Pharmaceutical Store Management System

(Only the overview of project shared with replacing the actual dataset due to confidential clause)

Table of Contents :-

Description:-

The system was implemented by creating a database containing information about the stored medicines in the inventory, their suppliers, prices, and quantities. The main aim was to design and implement a pharmaceutical inventory database management system that can be used to facilitate a smooth workflow of purchase and sales operations of drugs and many other related actions and bring the advantages of having the most efficient control with minimal effort.

It provides access to two types of users: manager and clerk, both with varied access to functionality, their details stored in the ‘employee’ table of the database ‘pharma’.

Functionalities:

Sr no.	**Functionality**	**Who can access**
1.	Add an employee	Manager
2.	Remove an employee	Manager
3.	Check inventory/add stock	Manager/Clerk
4.	Administer a purchase	Manager/Clerk

System requirement specification:

```
Python 3.0 with the following libraries: Tkinter, psycopg2

PostgreSQL
```

E-R Diagram :-

Database name: pharma

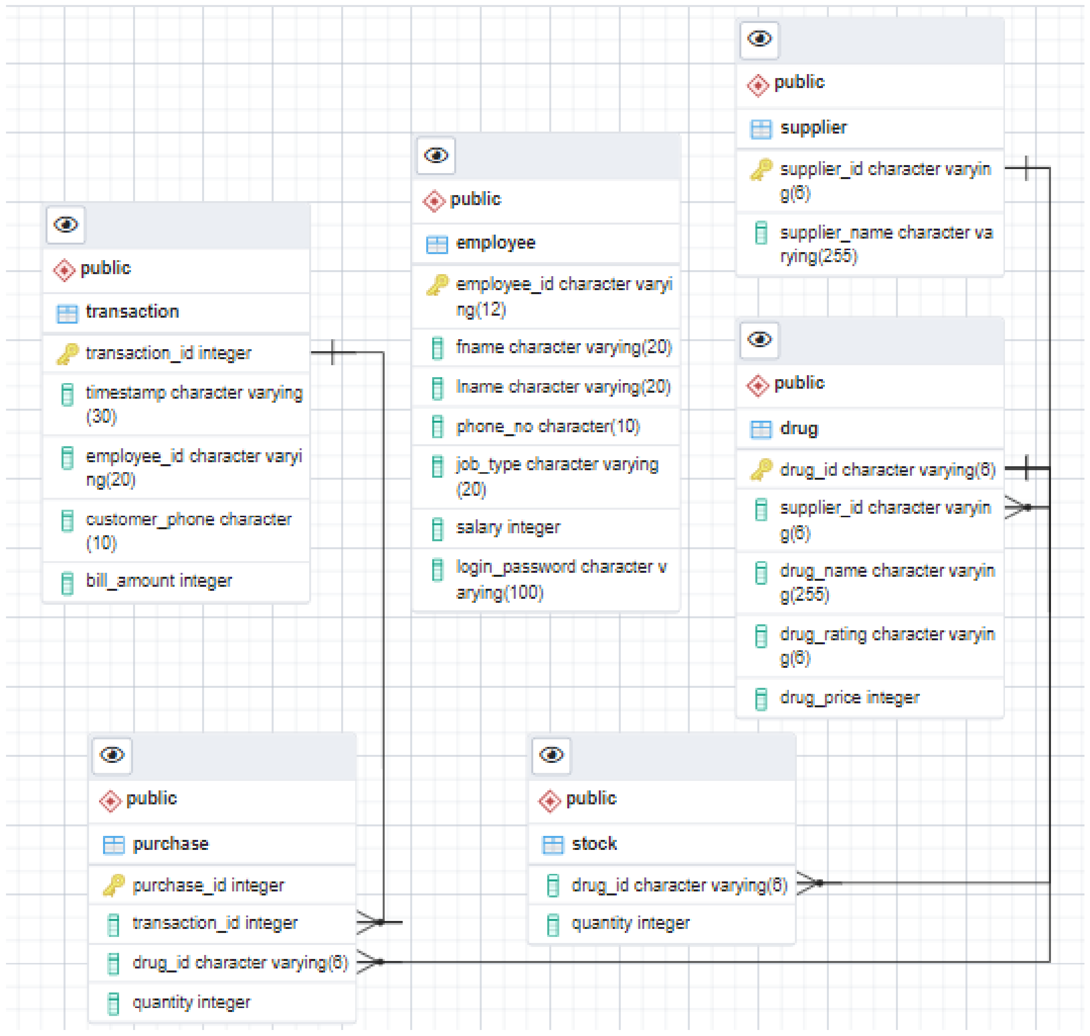
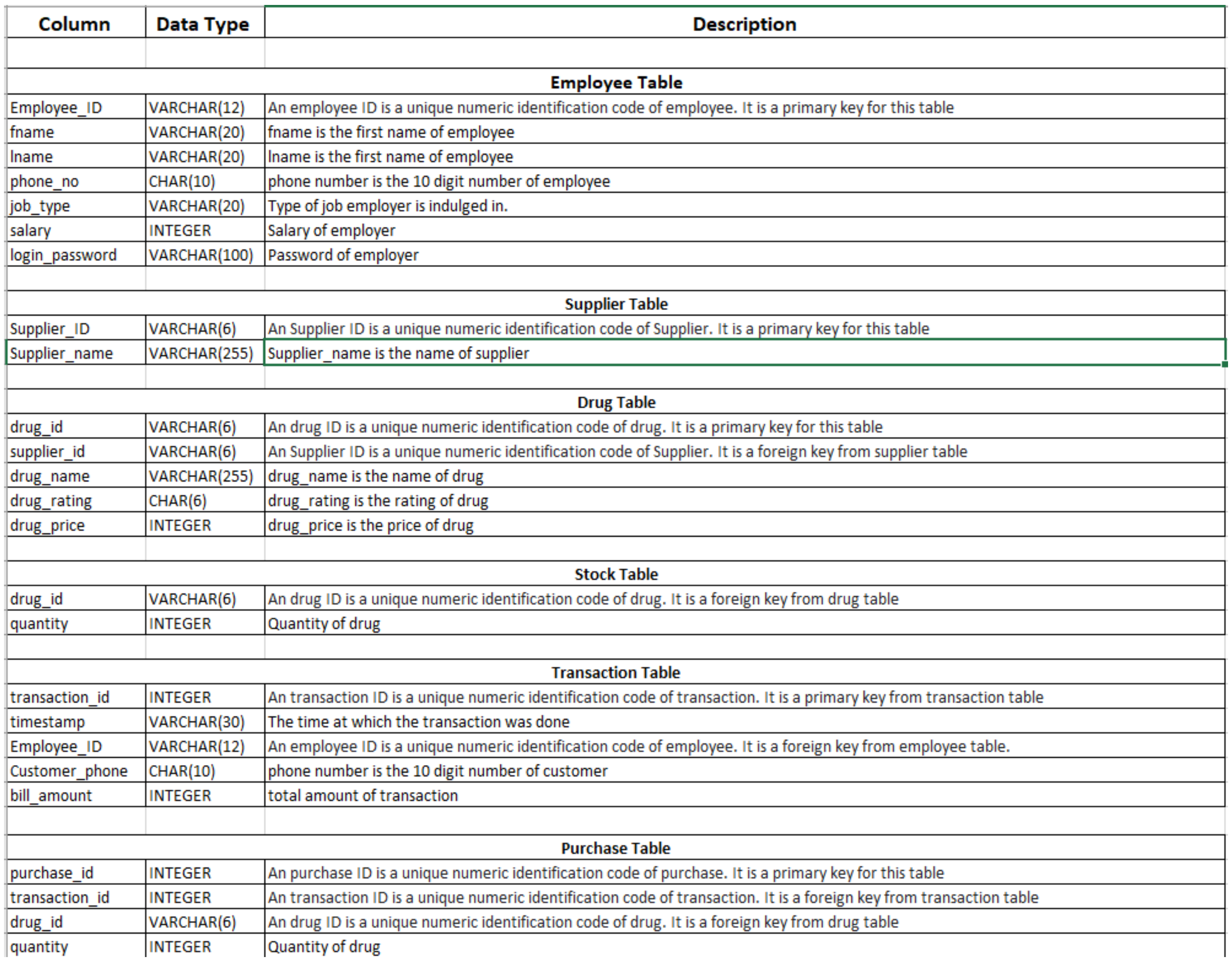

public
supplier
supplier_id character varying(6)
supplier_name character varying(255)
public
employee
employee_id character varying(12)
fname character varying(20)
lname character varying(20)
phone_no character(10)
job_type character varying (20)
salary integer
login_password character varying(100)
public
transaction
transaction_id integer
timestamp character varying (30)
employee_id character varying(20)
customer_phone character (10)
bill_amount integer
public
drug
drug_id character varying(6)
supplier_id character varying(6)
drug_name character varying(255)
drug_rating character varying(6)
drug_price integer
public
purchase
purchase_id integer
transaction_id integer
drug_id character varying(6)
quantity integer
public
stock
drug_id character varying(6)
quantity integer

