Department of Information Technology
Faculty of Technology, Dharmsinh Desai University
College Road, Nadiad-387001
October-2021

A Project Report
On
<u>**Medicine Supply Management System**</u>

Department of Information Technology
Faculty of Technology
DD University

Developed by:-

1)Vishnu Sanariya (IT-138) – Department of IT DD University)
2)Sahil Sheta(IT-150) - Department of IT, DD University)

Guided By
Internal Guide:
Prof. Archana N. Vyas.

Department of Information Technology
Faculty of Technology, Dharmsinh Desai University
College Road, Nadiad-387001
October-2021

CERTIFICATE

This is to certify that the project entitled "**Medicine supply Management System**"is a bonafied report of the work carried out by

1) **Mr.VISHNUSANARIYA ,** Student ID No : **19ITUOS024**

2) **Mr.SAHILSHETA**, Student ID No : **19ITUOS144**

of Department of Information Technology, semester V, under the guidance and supervision for the subject Database Management System. They were involved in Project training during academic year 2019-2020.

Prof. Archana N. Vyas
(Project Guide)
Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:

Prof. VipulDabhi
Head , Department of Information Technology,
Faculty of Technology,
Dharmsinh Desai University, Nadiad
Date:

ACKNOWLEDGEMENT

We would like to give our sincere acknowledgement to everybody responsible for the successful completion of our project “MEDICINE SUPPLY MANAGEMENT SYSTEM”.

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of this project.

We owe our deep gratitude to our project guide Prof. Archana N. Vyas, who took been interest on our project work and guided us all along till the completion of our project work by providing all the necessary help for developing a good Database System.

We would also like to thank all our lecturers.

Finally we convoy our acknowledgement to all our friends and family members who directly or indirectly associated with us in the successful completion of the project. We thank one and all.

TABLE OF CONTENTS

1. SYSTEM OVERVIEW

1.1 Project Details:

The online medicine supplier is project for the managing the medicine delivery from the manufacture to the our door step using intermediate entity which is the wholesalers.The customer which want buy the medicine using online medium they must be sing in is they use first time. After the sign/log in they should able to purchase the medicine. This entered medicine is check in the nearest wholesalers' shop by system if available then they get name of the shops and address. And the all type of medicine is available at wholesaler shop. After this is customer want to order then it place order and orders delivery is managed by the manufacture for simplicity. Wholesalers can manage inventory as well as sell records.

1.2 System Analysis:

The current system for purchasing medicine is:

- We need to purchase medicine by manually from the nearest store.
- If the medicine is not available at one store then we need to find it on another store.
- If we purchase medicine online then it takes several days for delivery.

The new system for purchasing medicine is:

- We do not need to go every medical store for purchasing medicine, because system show which medical have your appropriate medicine.
- If any customer purchase medicine online then the delivery of order is done within the day.
- This system also helps the manufacturer that how the demand of the medicine at particular city or area.

1.3 Purpose:

The purpose of this project is the manage the fast delivery of the medicine so that the customer get medicine as soon as possible. Current system is if we want to get the medicine then we need to go every shop and ask for that medicine. But using this system we get the name of the shop which have the medicine we get in emergency.

1.4 Scope:

This purposed system can manage the delivery of medicine. This system also manages the inventory of medicine which the wholesalers have. This system can manage the sales records for the profit and loss statement.

This system will be not help at night. In an emergency customer will visit the store and make a face-to-face purchase. For this we make assumption that manufacture is responsible to manage the delivery of the order.

1.5 Objective:

The overall objective of medicine supplier management system is:

- To provide easily accessibility to customers management.
- To provide easily accessibility to wholesalers' management.
- To provide easily accessibility to purchases report, sales report and stock reports.
- To provide easily accessibility to prepare and printing invoice of the customers.
- To minimize the human error.
- To provide optimal drugs inventory management by monitoring drugs' movement in the unit.

1.6 Advantages:

Using this system, we have some advantages over the current system:

- Aim towards large amount of customer that can purchase online.
- The delivery time is less due to main shop of supplier is closest.
- Anybody can purchase medicine 24×7 using online platform.
- Easily ordered medicine in three steps:
  1. Add your details
  2. Select medicine
  3. Purchase the order

1.7 Disadvantages:

Using this system, we have some disadvantages

- The delivery of the medicine at night not possible.
- In emergency we need to go to the store for medicine

2. E-R DIAGRAM

2.1 User Roles :

This database management based project will able to manage the online ordering and home delivery of medicines. There will be 3 types of user login.

1. Customers
2. Wholesalers
3. Manufacturer

2.2 Role wise requirement listing :

1.) Customers:

Anyone sitting at home can put the prescription in and get information about pharmacies nearby. They can also order the medicines and make payment online. The home delivery facility will also be there.

2.) Wholesalers:

The whole sellers can track on the medicines stock they have. They will sell to the customers using facilities like online payment, making delivery etc. They also can track of sell records.

3.) Manufacturer:

Manufacturers are responsible for the producing medicine which is ordered by the wholesalers. Also manufacturer manages the delivery of the medicine which is ordered by wholesalers.

2.3 Requirement Analysis :

R1>This system can used by customers, wholesaler and manufacturer.

R2>The customers, wholesaler and manufacturer can login into the system, and it can insert, update and delete their details.

R3>The customers can only view the inventory record of the wholesaler. This view limited up to the searched medicine name and at what quantity is available.

R4>The wholesaler can update the inventory of the store. They can add customers details. They also can view the sell record of the store.

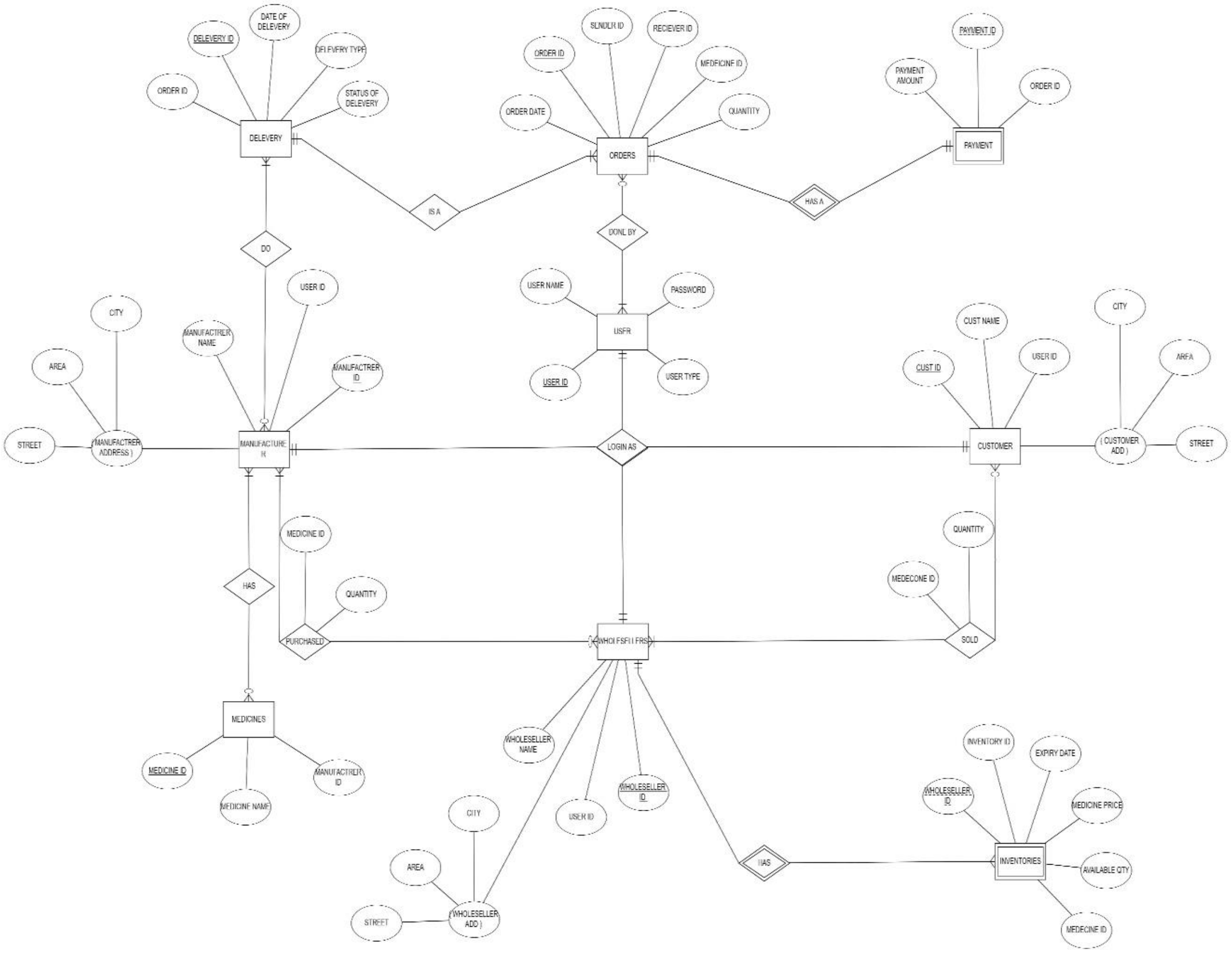
R5>The wholesaler can direct contact the manufacture for purchasing the medicine.

R6>The manufacturer can add the details of the wholesaler only and can add details of medicines produced by manufacturer.

R7>The customers and wholesaler must place order online and also pay for that purchased medicine online.

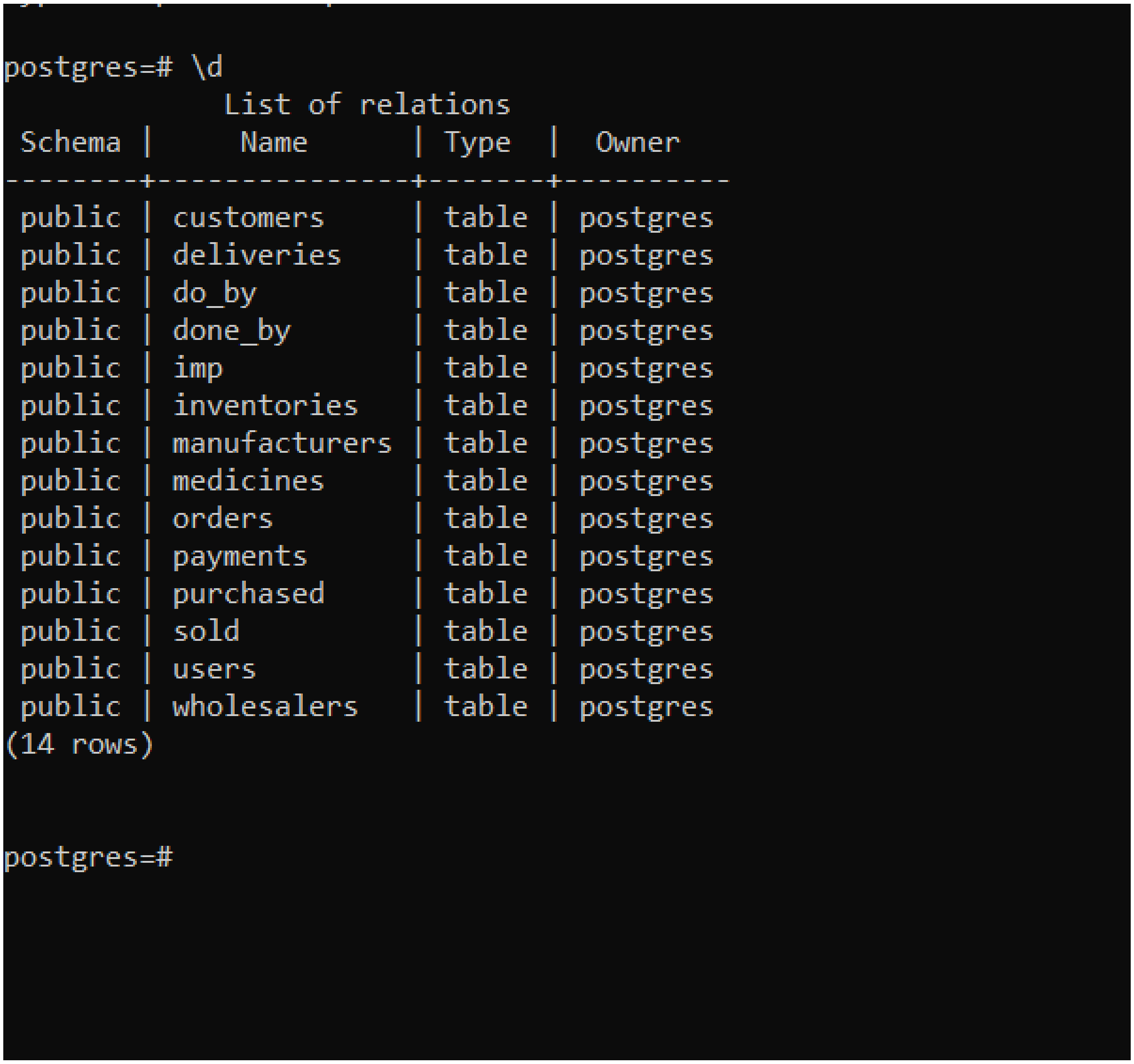
R8> Manufacturer manages deliveries of orders

2.4 Detailed ER Diagram:

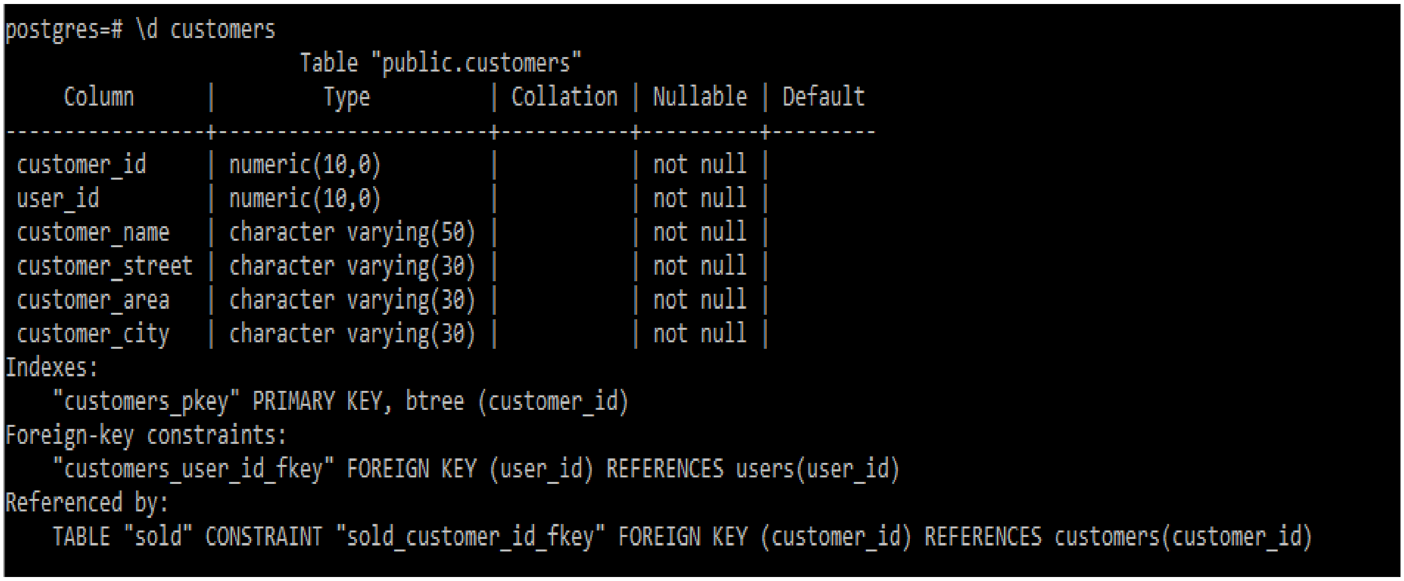


E-R MODEL

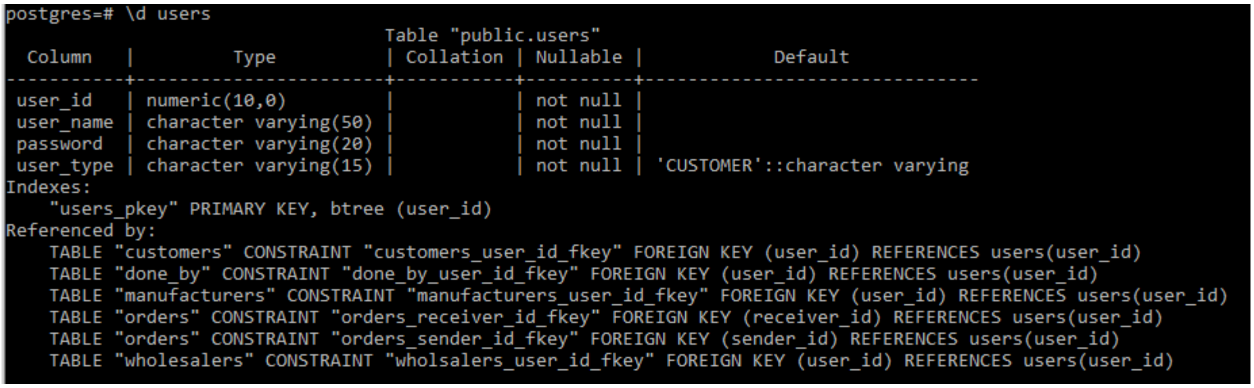
3. DATA DICTIONARY



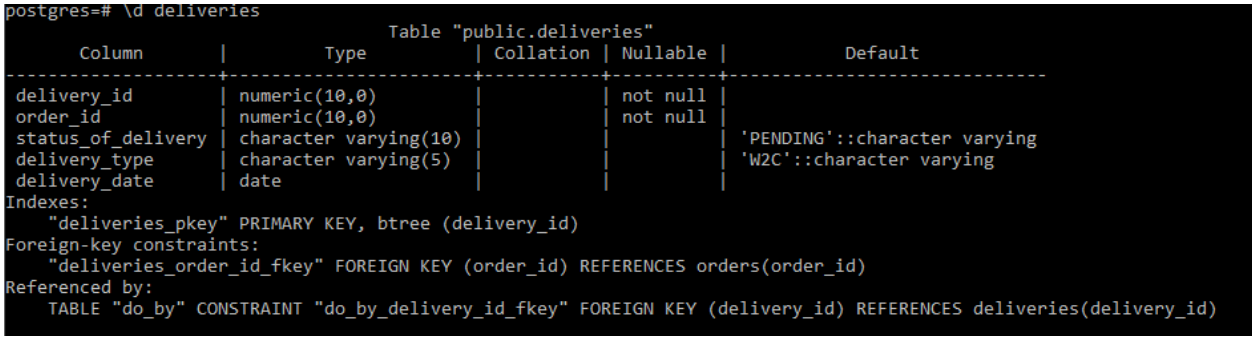
```
postgres=# \d
             List of relations
 Schema |     Name      | Type  |  Owner
--------+---------------+-------+----------
 public | customers     | table | postgres
 public | deliveries    | table | postgres
 public | do_by         | table | postgres
 public | done_by       | table | postgres
 public | imp           | table | postgres
 public | inventories   | table | postgres
 public | manufacturers | table | postgres
 public | medicines     | table | postgres
 public | orders        | table | postgres
 public | payments      | table | postgres
 public | purchased     | table | postgres
 public | sold          | table | postgres
 public | users         | table | postgres
 public | wholesalers   | table | postgres
(14 rows)


postgres=#
```



