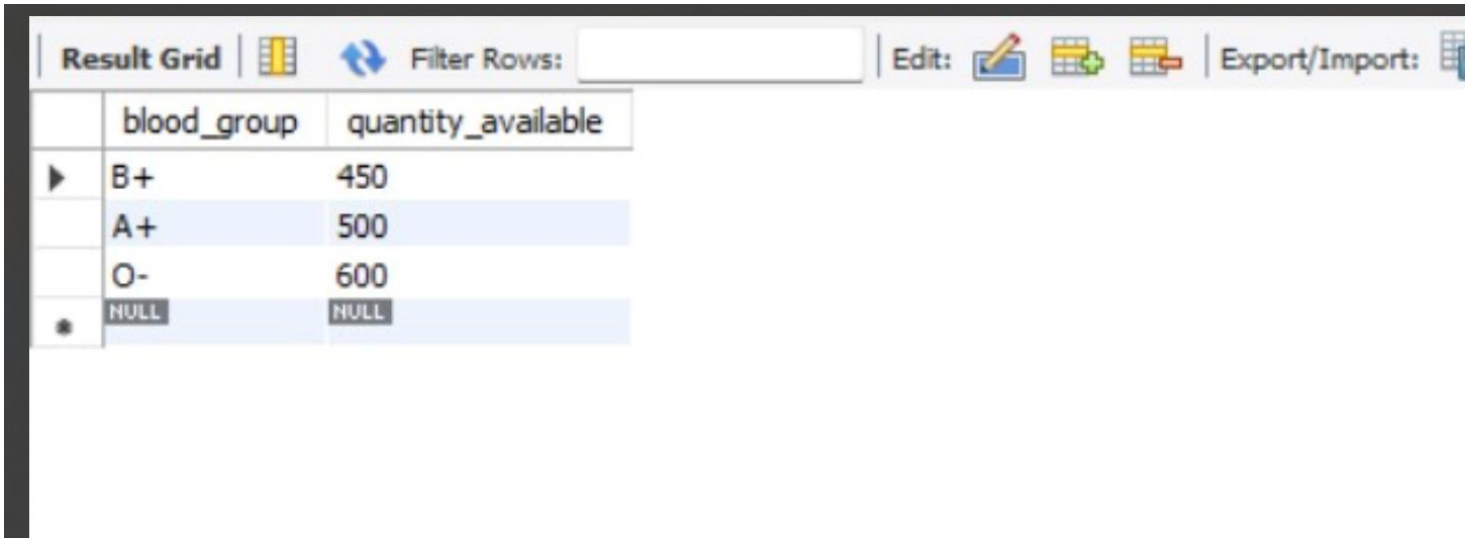
blood_group	avg_donation
A+	500.0000
B+	450.0000
O-	600.0000

9. SELECT name, contact FROM Donor WHERE blood_group = 'A+';



name	contact
Alice	1234567890

10. SELECT blood_group, quantity_available FROM Blood_Inventory ORDER BY quantity_available asc;



Result Grid | Filter Rows: | Edit: | Export/Import:

blood_group	quantity_available
B+	450
A+	500
O-	600
NULL	NULL

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

SUMMARY:

This SQL script sets up a Blood Bank Management System using relational database principles. It creates five main tables: Donor, Hospital, Blood_Donation, Blood_Inventory, and Recipient. These tables store information about donors, hospitals, donation records, blood stock levels, and recipients.

Key features include:

Use of primary and foreign keys to maintain data integrity.

Sample data insertion for testing purposes.

Multiple queries to retrieve and analyze data such as total blood donated, available inventory, recipient details, and donation statistics.

The code demonstrates how SQL can be used to manage real-world data for a blood bank in a structured and efficient way.

CONCLUSION:

The Blood Bank Management SQL code provides a structured and efficient way to store, manage, and analyze blood donation and distribution data. By using relational tables with proper keys and constraints, it ensures data integrity and supports essential operations like tracking donations, managing inventory, and handling recipient details. This system lays a strong foundation for building more advanced features like alerts, donor eligibility checks, and reporting in a real-world blood bank environment.