Table Design:-



Column	Data Type	Description
Employee Table		
Employee_ID	VARCHAR(12)	An employee ID is a unique numeric identification code of employee. It is a primary key for this table
fname	VARCHAR(20)	fname is the first name of employee
lname	VARCHAR(20)	lname is the first name of employee
phone_no	CHAR(10)	phone number is the 10 digit number of employee
job_type	VARCHAR(20)	Type of job employer is indulged in.
salary	INTEGER	Salary of employer
login_password	VARCHAR(100)	Password of employer
Supplier Table		
Supplier_ID	VARCHAR(6)	An Supplier ID is a unique numeric identification code of Supplier. It is a primary key for this table
Supplier_name	VARCHAR(255)	Supplier_name is the name of supplier
Drug Table		
drug_id	VARCHAR(6)	An drug ID is a unique numeric identification code of drug. It is a primary key for this table
supplier_id	VARCHAR(6)	An Supplier ID is a unique numeric identification code of Supplier. It is a foreign key from supplier table
drug_name	VARCHAR(255)	drug_name is the name of drug
drug_rating	CHAR(6)	drug_rating is the rating of drug
drug_price	INTEGER	drug_price is the price of drug
Stock Table		
drug_id	VARCHAR(6)	An drug ID is a unique numeric identification code of drug. It is a foreign key from drug table
quantity	INTEGER	Quantity of drug
Transaction Table		
transaction_id	INTEGER	An transaction ID is a unique numeric identification code of transaction. It is a primary key from transaction table
timestamp	VARCHAR(30)	The time at which the transaction was done
Employee_ID	VARCHAR(12)	An employee ID is a unique numeric identification code of employee. It is a foreign key from employee table.
Customer_phone	CHAR(10)	phone number is the 10 digit number of customer
bill_amount	INTEGER	total amount of transaction
Purchase Table		
purchase_id	INTEGER	An purchase ID is a unique numeric identification code of purchase. It is a primary key for this table
transaction_id	INTEGER	An transaction ID is a unique numeric identification code of transaction. It is a foreign key from transaction table
drug_id	VARCHAR(6)	An drug ID is a unique numeric identification code of drug. It is a foreign key from drug table
quantity	INTEGER	Quantity of drug

List of Procedures & Functions:-

1. Get employee details for login
2. Add employee
3. Remove employee
4. Retrieve stock quantity and price
5. Add stock
6. Initiate transaction
7. Terminate transaction
8. Insert purchase
9. Complete transaction
10. Trigger functions