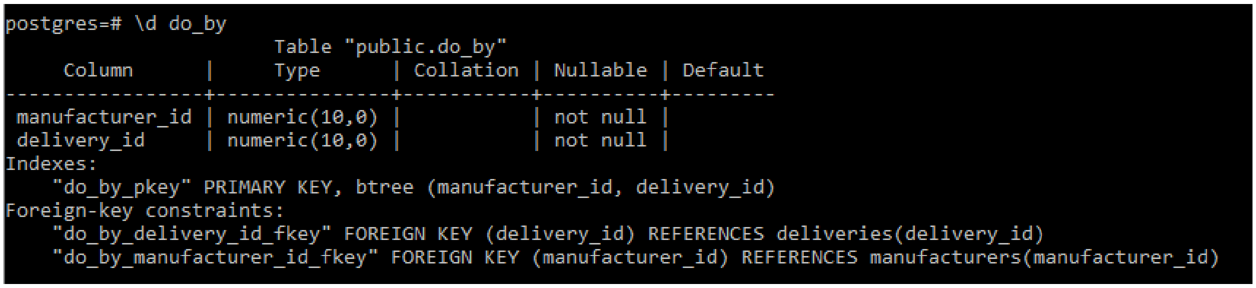
```
postgres=# \d customers
                         Table "public.customers"
     Column      |         Type          | Collation | Nullable | Default
-----------------+-----------------------+-----------+----------+---------
 customer_id     | numeric(10,0)         |           | not null |
 user_id         | numeric(10,0)         |           | not null |
 customer_name   | character varying(50) |           | not null |
 customer_street | character varying(30) |           | not null |
 customer_area   | character varying(30) |           | not null |
 customer_city   | character varying(30) |           | not null |
Indexes:
    "customers_pkey" PRIMARY KEY, btree (customer_id)
Foreign-key constraints:
    "customers_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id)
Referenced by:
    TABLE "sold" CONSTRAINT "sold_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
```



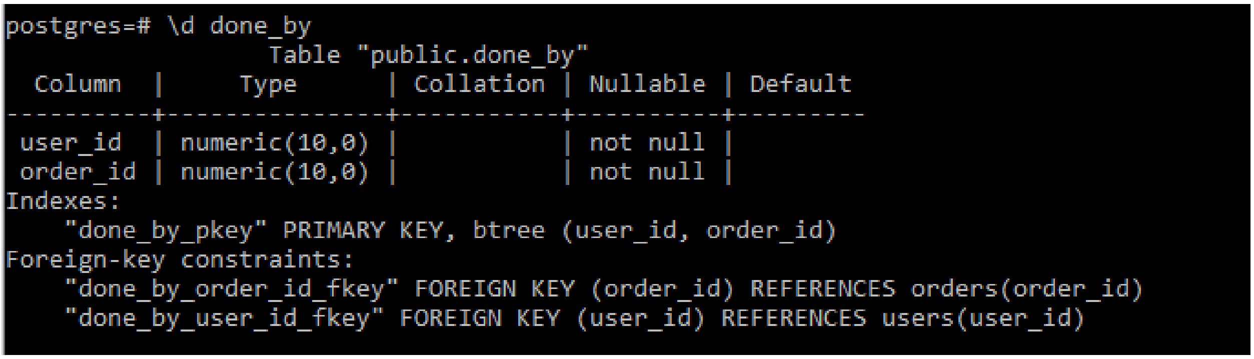
```
postgres=# \d users
                                    Table "public.users"
  Column   |         Type          | Collation | Nullable |            Default
-----------+-----------------------+-----------+----------+-------------------------------
 user_id   | numeric(10,0)         |           | not null |
 user_name | character varying(50) |           | not null |
 password  | character varying(20) |           | not null |
 user_type | character varying(15) |           | not null | 'CUSTOMER'::character varying
Indexes:
    "users_pkey" PRIMARY KEY, btree (user_id)
Referenced by:
    TABLE "customers" CONSTRAINT "customers_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id)
    TABLE "done_by" CONSTRAINT "done_by_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id)
    TABLE "manufacturers" CONSTRAINT "manufacturers_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id)
    TABLE "orders" CONSTRAINT "orders_receiver_id_fkey" FOREIGN KEY (receiver_id) REFERENCES users(user_id)
    TABLE "orders" CONSTRAINT "orders_sender_id_fkey" FOREIGN KEY (sender_id) REFERENCES users(user_id)
    TABLE "wholesalers" CONSTRAINT "wholsalers_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id)
```



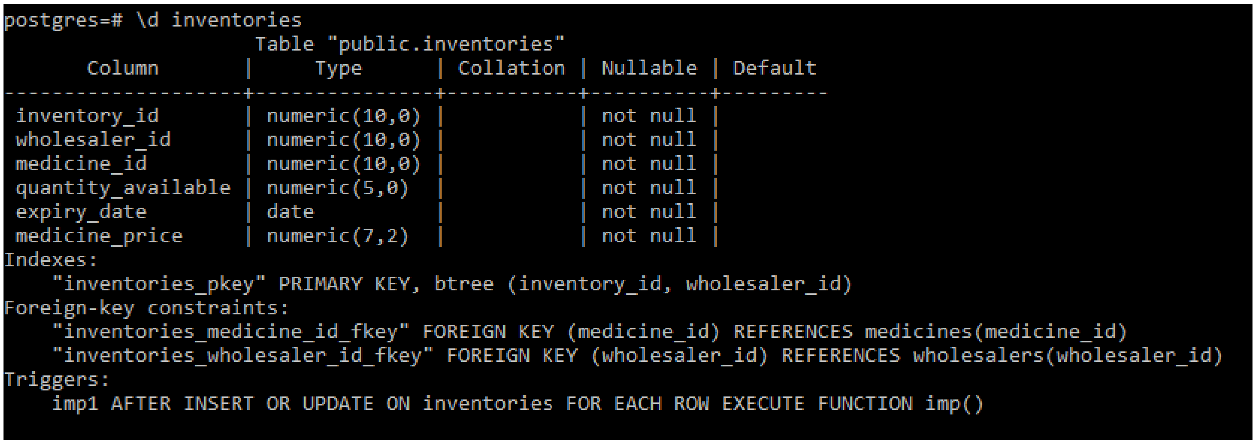
```
postgres=# \d deliveries
                                     Table "public.deliveries"
       Column       |         Type          | Collation | Nullable |            Default
--------------------+-----------------------+-----------+----------+------------------------------
 delivery_id        | numeric(10,0)         |           | not null |
 order_id           | numeric(10,0)         |           | not null |
 status_of_delivery | character varying(10) |           |          | 'PENDING'::character varying
 delivery_type      | character varying(5)  |           |          | 'W2C'::character varying
 delivery_date      | date                  |           |          |
Indexes:
    "deliveries_pkey" PRIMARY KEY, btree (delivery_id)
Foreign-key constraints:
    "deliveries_order_id_fkey" FOREIGN KEY (order_id) REFERENCES orders(order_id)
Referenced by:
    TABLE "do_by" CONSTRAINT "do_by_delivery_id_fkey" FOREIGN KEY (delivery_id) REFERENCES deliveries(delivery_id)
```



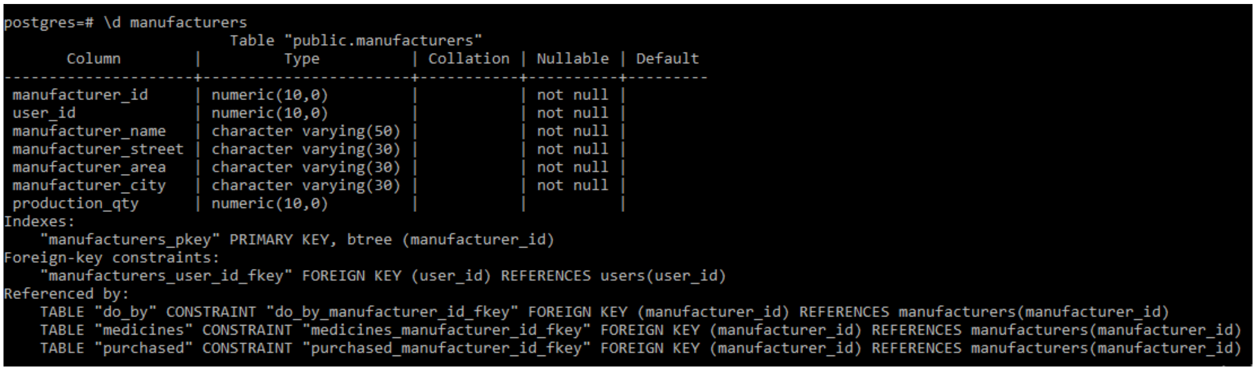
```
postgres=# \d do_by
                      Table "public.do_by"
     Column      |     Type      | Collation | Nullable | Default
-----------------+---------------+-----------+----------+---------
 manufacturer_id | numeric(10,0) |           | not null |
 delivery_id     | numeric(10,0) |           | not null |
Indexes:
    "do_by_pkey" PRIMARY KEY, btree (manufacturer_id, delivery_id)
Foreign-key constraints:
    "do_by_delivery_id_fkey" FOREIGN KEY (delivery_id) REFERENCES deliveries(delivery_id)
    "do_by_manufacturer_id_fkey" FOREIGN KEY (manufacturer_id) REFERENCES manufacturers(manufacturer_id)
```



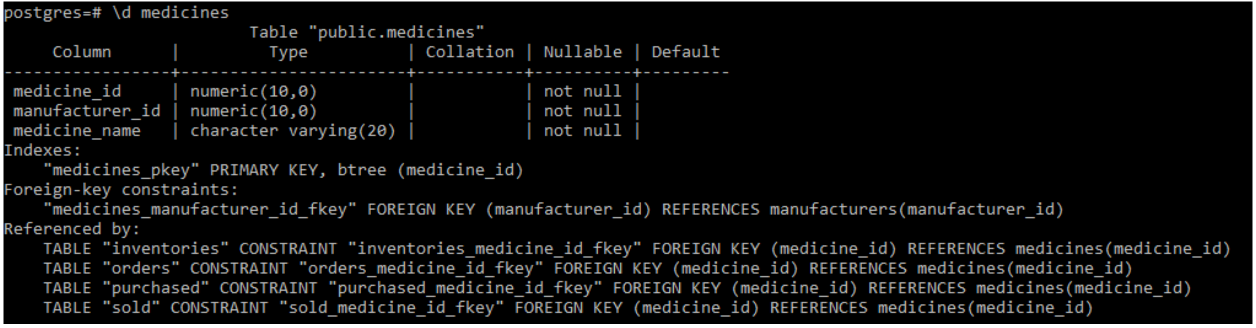
```
postgres=# \d done_by
                 Table "public.done_by"
  Column  |     Type      | Collation | Nullable | Default
----------+---------------+-----------+----------+---------
 user_id  | numeric(10,0) |           | not null |
 order_id | numeric(10,0) |           | not null |
Indexes:
    "done_by_pkey" PRIMARY KEY, btree (user_id, order_id)
Foreign-key constraints:
    "done_by_order_id_fkey" FOREIGN KEY (order_id) REFERENCES orders(order_id)
    "done_by_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id)
```



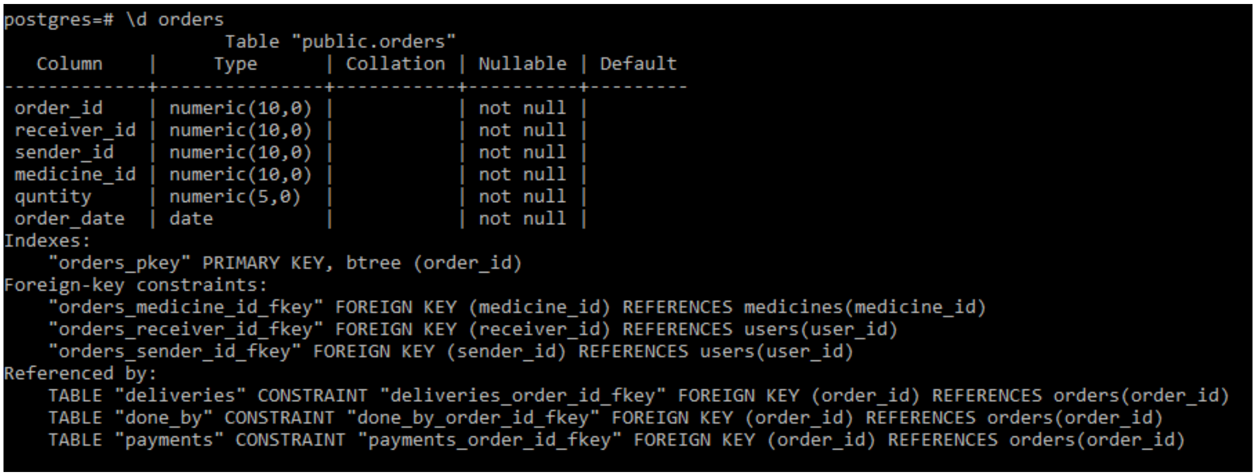
```
postgres=# \d inventories
                     Table "public.inventories"
       Column       |     Type      | Collation | Nullable | Default
--------------------+---------------+-----------+----------+---------
 inventory_id       | numeric(10,0) |           | not null |
 wholesaler_id      | numeric(10,0) |           | not null |
 medicine_id        | numeric(10,0) |           | not null |
 quantity_available | numeric(5,0)  |           | not null |
 expiry_date        | date          |           | not null |
 medicine_price     | numeric(7,2)  |           | not null |
Indexes:
    "inventories_pkey" PRIMARY KEY, btree (inventory_id, wholesaler_id)
Foreign-key constraints:
    "inventories_medicine_id_fkey" FOREIGN KEY (medicine_id) REFERENCES medicines(medicine_id)
    "inventories_wholesaler_id_fkey" FOREIGN KEY (wholesaler_id) REFERENCES wholesalers(wholesaler_id)
Triggers:
    imp1 AFTER INSERT OR UPDATE ON inventories FOR EACH ROW EXECUTE FUNCTION imp()
```



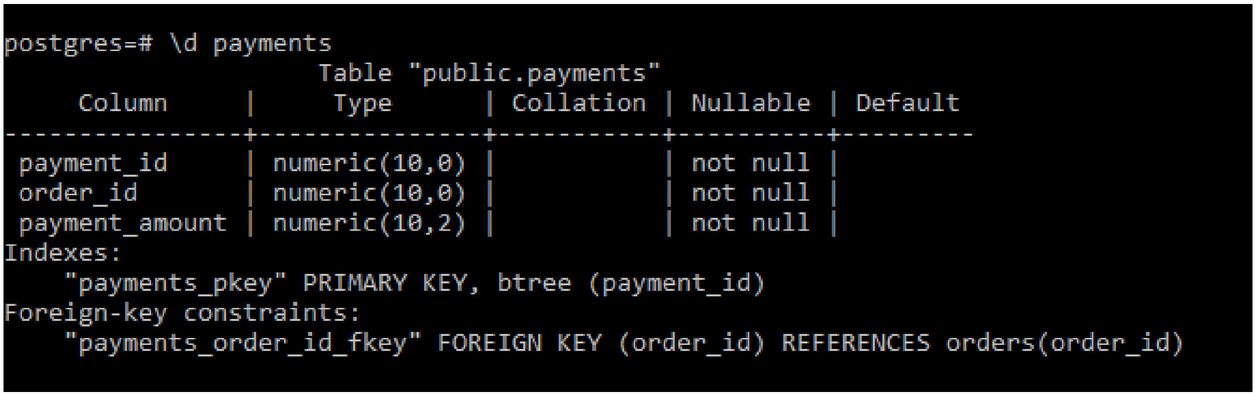
```
postgres=# \d manufacturers
                         Table "public.manufacturers"
       Column        |         Type          | Collation | Nullable | Default
---------------------+-----------------------+-----------+----------+---------
 manufacturer_id     | numeric(10,0)         |           | not null |
 user_id             | numeric(10,0)         |           | not null |
 manufacturer_name   | character varying(50) |           | not null |
 manufacturer_street | character varying(30) |           | not null |
 manufacturer_area   | character varying(30) |           | not null |
 manufacturer_city   | character varying(30) |           | not null |
 production_qty      | numeric(10,0)         |           |          |
Indexes:
    "manufacturers_pkey" PRIMARY KEY, btree (manufacturer_id)
Foreign-key constraints:
    "manufacturers_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id)
Referenced by:
    TABLE "do_by" CONSTRAINT "do_by_manufacturer_id_fkey" FOREIGN KEY (manufacturer_id) REFERENCES manufacturers(manufacturer_id)
    TABLE "medicines" CONSTRAINT "medicines_manufacturer_id_fkey" FOREIGN KEY (manufacturer_id) REFERENCES manufacturers(manufacturer_id)
    TABLE "purchased" CONSTRAINT "purchased_manufacturer_id_fkey" FOREIGN KEY (manufacturer_id) REFERENCES manufacturers(manufacturer_id)
```



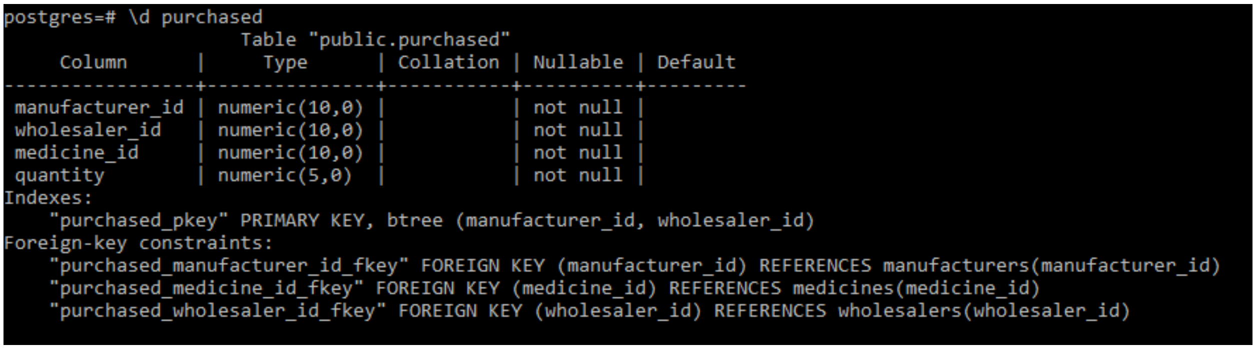
```
postgres=# \d medicines
                        Table "public.medicines"
     Column      |         Type          | Collation | Nullable | Default
-----------------+-----------------------+-----------+----------+---------
 medicine_id     | numeric(10,0)         |           | not null |
 manufacturer_id | numeric(10,0)         |           | not null |
 medicine_name   | character varying(20) |           | not null |
Indexes:
    "medicines_pkey" PRIMARY KEY, btree (medicine_id)
Foreign-key constraints:
    "medicines_manufacturer_id_fkey" FOREIGN KEY (manufacturer_id) REFERENCES manufacturers(manufacturer_id)
Referenced by:
    TABLE "inventories" CONSTRAINT "inventories_medicine_id_fkey" FOREIGN KEY (medicine_id) REFERENCES medicines(medicine_id)
    TABLE "orders" CONSTRAINT "orders_medicine_id_fkey" FOREIGN KEY (medicine_id) REFERENCES medicines(medicine_id)
    TABLE "purchased" CONSTRAINT "purchased_medicine_id_fkey" FOREIGN KEY (medicine_id) REFERENCES medicines(medicine_id)
    TABLE "sold" CONSTRAINT "sold_medicine_id_fkey" FOREIGN KEY (medicine_id) REFERENCES medicines(medicine_id)
```