Procedure & Functions:-

1. Get employee details:-

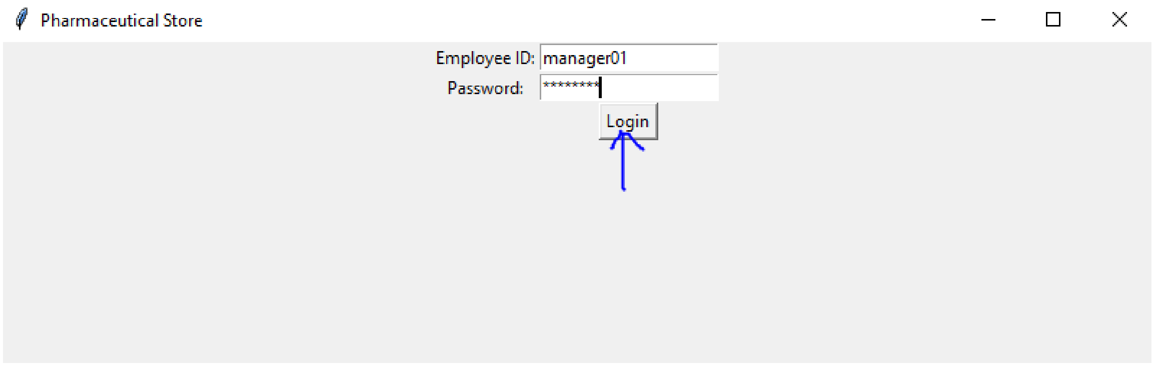
Code:-

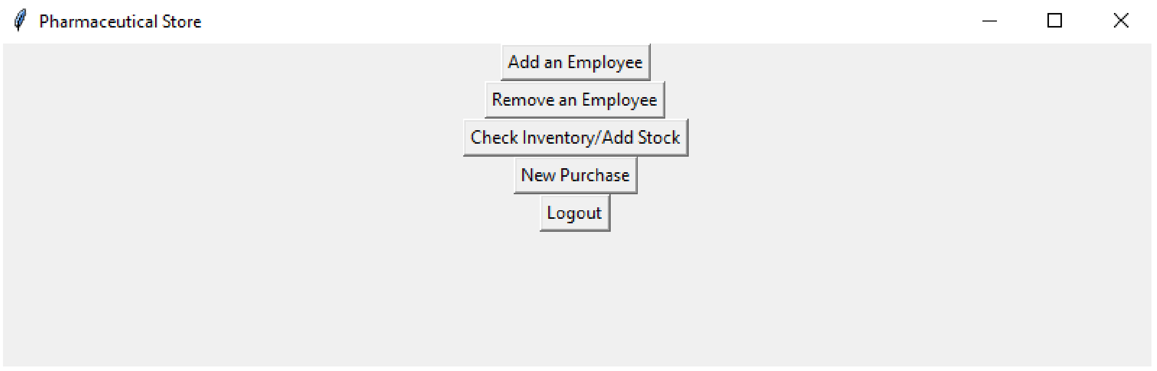
```
create or replace function get_empl(_emplid text)
RETURNS TABLE(loginpassword VARCHAR, jobtype VARCHAR)
AS $$
     begin
     RETURN QUERY
     SELECT login_password,job_type FROM employee WHERE employee_id=_emplid;
     end;
$$ LANGUAGE plpgsql;
```

Statement in python to call the function on front end:-

```
cur.callproc('get_empl',[removalID.get(),])
```

Results:- loginpassword is compared with entered password for login, jobtype is stored in a python variable. After successful login, a new frame is displayed.





2. Add employee:-

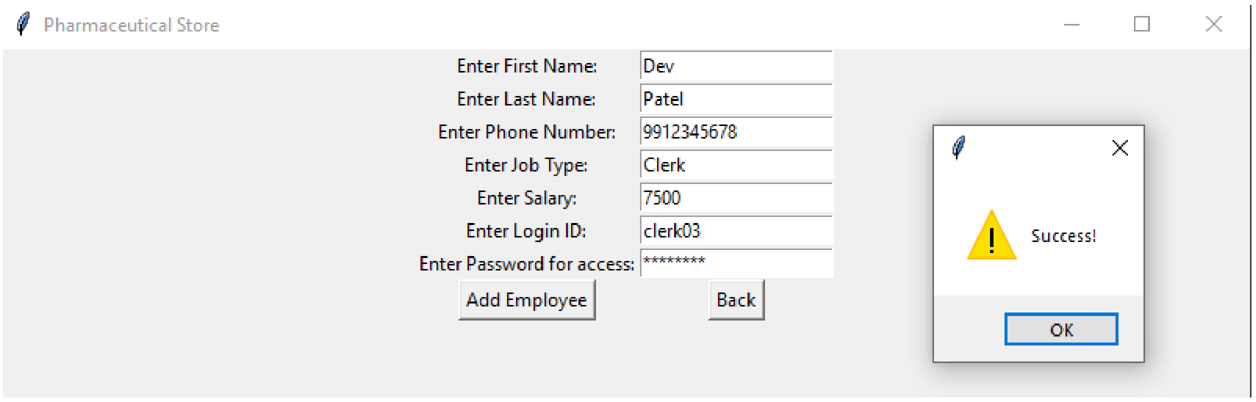
Code:-

```
create or replace function add_empl(_emplid text, _fname text, _lname text,
_phno text, _jobtype text, _salary integer, _loginpassword text)
returns void
AS $$
     begin
     INSERT INTO EMPLOYEE values(_emplid, _fname, _lname, _phno, _jobtype,
_salary, _loginpassword);
     end;
$$ LANGUAGE plpgsql;
```

Statement in python to call the function on front end:-

```
cur.callproc('add_empl',[loginId.get(),fname.get(),lname.get(),phNo.get(),job.
get(),int(salary.get()),loginPassword.get(),])
```

Results:-



Pharmaceutical Store

Enter First Name:	Dev
Enter Last Name:	Patel
Enter Phone Number:	9912345678
Enter Job Type:	Clerk
Enter Salary:	7500
Enter Login ID:	clerk03
Enter Password for access:	********

Success!