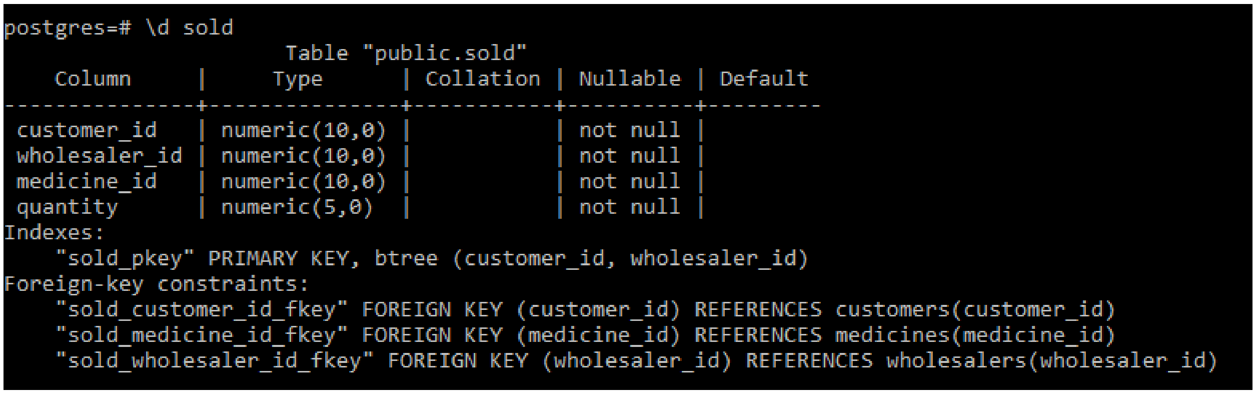
```
postgres=# \d orders
                    Table "public.orders"
   Column    |     Type      | Collation | Nullable | Default
-------------+---------------+-----------+----------+---------
 order_id    | numeric(10,0) |           | not null |
 receiver_id | numeric(10,0) |           | not null |
 sender_id   | numeric(10,0) |           | not null |
 medicine_id | numeric(10,0) |           | not null |
 quntity     | numeric(5,0)  |           | not null |
 order_date  | date          |           | not null |
Indexes:
    "orders_pkey" PRIMARY KEY, btree (order_id)
Foreign-key constraints:
    "orders_medicine_id_fkey" FOREIGN KEY (medicine_id) REFERENCES medicines(medicine_id)
    "orders_receiver_id_fkey" FOREIGN KEY (receiver_id) REFERENCES users(user_id)
    "orders_sender_id_fkey" FOREIGN KEY (sender_id) REFERENCES users(user_id)
Referenced by:
    TABLE "deliveries" CONSTRAINT "deliveries_order_id_fkey" FOREIGN KEY (order_id) REFERENCES orders(order_id)
    TABLE "done_by" CONSTRAINT "done_by_order_id_fkey" FOREIGN KEY (order_id) REFERENCES orders(order_id)
    TABLE "payments" CONSTRAINT "payments_order_id_fkey" FOREIGN KEY (order_id) REFERENCES orders(order_id)
```



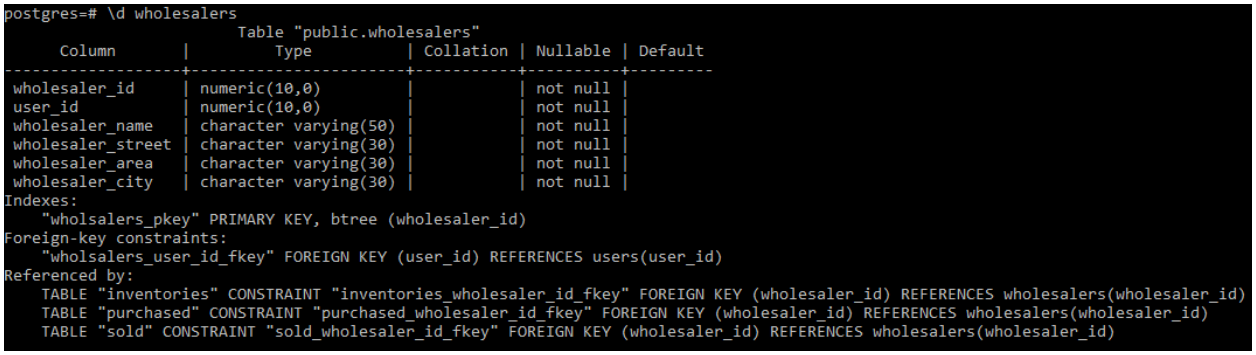
```
postgres=# \d payments
                    Table "public.payments"
     Column     |     Type      | Collation | Nullable | Default
----------------+---------------+-----------+----------+---------
 payment_id     | numeric(10,0) |           | not null |
 order_id       | numeric(10,0) |           | not null |
 payment_amount | numeric(10,2) |           | not null |
Indexes:
    "payments_pkey" PRIMARY KEY, btree (payment_id)
Foreign-key constraints:
    "payments_order_id_fkey" FOREIGN KEY (order_id) REFERENCES orders(order_id)
```



```
postgres=# \d purchased
                    Table "public.purchased"
     Column      |     Type      | Collation | Nullable | Default
-----------------+---------------+-----------+----------+---------
 manufacturer_id | numeric(10,0) |           | not null |
 wholesaler_id   | numeric(10,0) |           | not null |
 medicine_id     | numeric(10,0) |           | not null |
 quantity        | numeric(5,0)  |           | not null |
Indexes:
    "purchased_pkey" PRIMARY KEY, btree (manufacturer_id, wholesaler_id)
Foreign-key constraints:
    "purchased_manufacturer_id_fkey" FOREIGN KEY (manufacturer_id) REFERENCES manufacturers(manufacturer_id)
    "purchased_medicine_id_fkey" FOREIGN KEY (medicine_id) REFERENCES medicines(medicine_id)
    "purchased_wholesaler_id_fkey" FOREIGN KEY (wholesaler_id) REFERENCES wholesalers(wholesaler_id)
```

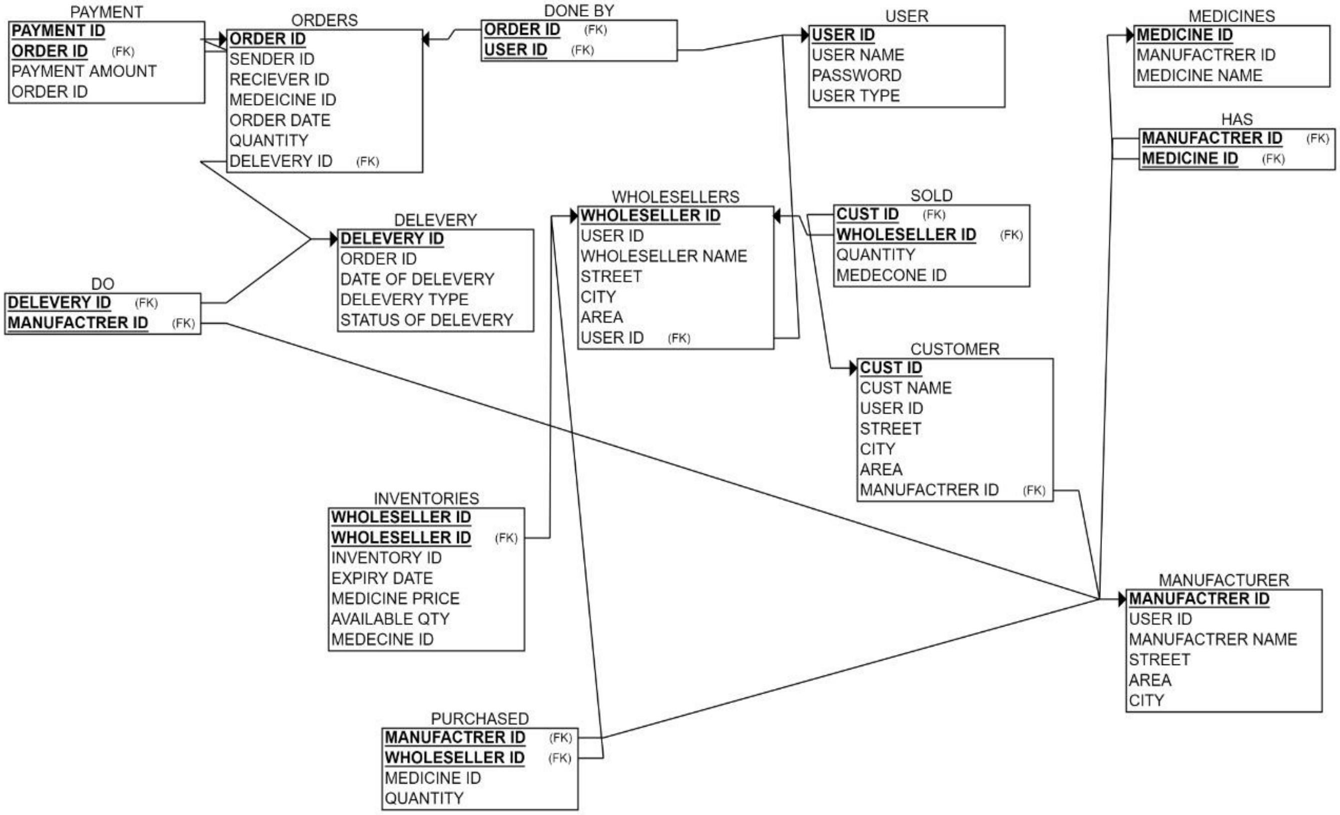


```
postgres=# \d sold
                    Table "public.sold"
    Column     |     Type      | Collation | Nullable | Default
---------------+---------------+-----------+----------+---------
 customer_id   | numeric(10,0) |           | not null |
 wholesaler_id | numeric(10,0) |           | not null |
 medicine_id   | numeric(10,0) |           | not null |
 quantity      | numeric(5,0)  |           | not null |
Indexes:
    "sold_pkey" PRIMARY KEY, btree (customer_id, wholesaler_id)
Foreign-key constraints:
    "sold_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
    "sold_medicine_id_fkey" FOREIGN KEY (medicine_id) REFERENCES medicines(medicine_id)
    "sold_wholesaler_id_fkey" FOREIGN KEY (wholesaler_id) REFERENCES wholesalers(wholesaler_id)
```



```
postgres=# \d wholesalers
                         Table "public.wholesalers"
      Column       |          Type          | Collation | Nullable | Default
-------------------+------------------------+-----------+----------+---------
 wholesaler_id     | numeric(10,0)          |           | not null |
 user_id           | numeric(10,0)          |           | not null |
 wholesaler_name   | character varying(50) |           | not null |
 wholesaler_street | character varying(30) |           | not null |
 wholesaler_area   | character varying(30) |           | not null |
 wholesaler_city   | character varying(30) |           | not null |
Indexes:
    "wholsalers_pkey" PRIMARY KEY, btree (wholesaler_id)
Foreign-key constraints:
    "wholsalers_user_id_fkey" FOREIGN KEY (user_id) REFERENCES users(user_id)
Referenced by:
    TABLE "inventories" CONSTRAINT "inventories_wholesaler_id_fkey" FOREIGN KEY (wholesaler_id) REFERENCES wholesalers(wholesaler_id)
    TABLE "purchased" CONSTRAINT "purchased_wholesaler_id_fkey" FOREIGN KEY (wholesaler_id) REFERENCES wholesalers(wholesaler_id)
    TABLE "sold" CONSTRAINT "sold_wholesaler_id_fkey" FOREIGN KEY (wholesaler_id) REFERENCES wholesalers(wholesaler_id)
```

4. SCHEMA DIAGRAM



SCHEMA DIAGRAM

5. DATABASE IMPLEMENTATION

5.1 Create Schema:

Table 1: Users Table

```
CREATE TABLE USERS(
USER_ID NUMeric(10) NOT NULL,
USER_NAME VARCHAR(50) NOT NULL,
PASSWORD VARCHAR(20) NOT NULL,
USER_TYPE VARCHAR(15) NOT NULL DEFAULT 'CUSTOMER',
CONSTRAINT USER_PK PRIMARY KEY (USER_ID));
```

Table 2: Customers Table

```
CREATE TABLE CUSTOMERS(
CUSTOMER_ID NUMERIC(10) NOT NULL,
USER_ID NUMERIC(10) NOT NULL,
CUSTOMER_NAME VARCHAR(50) NOT NULL,
CUSTOMER_STREET VARCHAR(30) NOT NULL,
CUSTOMER_AREA VARCHAR(30) NOT NULL,
CUSTOMER_CITY VARCHAR(30) NOT NULL,
CONSTRAINT CUSTOMER_PK PRIMARY KEY (CUSTOMER_ID),
CONSTRAINT CUSTOMER_USER_FK FOREIGN KEY (USER_ID)
REFERENCES USERS(USER_ID));
```

Table 3: Wholesalers Table

```
CREATE TABLE WHOLESALERS (
WHOLESALER_ID NUMERIC(10) NOT NULL,
USER_ID NUMERIC(10) NOT NULL,
WHOLESALER_NAME VARCHAR(50) NOT NULL,
WHOLESALER_STREET VARCHAR(30) NOT NULL,
WHOLESALER_AREA VARCHAR(30) NOT NULL,
WHOLESALER_CITY VARCHAR(30) NOT NULL,
CONSTRAINT WHOLESALER_PK PRIMARY KEY(WHOLESALER_ID),
CONSTRAINT WHOLESALER_USER_FK FOREIGN KEY (USER_ID)
REFERENCES USERS(USER_ID));
```

Table 4: Manufacturers Table

```
CREATE TABLE MANUFACTURERS (
MANUFACTURER_ID NUMERIC(10) NOT NULL,
USER_ID NUMERIC(10) NOT NULL,
MANUFACTURER_NAME VARCHAR(50) NOT NULL,
MANUFACTURER_STREET VARCHAR(30) NOT NULL,
MANUFACTURER_AREA VARCHAR(30) NOT NULL,
MANUFACTURER_CITY VARCHAR(30) NOT NULL,
CONSTRAINT MANUFACTURER_PK PRIMARY KEY
(MANUFACTURER_ID),
CONSTRAINT MANUFACTURER_USER_FK FOREIGN KEY (USER_ID)
REFERENCES USERS(USER_ID));
```

Table 5: Medicines Table

```
CREATE TABLE MEDICINES (
MEDICINE_ID NUMERIC(10) NOT NULL,
MANUFACTURER_ID NUMERIC(10) NOT NULL,
MEDICINE_NAME VARCHAR(20) NOT NULL,
CONSTRAINT MEDICINES_PK PRIMARY KEY (MEDICINE_ID),
CONSTRAINT MEDICINES_MANUFACTURERS_FK FOREIGN KEY
(MANUFACTURER_ID) REFERENCES MANUFACTURERS
(MANUFACTURER_ID));
```

Table 6: INVENTORIES Table

```
CREATE TABLE INVENTORIES(
INVENTORY_ID NUMERIC(10) NOT NULL,
WHOLESALER_ID NUMERIC(10) NOT NULL,
MEDICINE_ID NUMERIC(10) NOT NULL,
QUANTITY_AVAILABLE NUMERIC(5) NOT NULL,
EXPIRY_DATE DATE NOT NULL,
MEDICINE_PRICE NUMERIC(7,2) NOT NULL,
CONSTRAINT INVENTORIES_PK PRIMARY KEY (INVENTORY_ID,
WHOLESALER_ID),
CONSTRAINT INVENTORIES_WHOLESALER_FK FOREIGN KEY
(WHOLESALER_ID) REFERENCES WHOLESALERS
(WHOLESALER_ID),
CONSTRAINT INVENTORIES_MEDICINE_FK FOREIGN KEY
(MEDICINE_ID) REFERENCES MEDICINES(MEDICINE_ID));
```

Table 7: Orders Table

```
CREATE TABLE ORDERS(
ORDER_ID NUMERIC(10) NOT NULL,
RECEIVER_ID NUMERIC(10) NOT NULL,
SENDER_ID NUMERIC(10) NOT NULL,
MEDICINE_ID NUMERIC(10) NOT NULL,
QUNTITY NUMERIC(5) NOT NULL,
ORDER_DATE DATE NOT NULL,
CONSTRAINT ORDERS_PK PRIMARY KEY (ORDER_ID),
CONSTRAINT ORDERS_RECEIVER_FK FOREIGN KEY
(RECEIVER_ID) REFERENCES USERS(USER_ID),
CONSTRAINT ORDERS_SENDER_FK FOREIGN KEY (SENDER_ID)
REFERENCES USERS(USER_ID),
CONSTRAINT ORDERS_MEDICINES_FK FOREIGN KEY
(MEDICINE_ID) REFERENCES MEDICINES (MEDICINE_ID));
```

Table 8: Deliveries Table

```
CREATE TABLE DELIVERIES (
DELIVERY_ID NUMERIC(10) NOT NULL,
ORDER_ID NUMERIC(10) NOT NULL,
STATUS_OF_DELIVERY VARCHAR(10) DEFAULT 'PENDING',
DELIVERY_TYPE VARCHAR(5) DEFAULT 'W2C',
DELIVERY_DATE DATE,
CONSTRAINT DELIVERIES_PK PRIMARY KEY (DELIVERY_ID),
CONSTRAINT DELIVERIES_ORDERS_FK FOREIGN KEY (ORDER_ID)
REFERENCES ORDERS(ORDER_ID));
```

Table 9: Payments Table

```
CREATE TABLE PAYMENTS(
PAYMENT_ID NUMERIC(10) NOT NULL,
ORDER_ID NUMERIC(10) NOT NULL,
PAYMENT_AMOUNT NUMERIC(10,2) NOT NULL,
CONSTRAINT PAYMENTS_PK PRIMARY KEY (PAYMENT_ID),
CONSTRAINT PAYMENTS_ORDERS_FK FOREIGN KEY (ORDER_ID)
REFERENCES ORDERS(ORDER_ID));
```

Table 10: Sold Table

```
CREATE TABLE SOLD(
CUSTOMER_ID NUMERIC(10) NOT NULL,
WHOLESALER_ID NUMERIC(10) NOT NULL,
MEDICINE_ID NUMERIC(10) NOT NULL,
QUANTITY NUMERIC(5) NOT NULL,
CONSTRAINT SOLD_PK PRIMARY KEY (CUSTOMER_ID,
WHOLESALER_ID),
CONSTRAINT SOLD_CUSTOMERS_FK FOREIGN KEY (CUSTOMER_ID)
REFERENCES CUSTOMERS (CUSTOMER_ID),
CONSTRAINT SOLD_WHOLESALERS_FK FOREIGN KEY
(WHOLESALER_ID) REFERENCES WHOLESALERS
(WHOLESALER_ID),
CONSTRAINT SOLD_MEDICINE_FK FOREIGN KEY (MEDICINE_ID)
REFERENCES MEDICINES (MEDICINE_ID));
```

Table 11: Purchased Table

```
CREATE TABLE PURCHASED(
MANUFACTURER_ID NUMERIC(10) NOT NULL,
WHOLESALER_ID NUMERIC(10) NOT NULL,
MEDICINE_ID NUMERIC(10) NOT NULL,
QUANTITY NUMERIC(5) NOT NULL,
CONSTRAINT PURCHASED_PK PRIMARY KEY (MANUFACTURER_ID,
WHOLESALER_ID),
CONSTRAINT PURCHASED_CUSTOMERS_FK FOREIGN KEY
(MANUFACTURER_ID) REFERENCES MANUFACTURERS
(MANUFACTURER_ID),
CONSTRAINT PURCHASED_WHOLESALERS_FK FOREIGN KEY
(WHOLESALER_ID) REFERENCES WHOLESALERS
(WHOLESALER_ID),
CONSTRAINT PURCHASED_MEDICINE_FK FOREIGN KEY
(MEDICINE_ID) REFERENCES MEDICINES (MEDICINE_ID));
```

Table 12: Do Table

```
CREATE TABLE DO_BY(
MANUFACTURER_ID NUMERIC (10) NOT NULL,
DELIVERY_ID NUMERIC (10) NOT NULL,
CONSTRAINT DO_PK PRIMARY KEY (MANUFACTURER_ID,
DELIVERY_ID),
CONSTRAINT DO_MANUFACTURER_FK FOREIGN KEY
(MANUFACTURER_ID) REFERENCES MANUFACTURERS
(MANUFACTURER_ID),
CONSTRAINT DO_DELIVERIES_FK FOREIGN KEY (DELIVERY_ID)
REFERENCES DELIVERIES (DELIVERY_ID));
```

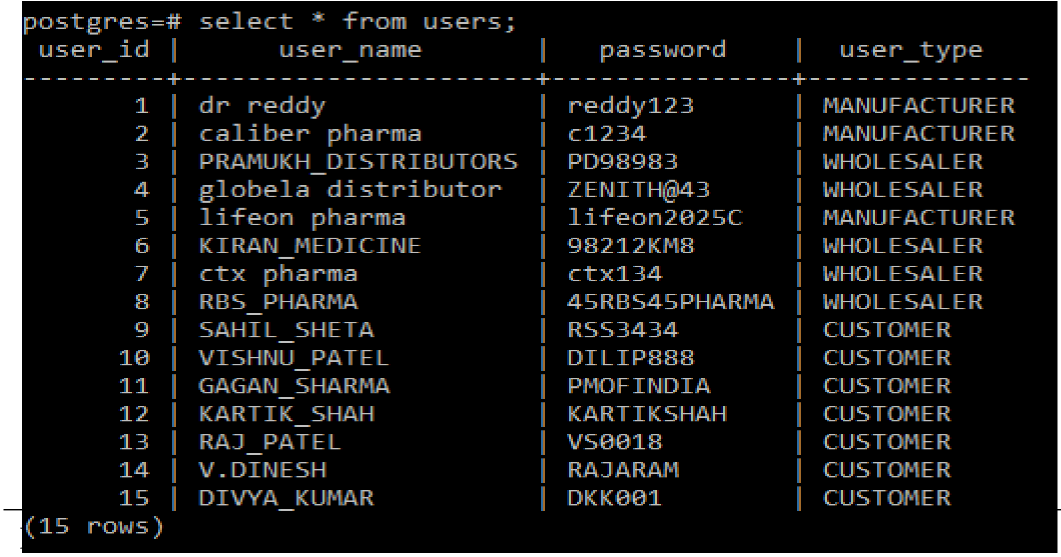
Table 13: Done By Table

```
CREATE TABLE DONE_BY(
USER_ID NUMERIC(10) NOT NULL,
ORDER_ID NUMERIC (10) NOT NULL,
CONSTRAINT DONE_BY_PK PRIMARY KEY (USER_ID, ORDER_ID),
CONSTRAINT DONE_BY_USER_FK FOREIGN KEY (USER_ID)
REFERENCES USERS (USER_ID),
CONSTRAINT DONE_BY_ORDER_FK FOREIGN KEY (ORDER_ID)
REFERENCES ORDERS (ORDER_ID));
```

5.2 Insert Data Values:

Table 1: Users Table

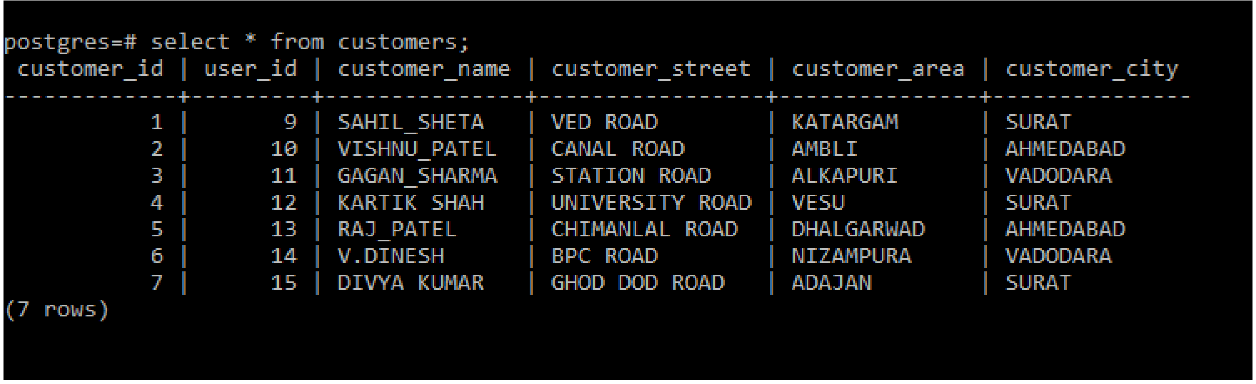
INSERT INTO USERS (USER_ID,USER_NAME, PASSWORD, USER_TYPE) VALUES ('1','dr reddy', 'reddy123', 'MANUFACTURER');
INSERT INTO USERS(USER_ID,USER_NAME, PASSWORD, USER_TYPE) VALUES ('2','caliber pharma', 'c1234', 'MANUFACTURER');
INSERT INTO USERS (USER_ID,USER_NAME, PASSWORD, USER_TYPE) VALUES ('3','PRAMUKH_DISTRIBUTORS', 'PD98983', 'WHOLESALER');
INSERT INTO USERS (USER_ID,USER_NAME, PASSWORD, USER_TYPE) VALUES ('4','globela distributor', 'ZENITH@43', 'WHOLESALER');
INSERT INTO USERS (USER_ID,USER_NAME, PASSWORD, USER_TYPE) VALUES ('5','lifeon pharma', 'lifeon2025C', 'MANUFACTURER');
INSERT INTO USERS (USER_ID,USER_NAME, PASSWORD, USER_TYPE) VALUES ('6','KIRAN_MEDICINE', '98212KM8', 'WHOLESALER');
INSERT INTO USERS (USER_ID,USER_NAME, PASSWORD, USER_TYPE) VALUES ('7','ctx pharma', 'ctx134', 'WHOLESALER');
INSERT INTO USERS (USER_ID,USER_NAME, PASSWORD, USER_TYPE) VALUES ('8','RBS_PHARMA', '45RBS45PHARMA', 'WHOLESALER');
INSERT INTO USERS (USER_ID,USER_NAME, PASSWORD, USER_TYPE) VALUES ('9','SAHIL_SHETA', 'RSS3434', 'CUSTOMER');
INSERT INTO USERS (USER_ID,USER_NAME, PASSWORD, USER_TYPE) VALUES ('10','VISHNU_PATEL', 'DILIP888', 'CUSTOMER');
INSERT INTO USERS (USER_ID,USER_NAME, PASSWORD, USER_TYPE) VALUES ('11','GAGAN_SHARMA', 'PMOFINDIA', 'CUSTOMER');
INSERT INTO USERS (USER_ID,USER_NAME, PASSWORD, USER_TYPE) VALUES ('12','KARTIK_SHAH', 'KARTIKSHAH', 'CUSTOMER');
INSERT INTO USERS (USER_ID,USER_NAME, PASSWORD, USER_TYPE) VALUES ('13','RAJ_PATEL', 'VS0018', 'CUSTOMER');
INSERT INTO USERS (USER_ID,USER_NAME, PASSWORD, USER_TYPE) VALUES ('14','V.DINESH', 'RAJARAM', 'CUSTOMER');
INSERT INTO USERS (USER_ID,USER_NAME, PASSWORD, USER_TYPE) VALUES ('15','DIVYA_KUMAR', 'DKK001', 'CUSTOMER');



```
postgres=# select * from users;
 user_id |      user_name       |   password    |  user_type
---------+----------------------+---------------+--------------
       1 | dr reddy             | reddy123      | MANUFACTURER
       2 | caliber pharma       | c1234         | MANUFACTURER
       3 | PRAMUKH_DISTRIBUTORS | PD98983       | WHOLESALER
       4 | globela distributor  | ZENITH@43     | WHOLESALER
       5 | lifeon pharma        | lifeon2025C   | MANUFACTURER
       6 | KIRAN_MEDICINE       | 98212KM8      | WHOLESALER
       7 | ctx pharma           | ctx134        | WHOLESALER
       8 | RBS_PHARMA           | 45RBS45PHARMA | WHOLESALER
       9 | SAHIL_SHETA          | RSS3434       | CUSTOMER
      10 | VISHNU_PATEL         | DILIP888      | CUSTOMER
      11 | GAGAN_SHARMA         | PMOFINDIA     | CUSTOMER
      12 | KARTIK_SHAH          | KARTIKSHAH    | CUSTOMER
      13 | RAJ_PATEL            | VS0018        | CUSTOMER
      14 | V.DINESH             | RAJARAM       | CUSTOMER
      15 | DIVYA_KUMAR          | DKK001        | CUSTOMER
(15 rows)
```

Table 2: Customers Table

INSERT INTO CUSTOMERS (CUSTOMER_ID,USER_ID, CUSTOMER_NAME, CUSTOMER_STREET, CUSTOMER_AREA,CUSTOMER_CITY) VALUES ('1','9','SAHIL_SHETA', 'VED ROAD', 'KATARGAM', 'SURAT');
INSERT INTO CUSTOMERS (CUSTOMER_ID,USER_ID, CUSTOMER_NAME, CUSTOMER_STREET, CUSTOMER_AREA,CUSTOMER_CITY) VALUES ('2','10', 'VISHNU_PATEL', 'CANAL ROAD', 'AMBLI', 'AHMEDABAD');
INSERT INTO CUSTOMERS (CUSTOMER_ID,USER_ID, CUSTOMER_NAME, CUSTOMER_STREET, CUSTOMER_AREA,CUSTOMER_CITY) VALUES ('3','11', 'GAGAN_SHARMA', 'STATION ROAD', 'ALKAPURI', 'VADODARA');
INSERT INTO CUSTOMERS (CUSTOMER_ID,USER_ID, CUSTOMER_NAME, CUSTOMER_STREET, CUSTOMER_AREA,CUSTOMER_CITY) VALUES ('4','12', 'KARTIK SHAH', 'UNIVERSITY ROAD', 'VESU', 'SURAT');
INSERT INTO CUSTOMERS (CUSTOMER_ID,USER_ID, CUSTOMER_NAME, CUSTOMER_STREET, CUSTOMER_AREA,CUSTOMER_CITY) VALUES ('5','13','RAJ_PATEL', 'CHIMANLAL ROAD', 'DHALGARWAD', 'AHMEDABAD');
INSERT INTO CUSTOMERS (CUSTOMER_ID,USER_ID, CUSTOMER_NAME, CUSTOMER_STREET, CUSTOMER_AREA,CUSTOMER_CITY) VALUES ('6','14', 'V.DINESH', 'BPC ROAD', 'NIZAMPURA', 'VADODARA');
INSERT INTO CUSTOMERS (CUSTOMER_ID,USER_ID, CUSTOMER_NAME, CUSTOMER_STREET, CUSTOMER_AREA,CUSTOMER_CITY) VALUES ('7','15', 'DIVYA KUMAR', 'GHOD DOD ROAD', 'ADAJAN', 'SURAT');



```
postgres=# select * from customers;
 customer_id | user_id | customer_name | customer_street | customer_area | customer_city
-------------+---------+---------------+-----------------+---------------+---------------
           1 |       9 | SAHIL_SHETA   | VED ROAD        | KATARGAM      | SURAT
           2 |      10 | VISHNU_PATEL  | CANAL ROAD      | AMBLI         | AHMEDABAD
           3 |      11 | GAGAN_SHARMA  | STATION ROAD    | ALKAPURI      | VADODARA
           4 |      12 | KARTIK SHAH   | UNIVERSITY ROAD | VESU          | SURAT
           5 |      13 | RAJ_PATEL     | CHIMANLAL ROAD  | DHALGARWAD    | AHMEDABAD
           6 |      14 | V.DINESH      | BPC ROAD        | NIZAMPURA     | VADODARA
           7 |      15 | DIVYA KUMAR   | GHOD DOD ROAD   | ADAJAN        | SURAT
(7 rows)
```

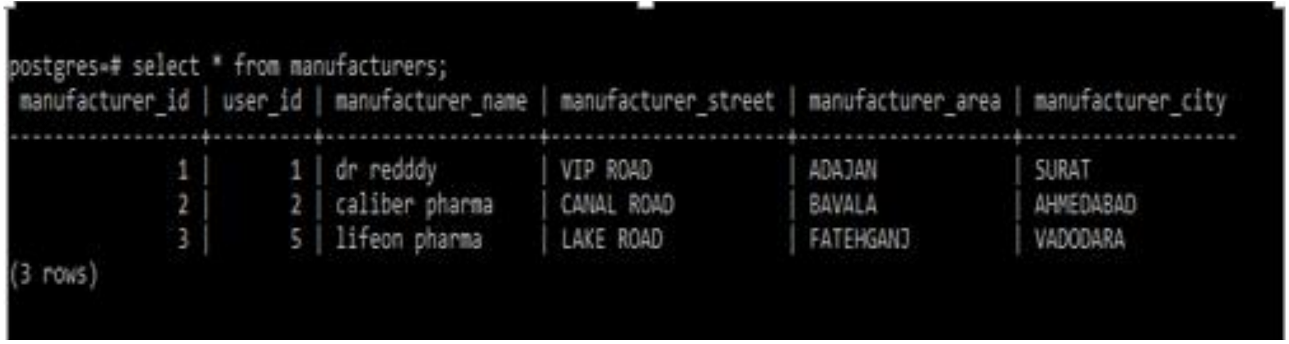
Table 3: Wholesalers Table

INSERT INTO WHOLESALERS (WHOLESALER_ID,USER_ID, WHOLESALER_NAME, WHOLESALER_STREET, WHOLESALER_AREA, WHOLESALER_CITY) VALUES ('1','3', 'PRAMUKH DISTRIBUTORS', 'RANDER ROAD', 'PALANPUR PATIA', 'SURAT');
INSERT INTO WHOLESALERS (WHOLESALER_ID,USER_ID, WHOLESALER_NAME, WHOLESALER_STREET, WHOLESALER_AREA, WHOLESALER_CITY) VALUES ('2','4', 'globela distributor', 'VARACHHA MAIN ROAD', 'VARACHHA', 'SURAT');
INSERT INTO WHOLESALERS (WHOLESALER_ID,USER_ID, WHOLESALER_NAME, WHOLESALER_STREET, WHOLESALER_AREA, WHOLESALER_CITY) VALUES ('3','6', 'KIRAN_MEDICINE', 'SALAPOSE ROAD', 'BHADRA', 'AHMEDABAD');

```
INSERT INTO WHOLESALERS (WHOLESALER_ID,USER_ID, WHOLESALER_NAME,
WHOLESALER_STREET, WHOLESALER_AREA, WHOLESALER_CITY) VALUES
('4','7', 'ctxpharma', 'ASHRAM ROAD', 'ASHRAM ROAD',
'AHMEDABAD');
INSERT INTO WHOLESALERS (WHOLESALER_ID,USER_ID, WHOLESALER_NAME,
WHOLESALER_STREET, WHOLESALER_AREA, WHOLESALER_CITY) VALUES
('5','8', 'RBS_PHARMA DISTRIBUTORS', 'KHARIVAV ROAD', 'RAOPURA',
'VADODARA');
```

Table 4: Manufacturers Table

```
INSERT INTO MANUFACTURERS (MANUFACTURER_ID,USER_ID, MANUFACTURER_NAME,
MANUFACTURER_STREET, MANUFACTURER_AREA, MANUFACTURER_CITY)
VALUES ('1','1', 'drredddy', 'VIP ROAD',
'ADAJAN', 'SURAT');
INSERT INTO MANUFACTURERS (MANUFACTURER_ID,USER_ID, MANUFACTURER_NAME,
MANUFACTURER_STREET, MANUFACTURER_AREA, MANUFACTURER_CITY)
VALUES ('2','2','caliber pharma','CANAL ROAD', 'BAVALA',
'AHMEDABAD');
INSERT INTO MANUFACTURERS (MANUFACTURER_ID,USER_ID, MANUFACTURER_NAME,
MANUFACTURER_STREET, MANUFACTURER_AREA, MANUFACTURER_CITY)
VALUES ('3','5','lifeon pharma','LAKE ROAD',
'FATEHGANJ','VADODARA');
```

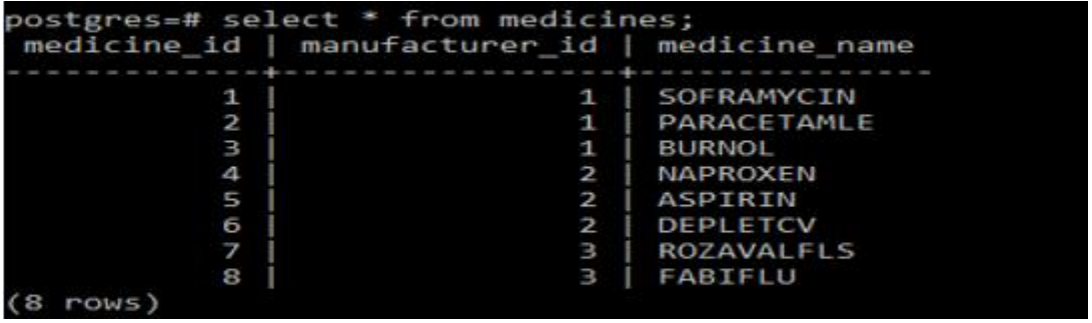


```
postgres=# select * from manufacturers;
 manufacturer_id | user_id | manufacturer_name | manufacturer_street | manufacturer_area | manufacturer_city
-----------------+---------+-------------------+---------------------+-------------------+-------------------
               1 |       1 | dr redddy         | VIP ROAD            | ADAJAN            | SURAT
               2 |       2 | caliber pharma    | CANAL ROAD          | BAVALA            | AHMEDABAD
               3 |       5 | lifeon pharma     | LAKE ROAD           | FATEHGANJ         | VADODARA
(3 rows)
```

Table 5: Medicines Table

```
INSERT INTO MEDICINES (MEDICINE_ID,MANUFACTURER_ID, MEDICINE_NAME)
VALUES ('1','1', 'SOFRAMYCIN');
INSERT INTO MEDICINES (MEDICINE_ID,MANUFACTURER_ID, MEDICINE_NAME)
VALUES ('2','1', 'PARACETAMLE');
INSERT INTO MEDICINES (MEDICINE_ID,MANUFACTURER_ID, MEDICINE_NAME)
VALUES ('3','1', 'BURNOL');
INSERT INTO MEDICINES (MEDICINE_ID,MANUFACTURER_ID, MEDICINE_NAME)
VALUES ('4','2', 'NAPROXEN');
INSERT INTO MEDICINES (MEDICINE_ID,MANUFACTURER_ID, MEDICINE_NAME)
VALUES ('5','2', 'ASPIRIN');
INSERT INTO MEDICINES (MEDICINE_ID,MANUFACTURER_ID, MEDICINE_NAME)
VALUES ('6','2', 'DEPLETCV');
INSERT INTO MEDICINES (MEDICINE_ID,MANUFACTURER_ID, MEDICINE_NAME)
VALUES ('7','3', 'ROZAVALFLS');
INSERT INTO MEDICINES (MEDICINE_ID,MANUFACTURER_ID, MEDICINE_NAME)
```

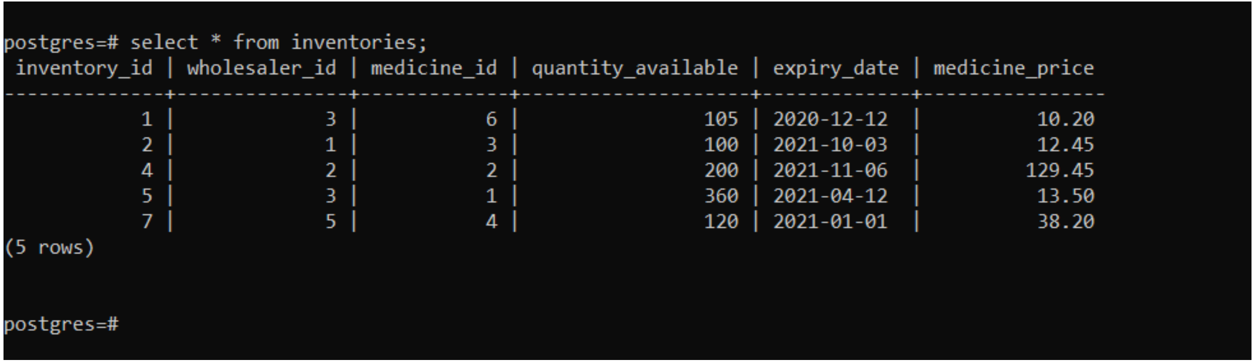
VALUES ('8','3', 'FABIFLU');



```
postgres=# select * from medicines;
 medicine_id | manufacturer_id | medicine_name
-------------+-----------------+---------------
           1 |               1 | SOFRAMYCIN
           2 |               1 | PARACETAMLE
           3 |               1 | BURNOL
           4 |               2 | NAPROXEN
           5 |               2 | ASPIRIN
           6 |               2 | DEPLETCV
           7 |               3 | ROZAVALFLS
           8 |               3 | FABIFLU
(8 rows)
```