3. Remove employee:-

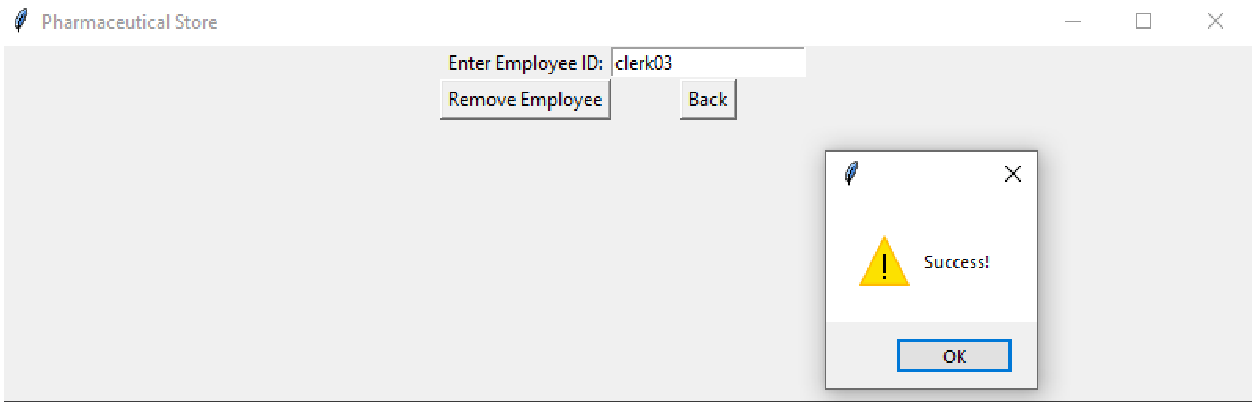
Code:-

```
create or replace function remove_empl(_emplid text)
RETURNS void
AS $$
     begin
     DELETE FROM employee WHERE employee_id=_emplid;
     end;
$$ LANGUAGE plpgsql;
```

Statement in python to call the function on front end:-

```
cur.callproc('remove_empl',[removalID.get(),])
```

Result:-


Pharmaceutical Store

Enter Employee ID: clerk03

Success!

4. Retrieve stock quantity and price:-

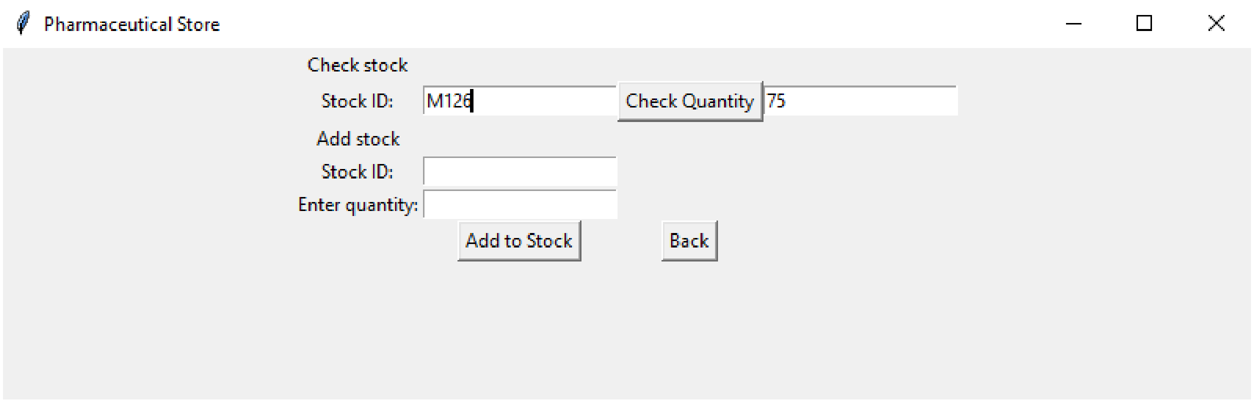
Code:-

```
create or replace function get_stock(_stockid text)
RETURNS TABLE(quantity INTEGER, drug_price INTEGER)
AS $$
begin
     RETURN QUERY
     select stock.quantity, drug.drug_price from
     drug join stock
     on drug.drug_id=stock.drug_id
     where drug.drug_id=_stockid;
end;
$$ LANGUAGE plpgsql;
```

Statement in python to call the function on front end:-

```
cur.callproc('get_stock',[drug[drug_id],])
```

Result:- Quantity available and price for the drug id entered are retrieved from the database. Quantity is displayed, and price is stored in a python variable for future use.