Table 6: INVENTORIES Table

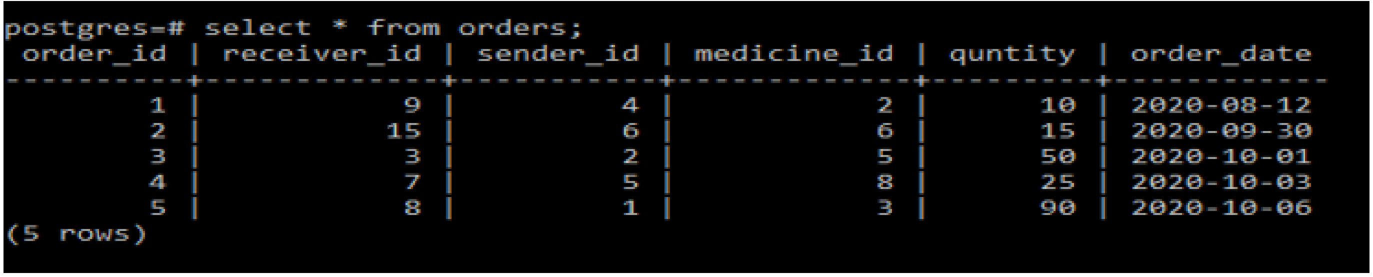
INSERT INTO INVENTORIES (INVENTORY_ID,WHOLESALER_ID, MEDICINE_ID, QUANTITY_AVAILABLE, EXPIRY_DATE, MEDICINE_PRICE) VALUES (1,3, 6, 105, '12-12-20', 10.2);
INSERT INTO INVENTORIES (INVENTORY_ID,WHOLESALER_ID, MEDICINE_ID, QUANTITY_AVAILABLE, EXPIRY_DATE, MEDICINE_PRICE) VALUES (2,1, 3, 100, '10-03-21', 12.45);
INSERT INTO INVENTORIES (INVENTORY_ID,WHOLESALER_ID, MEDICINE_ID, QUANTITY_AVAILABLE, EXPIRY_DATE, MEDICINE_PRICE) VALUES (3,2, 7, 75, '22-04-22', 15.40);
INSERT INTO INVENTORIES (INVENTORY_ID,WHOLESALER_ID, MEDICINE_ID, QUANTITY_AVAILABLE, EXPIRY_DATE, MEDICINE_PRICE) VALUES (4,2, 2, 200, '11-06-21', 129.45);
INSERT INTO INVENTORIES (INVENTORY_ID,WHOLESALER_ID, MEDICINE_ID, QUANTITY_AVAILABLE, EXPIRY_DATE, MEDICINE_PRICE) VALUES (5,3, 1, 360, '04-12-21', 13.5);
INSERT INTO INVENTORIES (INVENTORY_ID,WHOLESALER_ID, MEDICINE_ID, QUANTITY_AVAILABLE, EXPIRY_DATE, MEDICINE_PRICE) VALUES (6,4, 8, 112, '16-10-21', 58.6);
INSERT INTO INVENTORIES (INVENTORY_ID,WHOLESALER_ID, MEDICINE_ID, QUANTITY_AVAILABLE, EXPIRY_DATE, MEDICINE_PRICE) VALUES (7,5, 4, 120, '01-01-21', 38.20);
INSERT INTO INVENTORIES (INVENTORY_ID,WHOLESALER_ID,MEDICINE_ID, QUANTITY_AVAILABLE, EXPIRY_DATE, MEDICINE_PRICE) VALUES (8,5, 5, 450, '15-06-23', 5.62);



```
postgres=# select * from inventories;
 inventory_id | wholesaler_id | medicine_id | quantity_available | expiry_date | medicine_price
--------------+---------------+-------------+--------------------+-------------+----------------
            1 |             3 |           6 |                105 | 2020-12-12  |          10.20
            2 |             1 |           3 |                100 | 2021-10-03  |          12.45
            4 |             2 |           2 |                200 | 2021-11-06  |         129.45
            5 |             3 |           1 |                360 | 2021-04-12  |          13.50
            7 |             5 |           4 |                120 | 2021-01-01  |          38.20
(5 rows)


postgres=#
```

Table 7: Orders Table

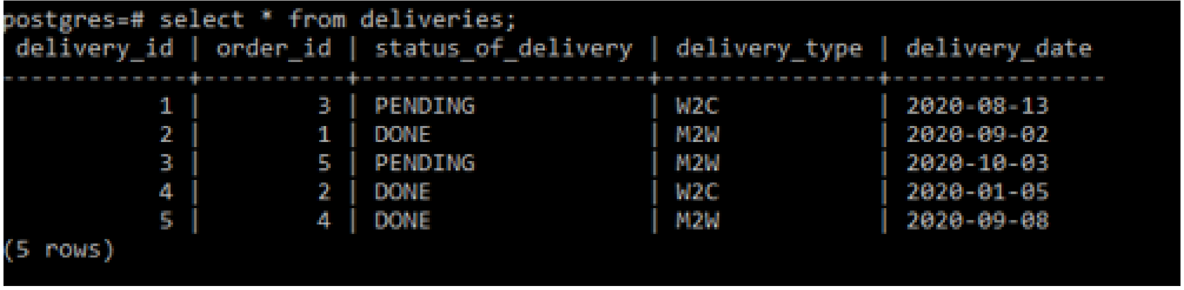
INSERT INTO ORDERS (ORDER_ID,RECEIVER_ID, SENDER_ID, MEDICINE_ID, QUNTITY, ORDER_DATE) VALUES (1,9, 4, 2, 10, '12-08-20');
INSERT INTO ORDERS (ORDER_ID,RECEIVER_ID, SENDER_ID, MEDICINE_ID, QUNTITY, ORDER_DATE) VALUES (2,15, 6, 6, 15, '30-09-20');
INSERT INTO ORDERS (ORDER_ID,RECEIVER_ID, SENDER_ID, MEDICINE_ID, QUNTITY, ORDER_DATE) VALUES (3,3, 2, 5, 50, '01-10-20');
INSERT INTO ORDERS (ORDER_ID,RECEIVER_ID, SENDER_ID, MEDICINE_ID, QUNTITY, ORDER_DATE) VALUES (4,7, 5, 8, 25, '03-10-20');
INSERT INTO ORDERS (ORDER_ID,RECEIVER_ID, SENDER_ID, MEDICINE_ID, QUNTITY, ORDER_DATE) VALUES (5,8, 1, 3, 90, '06-10-20');



```
postgres=# select * from orders;
 order_id | receiver_id | sender_id | medicine_id | quntity | order_date
----------+-------------+-----------+-------------+---------+------------
        1 |           9 |         4 |           2 |      10 | 2020-08-12
        2 |          15 |         6 |           6 |      15 | 2020-09-30
        3 |           3 |         2 |           5 |      50 | 2020-10-01
        4 |           7 |         5 |           8 |      25 | 2020-10-03
        5 |           8 |         1 |           3 |      90 | 2020-10-06
(5 rows)
```

Table 8: Deliveries Table

INSERT INTO DELIVERIES (DELIVERY_ID,ORDER_ID, STATUS_OF_DELIVERY, DELIVERY_TYPE, DELIVERY_DATE) VALUES (1,3, 'PENDING', 'W2C', '13-08-20');
INSERT INTO DELIVERIES (DELIVERY_ID,ORDER_ID, STATUS_OF_DELIVERY, DELIVERY_TYPE, DELIVERY_DATE) VALUES (2,1, 'DONE', 'M2W', '02-09-20');
INSERT INTO DELIVERIES (DELIVERY_ID,ORDER_ID, STATUS_OF_DELIVERY, DELIVERY_TYPE, DELIVERY_DATE) VALUES (3,5, 'PENDING', 'M2W', '03-10-20');
INSERT INTO DELIVERIES (DELIVERY_ID,ORDER_ID, STATUS_OF_DELIVERY, DELIVERY_TYPE, DELIVERY_DATE) VALUES (4,2, 'DONE', 'W2C', '05-01-20');
INSERT INTO DELIVERIES (DELIVERY_ID,ORDER_ID, STATUS_OF_DELIVERY, DELIVERY_TYPE, DELIVERY_DATE) VALUES (5,4, 'DONE', 'M2W', '08-09-20');

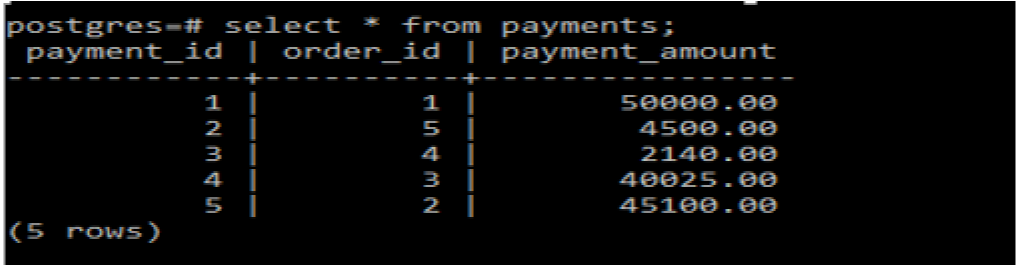


```
postgres=# select * from deliveries;
 delivery_id | order_id | status_of_delivery | delivery_type | delivery_date
-------------+----------+--------------------+---------------+---------------
           1 |        3 | PENDING            | W2C           | 2020-08-13
           2 |        1 | DONE               | M2W           | 2020-09-02
           3 |        5 | PENDING            | M2W           | 2020-10-03
           4 |        2 | DONE               | W2C           | 2020-01-05
           5 |        4 | DONE               | M2W           | 2020-09-08
(5 rows)
```

Table 9: Payments Table

INSERT INTO PAYMENTS(PAYMENT_ID,ORDER_ID, PAYMENT_AMOUNT) VALUES (1,1,

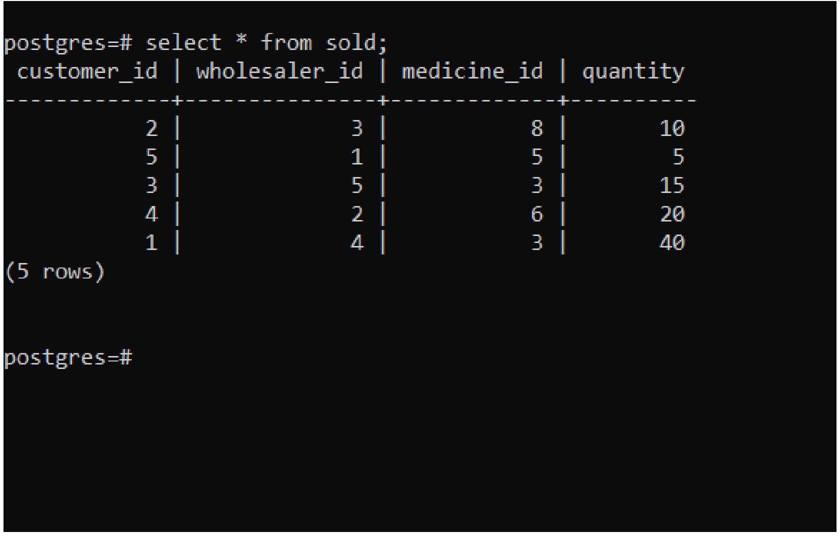
50000);
INSERT INTO PAYMENTS(PAYMENT_ID,ORDER_ID, PAYMENT_AMOUNT) VALUES (2,5, 4500);
INSERT INTO PAYMENTS(PAYMENT_ID,ORDER_ID, PAYMENT_AMOUNT) VALUES (3,4, 2140);
INSERT INTO PAYMENTS(PAYMENT_ID,ORDER_ID, PAYMENT_AMOUNT) VALUES (4,3, 40025);
INSERT INTO PAYMENTS(PAYMENT_ID,ORDER_ID, PAYMENT_AMOUNT) VALUES (5,2, 45100);



```
postgres=# select * from payments;
 payment_id | order_id | payment_amount
------------+----------+----------------
          1 |        1 |       50000.00
          2 |        5 |        4500.00
          3 |        4 |        2140.00
          4 |        3 |       40025.00
          5 |        2 |       45100.00
(5 rows)
```

Table 10: Sold Table

INSERT INTO SOLD (CUSTOMER_ID, WHOLESALER_ID, MEDICINE_ID, QUANTITY) VALUES (2, 3, 8, 10);
INSERT INTO SOLD (CUSTOMER_ID, WHOLESALER_ID, MEDICINE_ID, QUANTITY) VALUES (5, 1, 5, 5);
INSERT INTO SOLD (CUSTOMER_ID, WHOLESALER_ID, MEDICINE_ID, QUANTITY) VALUES (3, 5, 3, 15);
INSERT INTO SOLD (CUSTOMER_ID, WHOLESALER_ID, MEDICINE_ID, QUANTITY) VALUES (4, 2, 6, 20);
INSERT INTO SOLD (CUSTOMER_ID, WHOLESALER_ID, MEDICINE_ID, QUANTITY) VALUES (1, 4, 3, 40);

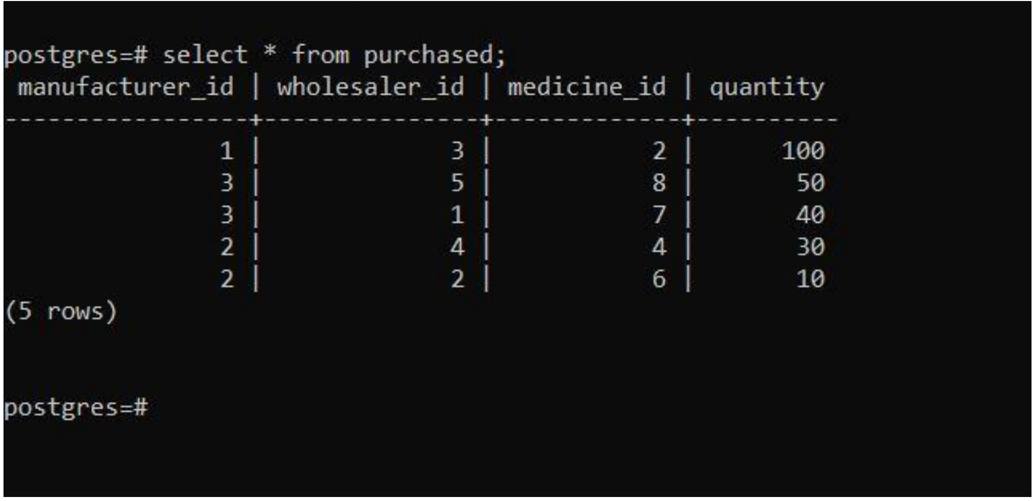


```
postgres=# select * from sold;
 customer_id | wholesaler_id | medicine_id | quantity
-------------+---------------+-------------+----------
           2 |             3 |           8 |       10
           5 |             1 |           5 |        5
           3 |             5 |           3 |       15
           4 |             2 |           6 |       20
           1 |             4 |           3 |       40
(5 rows)


postgres=#
```

Table 11: Purchased Table

INSERT INTO PURCHASED (MANUFACTURER_ID, WHOLESALER_ID, MEDICINE_ID, QUANTITY) VALUES (1, 3, 2, 100);

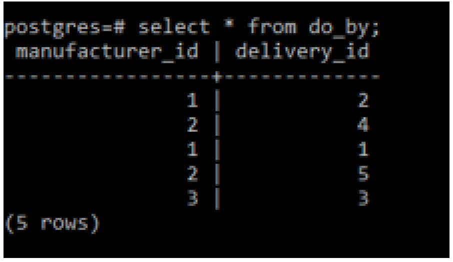
INSERT INTO PURCHASED (MANUFACTURER_ID, WHOLESALER_ID, MEDICINE_ID, QUANTITY) VALUES (3, 5, 8, 50);
INSERT INTO PURCHASED (MANUFACTURER_ID, WHOLESALER_ID, MEDICINE_ID, QUANTITY) VALUES (3, 1, 7, 40);
INSERT INTO PURCHASED (MANUFACTURER_ID, WHOLESALER_ID, MEDICINE_ID, QUANTITY) VALUES (2, 4, 4, 30);
INSERT INTO PURCHASED (MANUFACTURER_ID, WHOLESALER_ID, MEDICINE_ID, QUANTITY) VALUES (2, 2, 6, 10);



```
postgres=# select * from purchased;
 manufacturer_id | wholesaler_id | medicine_id | quantity
-----------------+---------------+-------------+----------
               1 |             3 |           2 |      100
               3 |             5 |           8 |       50
               3 |             1 |           7 |       40
               2 |             4 |           4 |       30
               2 |             2 |           6 |       10
(5 rows)


postgres=#
```

Table 12: Do Table

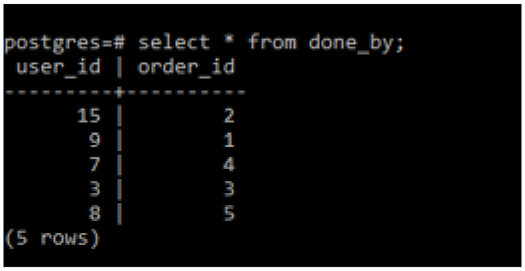
INSERT INTO DO_BY(MANUFACTURER_ID, DELIVERY_ID) VALUES (1, 2);
INSERT INTO DO_BY(MANUFACTURER_ID, DELIVERY_ID) VALUES (2, 4);
INSERT INTO DO_BY(MANUFACTURER_ID, DELIVERY_ID) VALUES (1, 1);
INSERT INTO DO_BY(MANUFACTURER_ID, DELIVERY_ID) VALUES (2, 5);
INSERT INTO DO_BY(MANUFACTURER_ID, DELIVERY_ID) VALUES (3, 3);



```
postgres=# select * from do_by;
 manufacturer_id | delivery_id
-----------------+-------------
               1 |           2
               2 |           4
               1 |           1
               2 |           5
               3 |           3
(5 rows)
```

Table 13: Done By Table

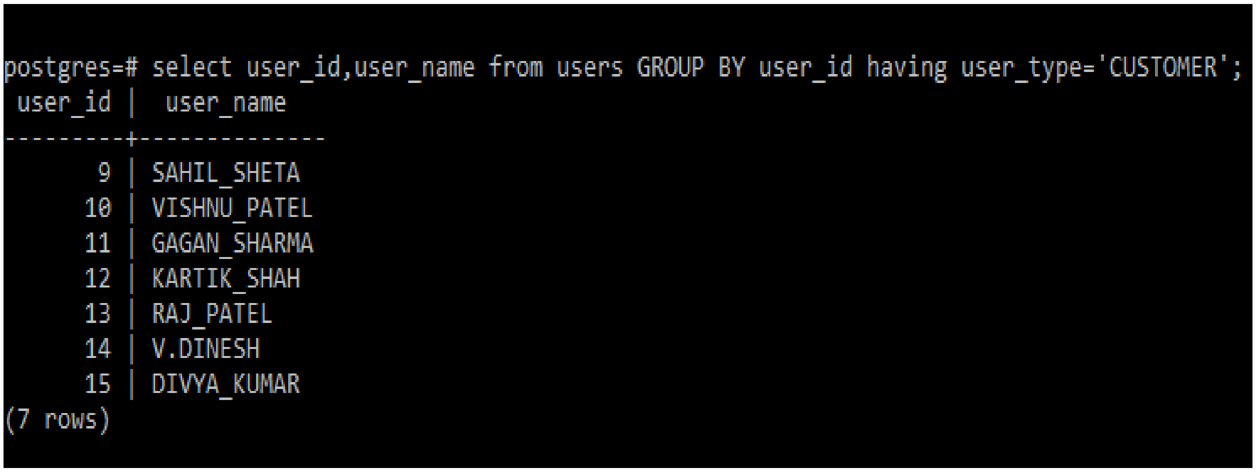
INSERT INTO DONE_BY(USER_ID, ORDER_ID) VALUES (15, 2);
INSERT INTO DONE_BY(USER_ID, ORDER_ID) VALUES (9, 1);
INSERT INTO DONE_BY(USER_ID, ORDER_ID) VALUES (7, 4);
INSERT INTO DONE_BY(USER_ID, ORDER_ID) VALUES (3, 3);
INSERT INTO DONE_BY(USER_ID, ORDER_ID) VALUES (8, 5);



```
postgres=# select * from done_by;
 user_id | order_id
---------+----------
      15 |        2
       9 |        1
       7 |        4
       3 |        3
       8 |        5
(5 rows)
```

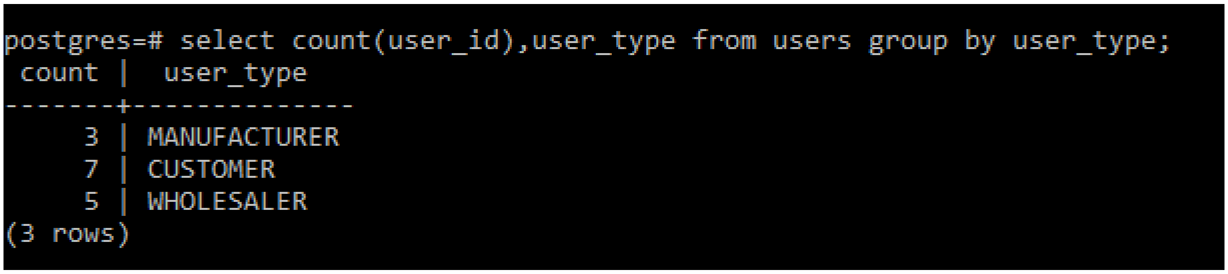
5.3 Queries:

1. **Fetch the user details where user type is customer:**



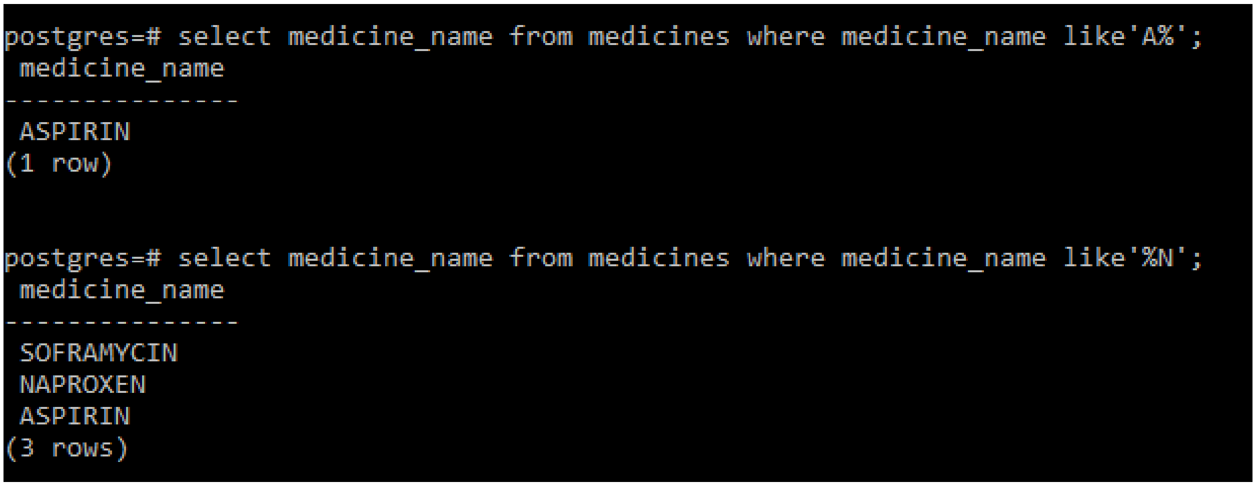
```
postgres=# select user_id,user_name from users GROUP BY user_id having user_type='CUSTOMER';
 user_id |  user_name
---------+-------------
       9 | SAHIL_SHETA
      10 | VISHNU_PATEL
      11 | GAGAN_SHARMA
      12 | KARTIK_SHAH
      13 | RAJ_PATEL
      14 | V.DINESH
      15 | DIVYA_KUMAR
(7 rows)
```

2. **Find the number of users based on user types:**



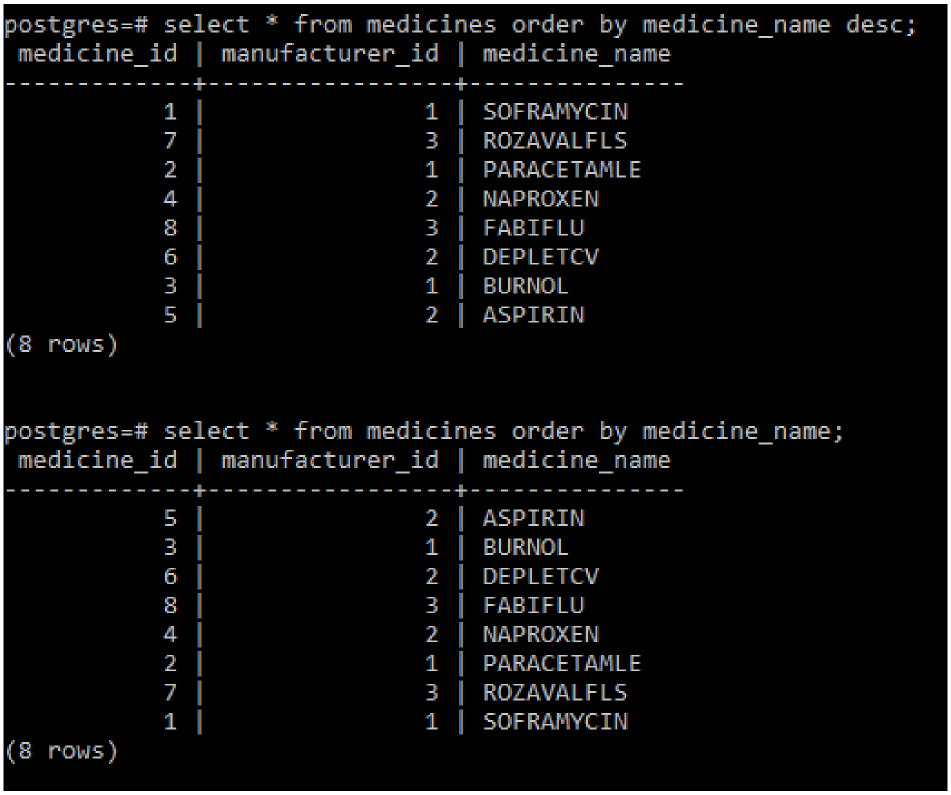
```
postgres=# select count(user_id),user_type from users group by user_type;
 count |  user_type
-------+--------------
     3 | MANUFACTURER
     7 | CUSTOMER
     5 | WHOLESALER
(3 rows)
```

3. **Fetch the medicine names :**



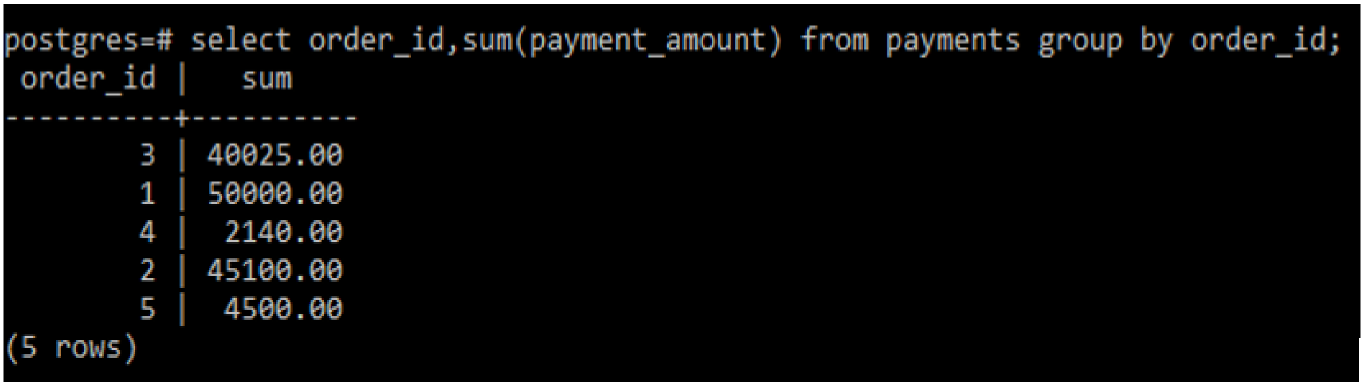
```
postgres=# select medicine_name from medicines where medicine_name like'A%';
 medicine_name
---------------
 ASPIRIN
(1 row)


postgres=# select medicine_name from medicines where medicine_name like'%N';
 medicine_name
---------------
 SOFRAMYCIN
 NAPROXEN
 ASPIRIN
(3 rows)
```

4. Medicine name in ascending & descending order:



```
postgres=# select * from medicines order by medicine_name desc;
 medicine_id | manufacturer_id | medicine_name
-------------+-----------------+---------------
           1 |               1 | SOFRAMYCIN
           7 |               3 | ROZAVALFLS
           2 |               1 | PARACETAMLE
           4 |               2 | NAPROXEN
           8 |               3 | FABIFLU
           6 |               2 | DEPLETCV
           3 |               1 | BURNOL
           5 |               2 | ASPIRIN
(8 rows)


postgres=# select * from medicines order by medicine_name;
 medicine_id | manufacturer_id | medicine_name
-------------+-----------------+---------------
           5 |               2 | ASPIRIN
           3 |               1 | BURNOL
           6 |               2 | DEPLETCV
           8 |               3 | FABIFLU
           4 |               2 | NAPROXEN
           2 |               1 | PARACETAMLE
           7 |               3 | ROZAVALFLS
           1 |               1 | SOFRAMYCIN
(8 rows)
```

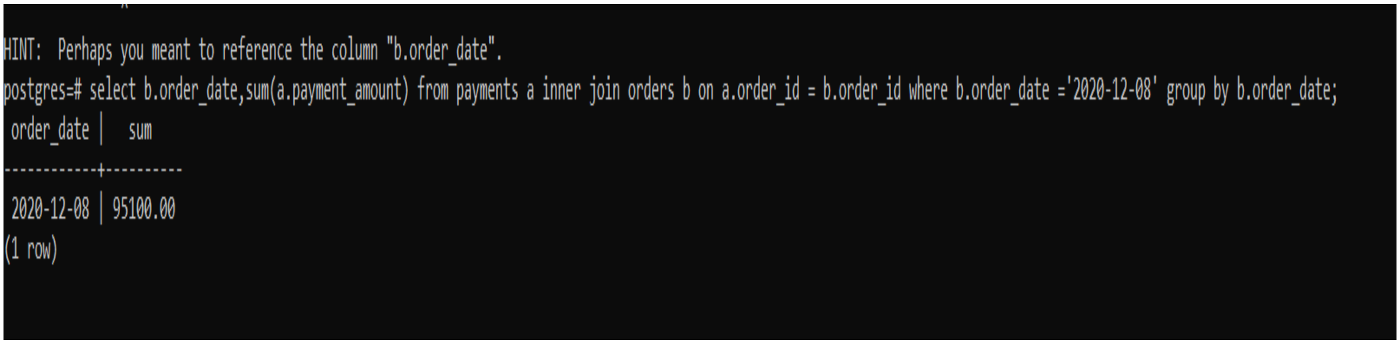
5. Find the total payment of order id:



```
postgres=# select order_id,sum(payment_amount) from payments group by order_id;
 order_id |   sum
----------+----------
        3 | 40025.00
        1 | 50000.00
        4 |  2140.00
        2 | 45100.00
        5 |  4500.00
(5 rows)
```

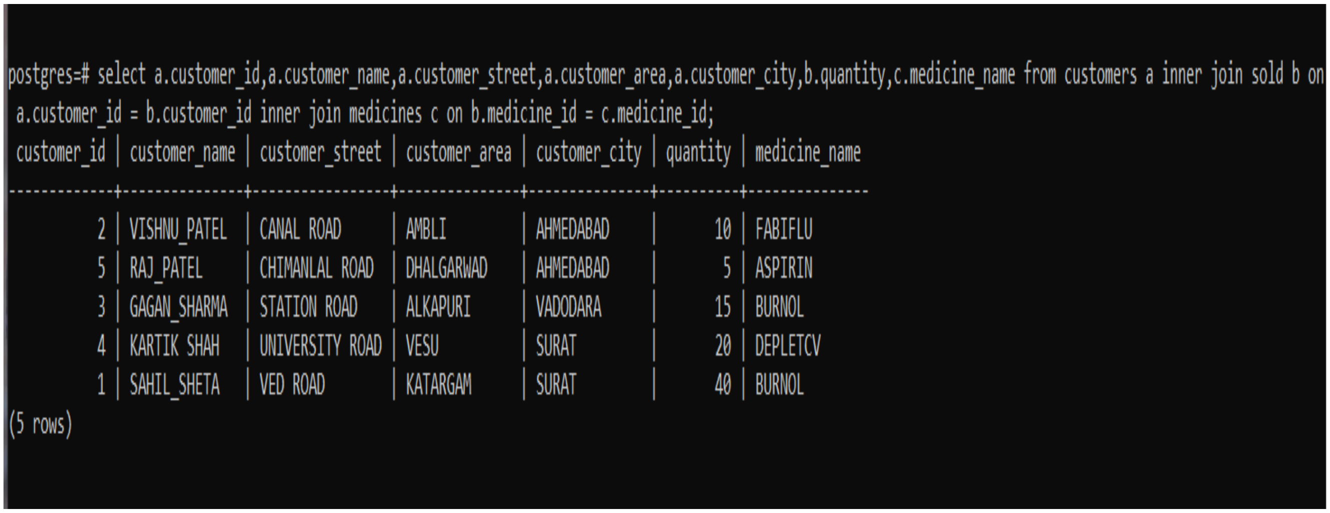
Joins & sub queries:

1. **Total sale of the particular day:**



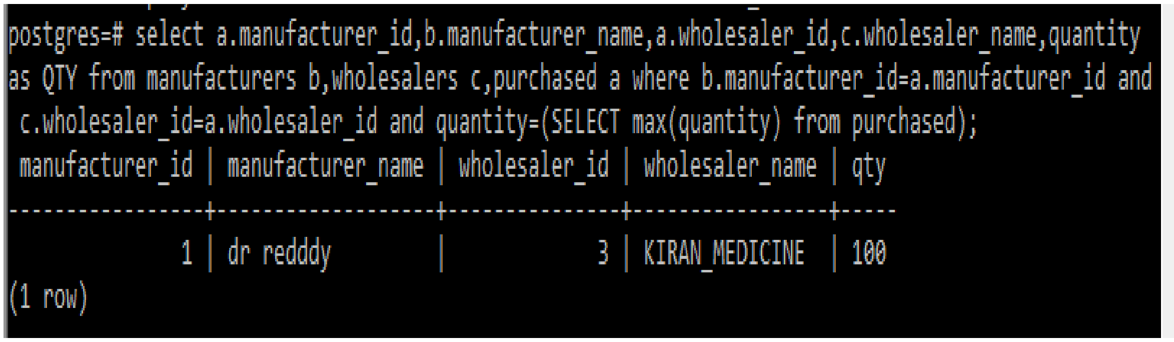
```
HINT:  Perhaps you meant to reference the column "b.order_date".
postgres=# select b.order_date,sum(a.payment_amount) from payments a inner join orders b on a.order_id = b.order_id where b.order_date ='2020-12-08' group by b.order_date;
 order_date |   sum
------------+----------
 2020-12-08 | 95100.00
(1 row)
```

2. **Details of customer to provide them home delivery of medication:**



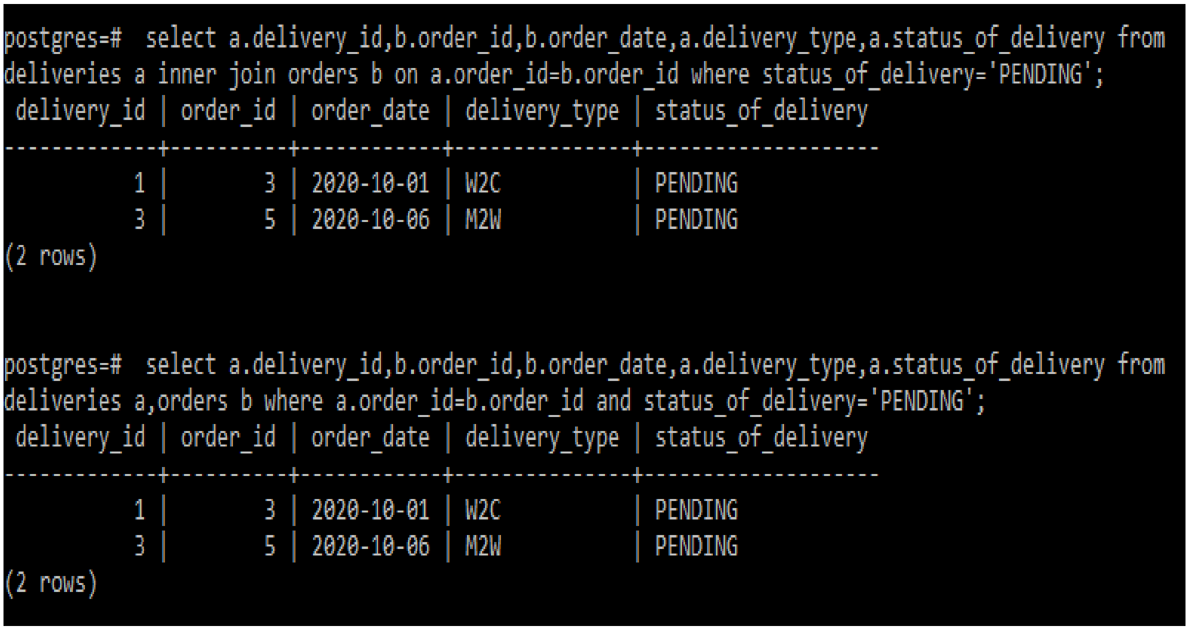
```
postgres=# select a.customer_id,a.customer_name,a.customer_street,a.customer_area,a.customer_city,b.quantity,c.medicine_name from customers a inner join sold b on
 a.customer_id = b.customer_id inner join medicines c on b.medicine_id = c.medicine_id;
 customer_id | customer_name | customer_street | customer_area | customer_city | quantity | medicine_name
-------------+---------------+-----------------+---------------+---------------+----------+---------------
           2 | VISHNU_PATEL  | CANAL ROAD      | AMBLI         | AHMEDABAD     |       10 | FABIFLU
           5 | RAJ_PATEL     | CHIMANLAL ROAD  | DHALGARWAD    | AHMEDABAD     |        5 | ASPIRIN
           3 | GAGAN_SHARMA  | STATION ROAD    | ALKAPURI      | VADODARA      |       15 | BURNOL
           4 | KARTIK SHAH   | UNIVERSITY ROAD | VESU          | SURAT         |       20 | DEPLETCV
           1 | SAHIL_SHETA   | VED ROAD        | KATARGAM      | SURAT         |       40 | BURNOL
(5 rows)
```