Stock ID: M126
Check Quantity: 75

5. Add stock to inventory:-

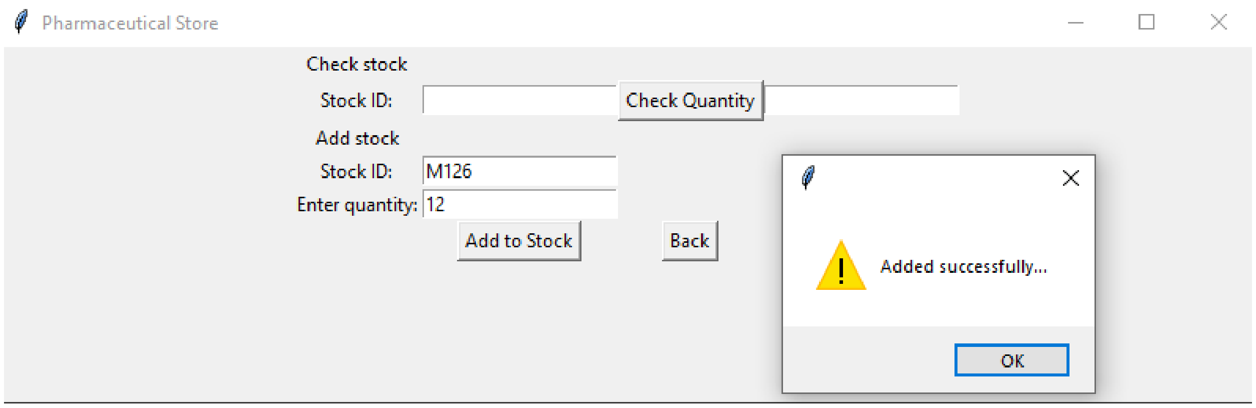
Code:-

```
create or replace function add_stock(_stockid text,_qty integer)
RETURNS void
AS $$
     begin
     UPDATE stock
     set quantity=quantity+_qty
     WHERE drug_id=_stockid;
     end;
$$ LANGUAGE plpgsql;
```

Statement in python to call the function on front end:-

```
cur.callproc('add_stock',[drugID,qty,])
```

Result:-



Stock ID: M126
Enter quantity: 12

Added successfully...

6. Initiate transaction:-

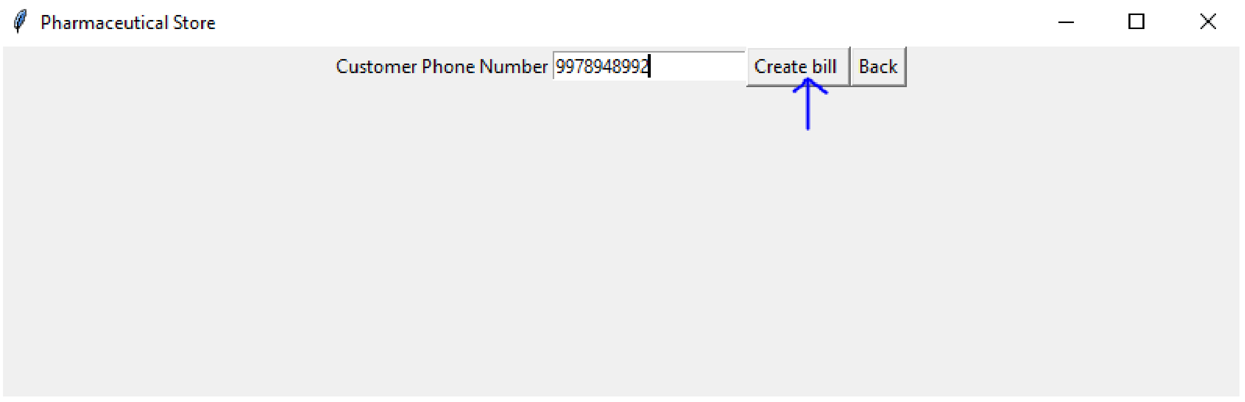
Code:-

```
create or replace function create_transaction(_transactionid integer, _emplid
text, _customerphone text)
returns void
AS $$
     begin
     INSERT INTO TRANSACTION values(_transactionid, current_timestamp(0),
_emplid, _customerphone, null);
     end;
$$ LANGUAGE plpgsql;
```