3. **Find the highest quantity purchased by wholesaler from manufacture:**



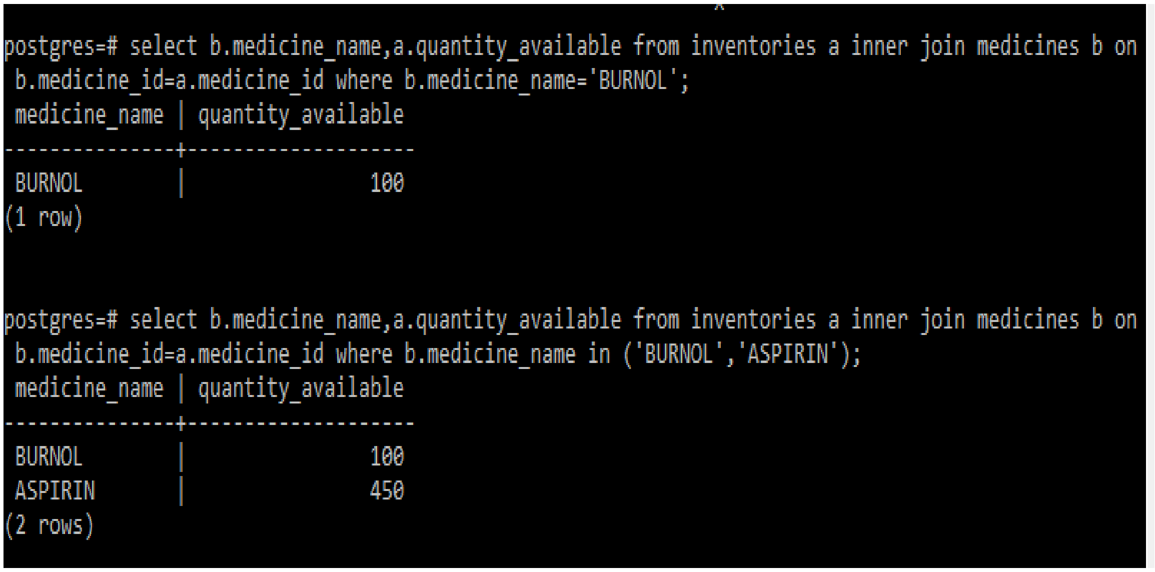
```
postgres=# select a.manufacturer_id,b.manufacturer_name,a.wholesaler_id,c.wholesaler_name,quantity
as QTY from manufacturers b,wholesalers c,purchased a where b.manufacturer_id=a.manufacturer_id and
 c.wholesaler_id=a.wholesaler_id and quantity=(SELECT max(quantity) from purchased);
 manufacturer_id | manufacturer_name | wholesaler_id | wholesaler_name | qty
-----------------+-------------------+---------------+-----------------+-----
               1 | dr redddy         |             3 | KIRAN_MEDICINE  | 100
(1 row)
```

4. **Display the orders where status of delivery is pending:**



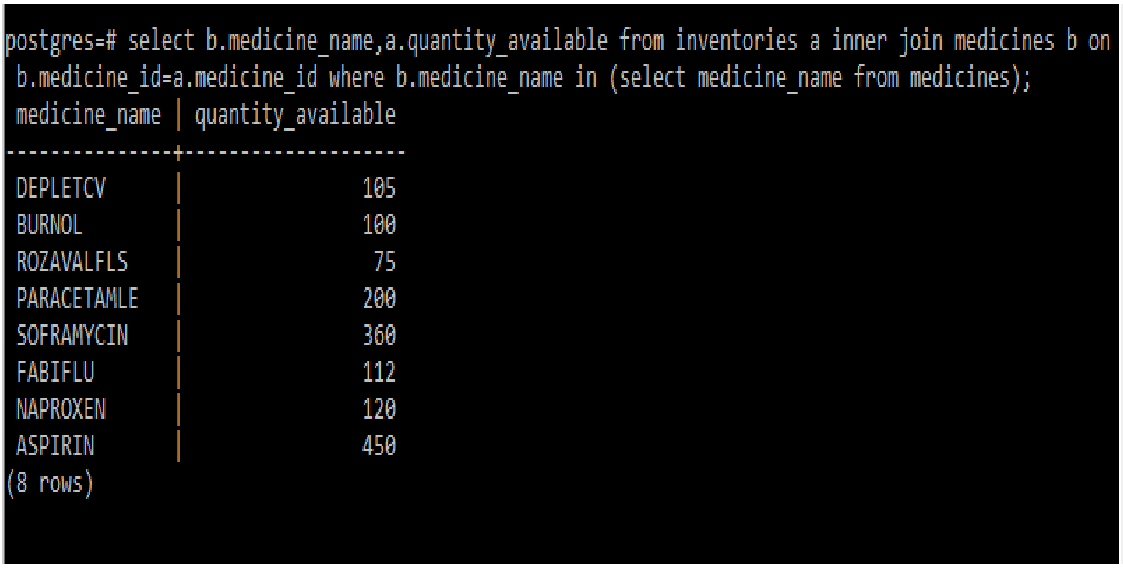
```
postgres=#  select a.delivery_id,b.order_id,b.order_date,a.delivery_type,a.status_of_delivery from
deliveries a inner join orders b on a.order_id=b.order_id where status_of_delivery='PENDING';
 delivery_id | order_id | order_date | delivery_type | status_of_delivery
-------------+----------+------------+---------------+--------------------
           1 |        3 | 2020-10-01 | W2C           | PENDING
           3 |        5 | 2020-10-06 | M2W           | PENDING
(2 rows)


postgres=#  select a.delivery_id,b.order_id,b.order_date,a.delivery_type,a.status_of_delivery from
deliveries a,orders b where a.order_id=b.order_id and status_of_delivery='PENDING';
 delivery_id | order_id | order_date | delivery_type | status_of_delivery
-------------+----------+------------+---------------+--------------------
           1 |        3 | 2020-10-01 | W2C           | PENDING
           3 |        5 | 2020-10-06 | M2W           | PENDING
(2 rows)
```

5. **Display the quantity of particular medicine:**



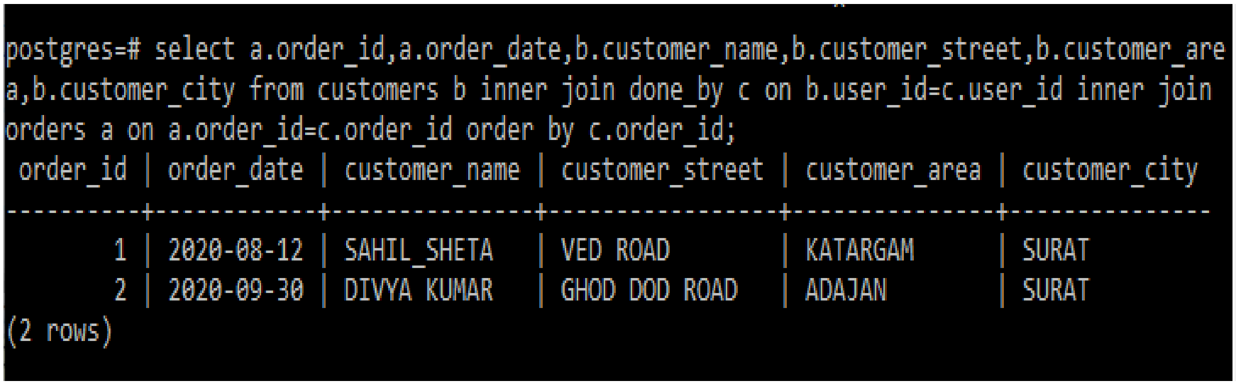
```
postgres=# select b.medicine_name,a.quantity_available from inventories a inner join medicines b on
 b.medicine_id=a.medicine_id where b.medicine_name='BURNOL';
 medicine_name | quantity_available
---------------+--------------------
 BURNOL        |                100
(1 row)


postgres=# select b.medicine_name,a.quantity_available from inventories a inner join medicines b on
 b.medicine_id=a.medicine_id where b.medicine_name in ('BURNOL','ASPIRIN');
 medicine_name | quantity_available
---------------+--------------------
 BURNOL        |                100
 ASPIRIN       |                450
(2 rows)
```

6. **Display the quantity of all medicine:**



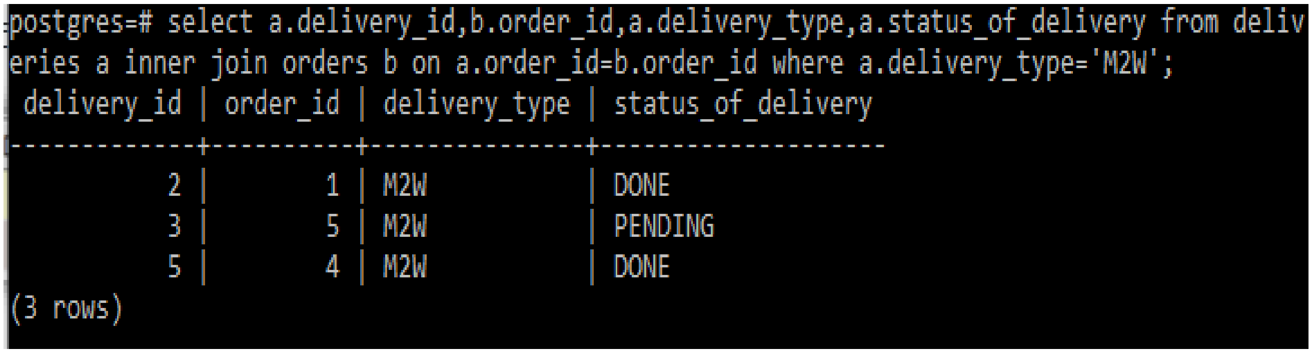
```
postgres=# select b.medicine_name,a.quantity_available from inventories a inner join medicines b on
 b.medicine_id=a.medicine_id where b.medicine_name in (select medicine_name from medicines);
 medicine_name | quantity_available
---------------+--------------------
 DEPLETCV      |                105
 BURNOL        |                100
 ROZAVALFLS    |                 75
 PARACETAMLE   |                200
 SOFRAMYCIN    |                360
 FABIFLU       |                112
 NAPROXEN      |                120
 ASPIRIN       |                450
(8 rows)
```

7. **Display the user details where they placed the order & user type is customer :**



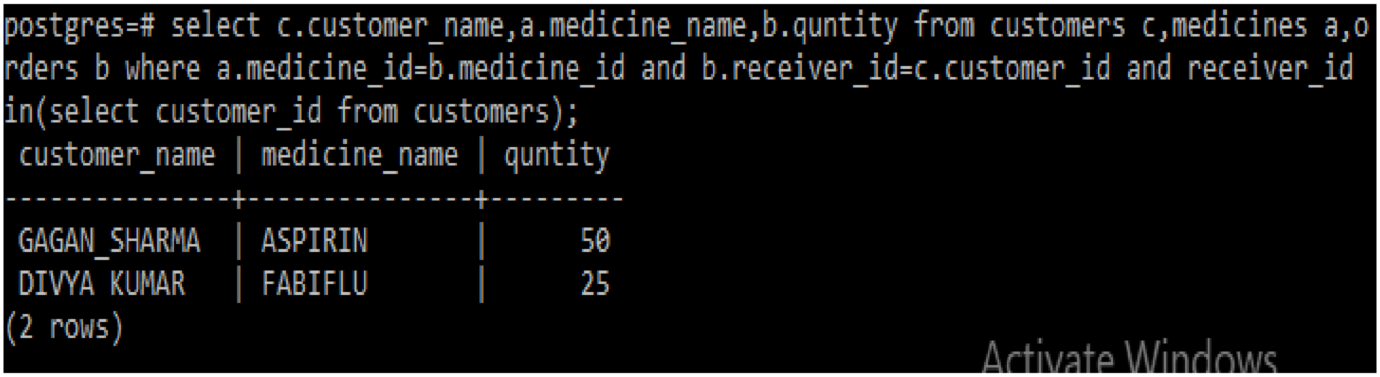
```
postgres=# select a.order_id,a.order_date,b.customer_name,b.customer_street,b.customer_are
a,b.customer_city from customers b inner join done_by c on b.user_id=c.user_id inner join
orders a on a.order_id=c.order_id order by c.order_id;
 order_id | order_date | customer_name | customer_street | customer_area | customer_city
----------+------------+---------------+-----------------+---------------+---------------
        1 | 2020-08-12 | SAHIL_SHETA   | VED ROAD        | KATARGAM      | SURAT
        2 | 2020-09-30 | DIVYA KUMAR   | GHOD DOD ROAD   | ADAJAN        | SURAT
(2 rows)
```

8. **Display the delivery detail where delivery is done by manufacturer to wholesaler:**



```
postgres=# select a.delivery_id,b.order_id,a.delivery_type,a.status_of_delivery from deliv
eries a inner join orders b on a.order_id=b.order_id where a.delivery_type='M2W';
 delivery_id | order_id | delivery_type | status_of_delivery
-------------+----------+---------------+--------------------
           2 |        1 | M2W           | DONE
           3 |        5 | M2W           | PENDING
           5 |        4 | M2W           | DONE
(3 rows)
```

9. Display the which medicine & how much quantity ordered by which customer:

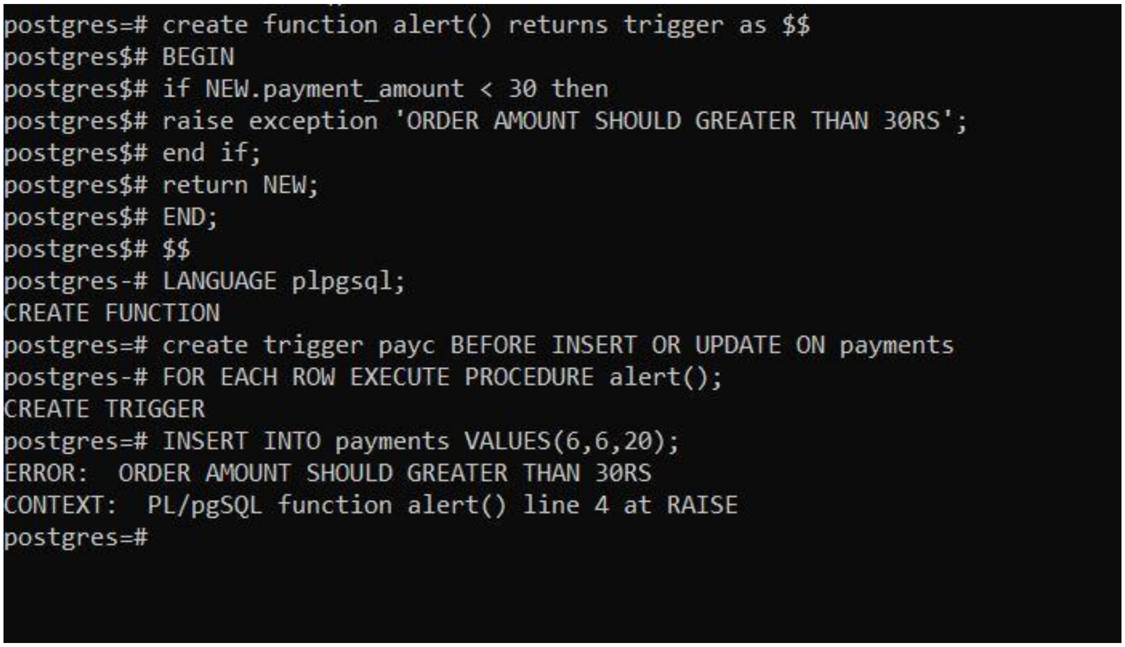


```
postgres=# select c.customer_name,a.medicine_name,b.quntity from customers c,medicines a,o
rders b where a.medicine_id=b.medicine_id and b.receiver_id=c.customer_id and receiver_id
in(select customer_id from customers);
 customer_name | medicine_name | quntity
---------------+---------------+---------
 GAGAN_SHARMA  | ASPIRIN       |      50
 DIVYA KUMAR   | FABIFLU       |      25
(2 rows)
```

PL/SQL BLOCKS:

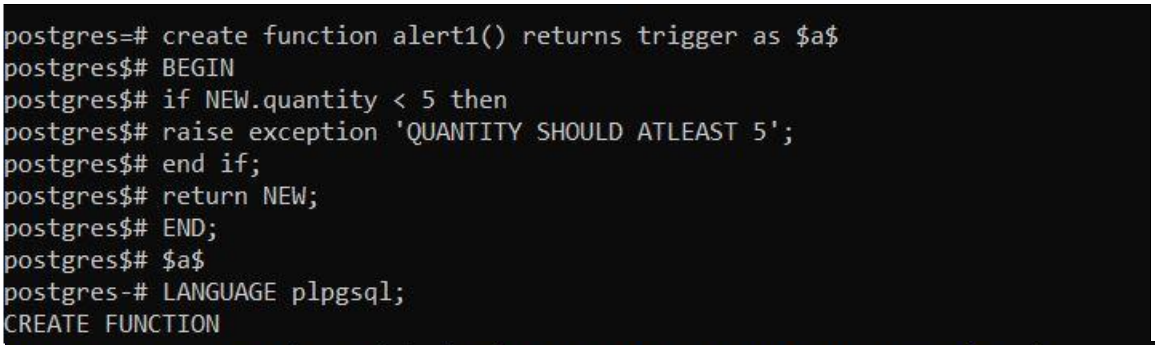
- **TRIGGER & FUNCTIONS:**

1. TRIGGER WITH FUNCTION THAT IF ORDER AMOUNT IS LESS THAN 30RS THEN WILL SHOW ERROR:-

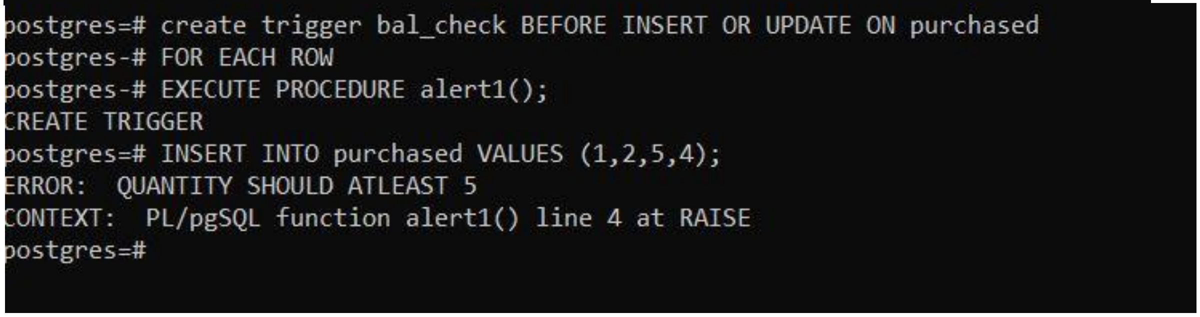


```
postgres=# create function alert() returns trigger as $$
postgres$# BEGIN
postgres$# if NEW.payment_amount < 30 then
postgres$# raise exception 'ORDER AMOUNT SHOULD GREATER THAN 30RS';
postgres$# end if;
postgres$# return NEW;
postgres$# END;
postgres$# $$
postgres-# LANGUAGE plpgsql;
CREATE FUNCTION
postgres=# create trigger payc BEFORE INSERT OR UPDATE ON payments
postgres-# FOR EACH ROW EXECUTE PROCEDURE alert();
CREATE TRIGGER
postgres=# INSERT INTO payments VALUES(6,6,20);
ERROR:  ORDER AMOUNT SHOULD GREATER THAN 30RS
CONTEXT:  PL/pgSQL function alert() line 4 at RAISE
postgres=#
```

2. TRIGGER WITH FUNCTION THAT IF WE ORDER QUANTITY LESS THAN 5 THEN WILL SHOW ALERT:-

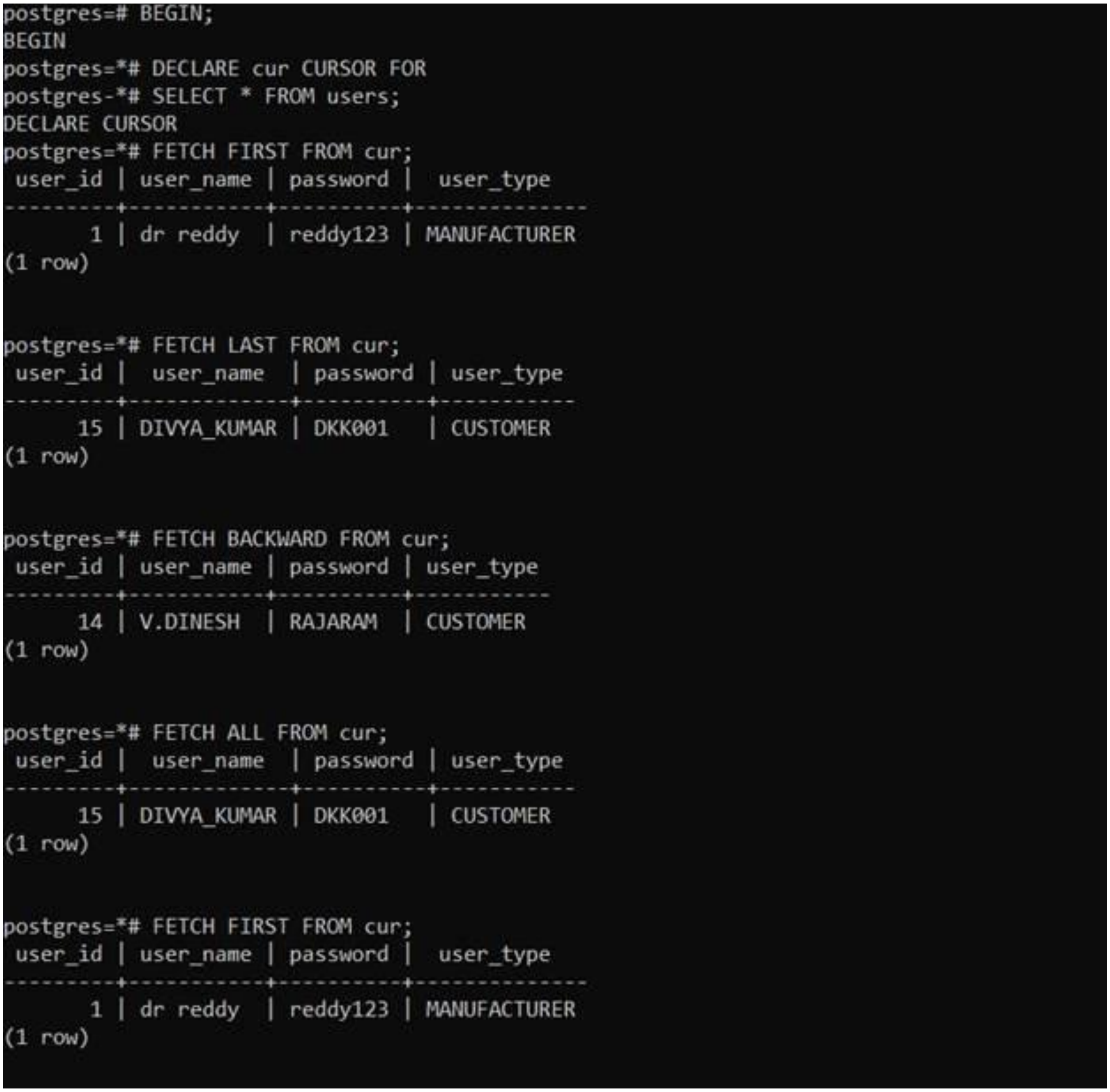


```
postgres=# create function alert1() returns trigger as $a$
postgres$# BEGIN
postgres$# if NEW.quantity < 5 then
postgres$# raise exception 'QUANTITY SHOULD ATLEAST 5';
postgres$# end if;
postgres$# return NEW;
postgres$# END;
postgres$# $a$
postgres-# LANGUAGE plpgsql;
CREATE FUNCTION
```

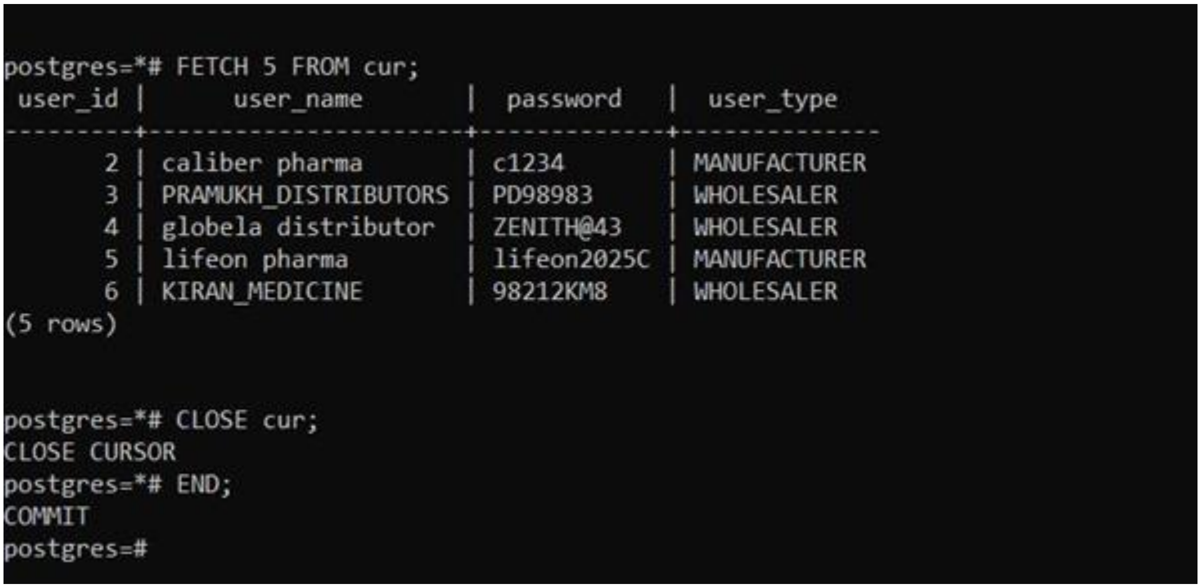


```
postgres=# create trigger bal_check BEFORE INSERT OR UPDATE ON purchased
postgres-# FOR EACH ROW
postgres-# EXECUTE PROCEDURE alert1();
CREATE TRIGGER
postgres=# INSERT INTO purchased VALUES (1,2,5,4);
ERROR:  QUANTITY SHOULD ATLEAST 5
CONTEXT:  PL/pgSQL function alert1() line 4 at RAISE
postgres=#
```

CURSOR:



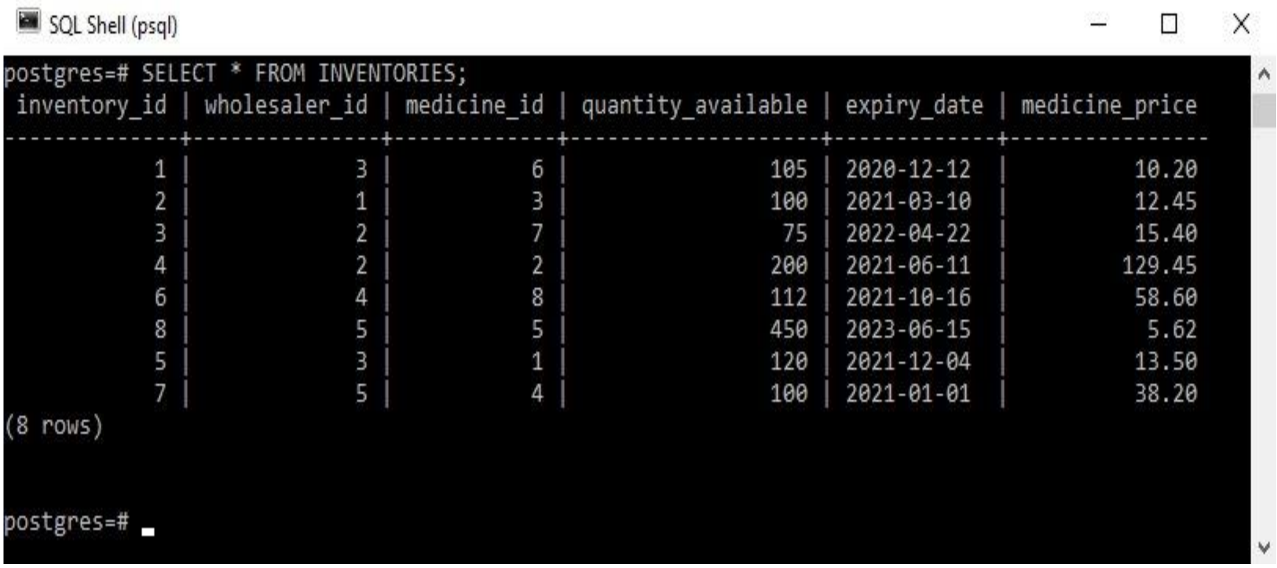
```
postgres=# BEGIN;
BEGIN
postgres=*# DECLARE cur CURSOR FOR
postgres-*# SELECT * FROM users;
DECLARE CURSOR
postgres=*# FETCH FIRST FROM cur;
 user_id | user_name | password |  user_type
---------+-----------+----------+--------------
       1 | dr reddy  | reddy123 | MANUFACTURER
(1 row)


postgres=*# FETCH LAST FROM cur;
 user_id |  user_name  | password | user_type
---------+-------------+----------+-----------
      15 | DIVYA_KUMAR | DKK001   | CUSTOMER
(1 row)


postgres=*# FETCH BACKWARD FROM cur;
 user_id | user_name | password | user_type
---------+-----------+----------+-----------
      14 | V.DINESH  | RAJARAM  | CUSTOMER
(1 row)


postgres=*# FETCH ALL FROM cur;
 user_id |  user_name  | password | user_type
---------+-------------+----------+-----------
      15 | DIVYA_KUMAR | DKK001   | CUSTOMER
(1 row)


postgres=*# FETCH FIRST FROM cur;
 user_id | user_name | password |  user_type
---------+-----------+----------+--------------
       1 | dr reddy  | reddy123 | MANUFACTURER
(1 row)
```



```
postgres=*# FETCH 5 FROM cur;
 user_id |      user_name       |  password   |  user_type
---------+----------------------+-------------+-------------
       2 | caliber pharma       | c1234       | MANUFACTURER
       3 | PRAMUKH_DISTRIBUTORS | PD98983     | WHOLESALER
       4 | globela distributor  | ZENITH@43   | WHOLESALER
       5 | lifeon pharma        | lifeon2025C | MANUFACTURER
       6 | KIRAN_MEDICINE       | 98212KM8    | WHOLESALER
(5 rows)


postgres=*# CLOSE cur;
CLOSE CURSOR
postgres=*# END;
COMMIT
postgres=#
```

- **OTHER:**

1. **Function trigger if quantity in inventories drops below certain amount then inserted into new table imp :-**

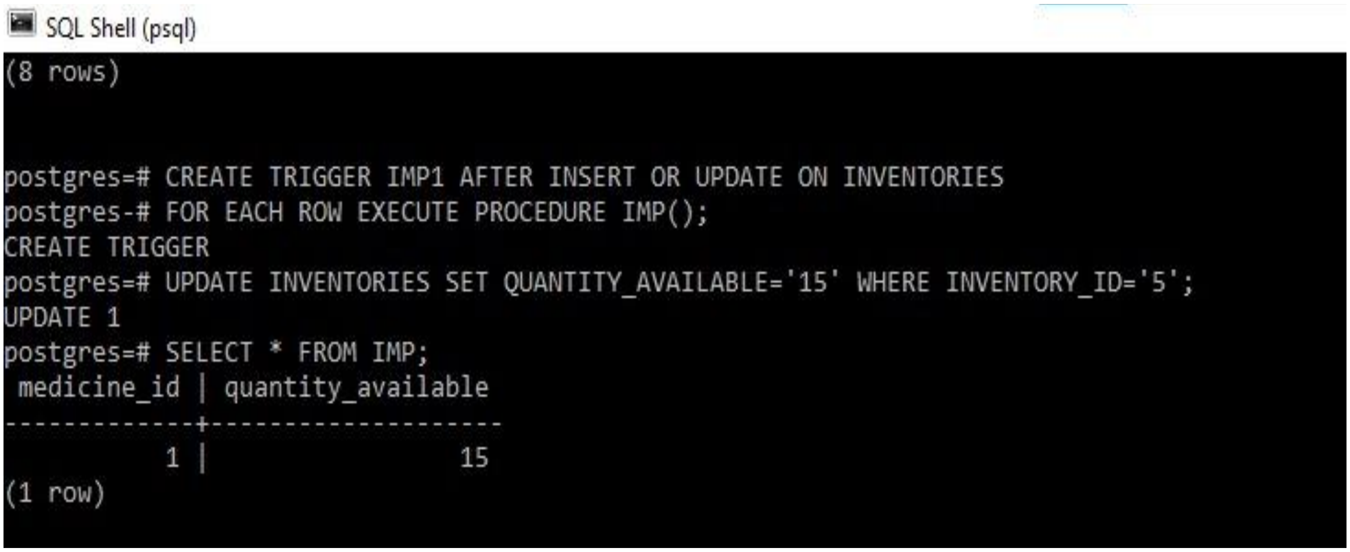


```
postgres=# SELECT * FROM INVENTORIES;
 inventory_id | wholesaler_id | medicine_id | quantity_available | expiry_date | medicine_price
--------------+---------------+-------------+--------------------+-------------+----------------
            1 |             3 |           6 |                105 | 2020-12-12  |          10.20
            2 |             1 |           3 |                100 | 2021-03-10  |          12.45
            3 |             2 |           7 |                 75 | 2022-04-22  |          15.40
            4 |             2 |           2 |                200 | 2021-06-11  |         129.45
            6 |             4 |           8 |                112 | 2021-10-16  |          58.60
            8 |             5 |           5 |                450 | 2023-06-15  |           5.62
            5 |             3 |           1 |                120 | 2021-12-04  |          13.50
            7 |             5 |           4 |                100 | 2021-01-01  |          38.20
(8 rows)


postgres=# _
```

TRIGGER FUNCTION:

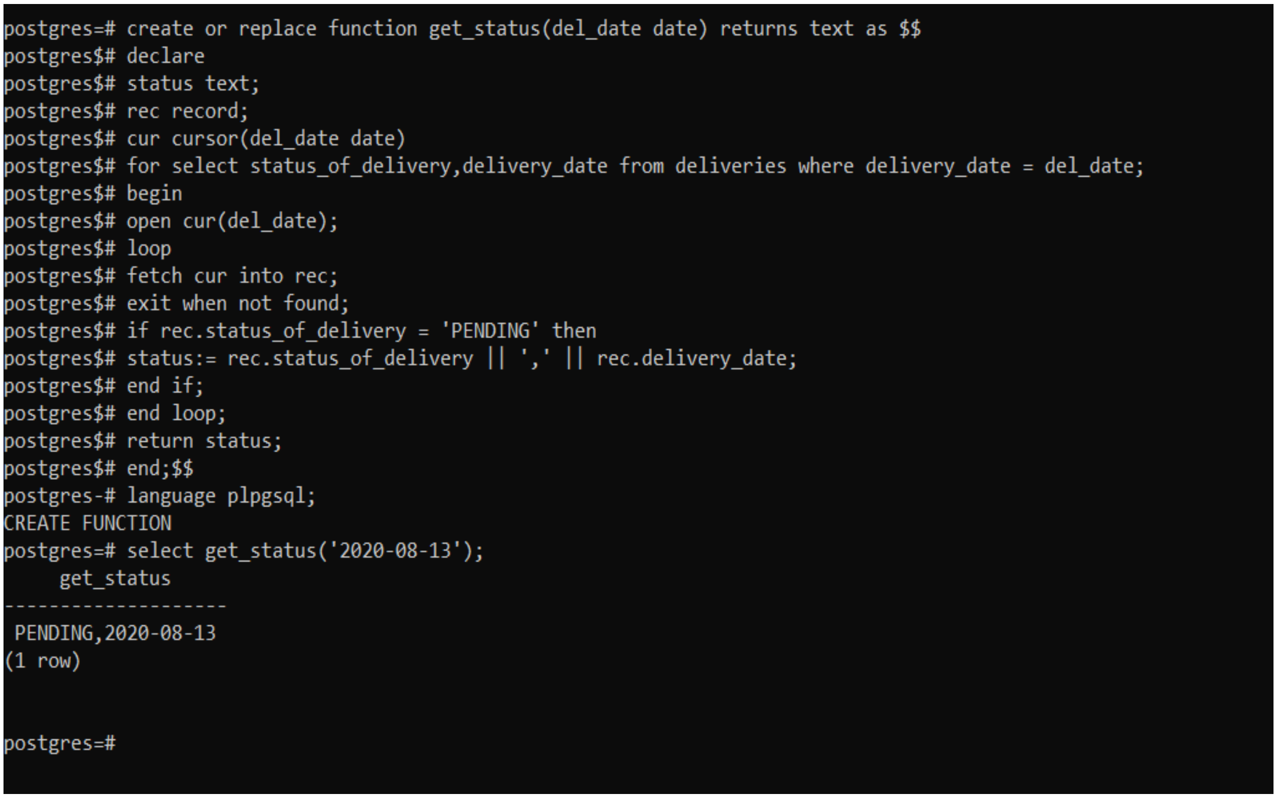
```
CREATE OR REPLACE FUNCTION imp() RETURNS TRIGGER AS $A1$
postgres$# BEGIN
postgres$# IF (NEW.quantity_available< 20) THEN
postgres$# INSERT INTO imp VALUES(NEW.quantity_availabel,NEW.medicine_id);
postgres$# RETURN NEW;
```

```
postgres$# END IF;
postgres$# END;
postgres$# $A1$
postgres-# LANGUAGE PLPGSQL;
CREATE FUNCTION
postgres=# CREATE TRIGGER imp1 AFTER INSERT OR UPDATE ON
INVENTORIES
postgres-# FOR EACH ROW EXECUTE PROCEDURE imp();
CREATE TRIGGER
```



```
(8 rows)


postgres=# CREATE TRIGGER IMP1 AFTER INSERT OR UPDATE ON INVENTORIES
postgres-# FOR EACH ROW EXECUTE PROCEDURE IMP();
CREATE TRIGGER
postgres=# UPDATE INVENTORIES SET QUANTITY_AVAILABLE='15' WHERE INVENTORY_ID='5';
UPDATE 1
postgres=# SELECT * FROM IMP;
 medicine_id | quantity_available
-------------+--------------------
           1 |                 15
(1 row)
```

2. FUNCTION TRIGGER TO FIND OUT THE STATUS OF DELIVERY OF DATE :



```
postgres=# create or replace function get_status(del_date date) returns text as $$
postgres$# declare
postgres$# status text;
postgres$# rec record;
postgres$# cur cursor(del_date date)
postgres$# for select status_of_delivery,delivery_date from deliveries where delivery_date = del_date;
postgres$# begin
postgres$# open cur(del_date);
postgres$# loop
postgres$# fetch cur into rec;
postgres$# exit when not found;
postgres$# if rec.status_of_delivery = 'PENDING' then
postgres$# status:= rec.status_of_delivery || ',' || rec.delivery_date;
postgres$# end if;
postgres$# end loop;
postgres$# return status;
postgres$# end;$$
postgres-# language plpgsql;
CREATE FUNCTION
postgres=# select get_status('2020-08-13');
     get_status
--------------------
 PENDING,2020-08-13
(1 row)


postgres=#
```

6. FUTURE ENHANCEMENTS OF THE SYSTEM

- Outputs , methods and user data input will be lot easy after the implement of GUI.

- We will design Front-end Design in HTML , CSS , JavaScript and Develop Bankend in Python.
- For security purpose New Registration for users is done using OTP.
- We will make database more consistent and We are making this database efficient and easy to implement with huge data capacity.
- We will also add some extra features and give consistent update so that users can get solutions of their problems.
- For designing this system we make assumption that delivery manages manufacturer but in actual this system may organize by other company which manages all deliveries and data of the users.

7. BIBLIOGRAPHY

- For the successful implementation of this project we referred to many websites and books.
- We created the ER Diagram and Schema Diagram on “Creatly.com”.
- we referred the online material for syntax of procedures, triggers,Exception and cursors.
- We also referred to ppts and lectures shared by faculties.
- For implementing database we use POSTGRES SQL and PGADMIN 4 software.

Reference Websites:
https://www.stackoverflow.com/

https://www.w3school.com/

https://www.tutorialspoint.com/

http://www.oracletutorial.com/

http://www.postgresqltutorial.com/