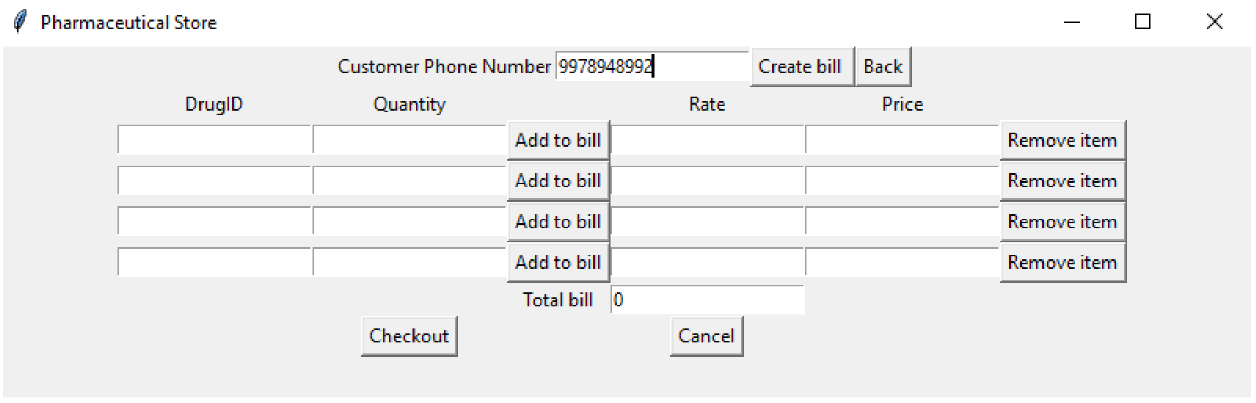
Statement in python to call the function on front end:-

```
cur.callproc('create_transaction',[transactionId,emplID,phoneNo,])
```

Result:-



Customer Phone Number 9978948992



Customer Phone Number 9978948992

Total bill 0

7. Terminate current transaction:-

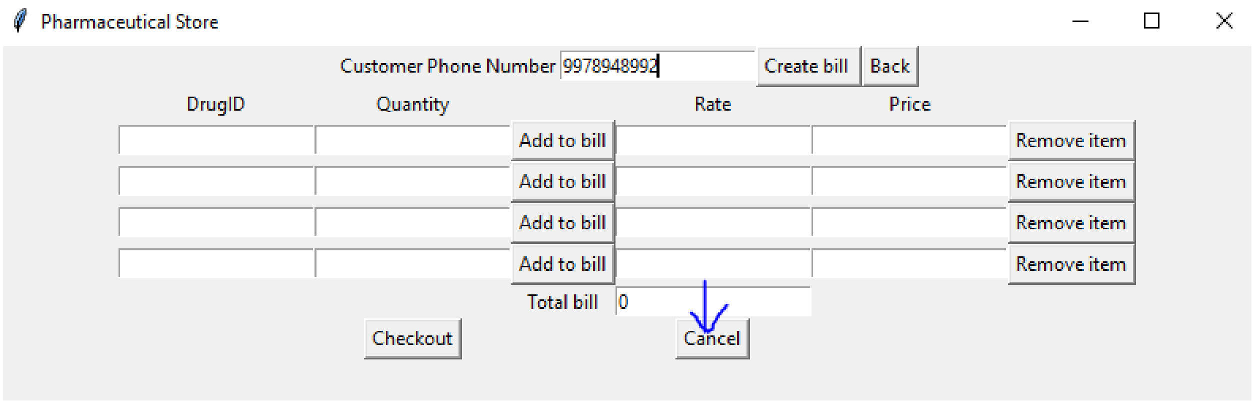
Code:-

```
create or replace function terminate_transaction(_transactionid integer)
RETURNS void
AS $$
     begin
     DELETE FROM transaction WHERE transaction_id=_transactionid;
     end;
$$ LANGUAGE plpgsql;
```

Statement in python to call the function on front end:-

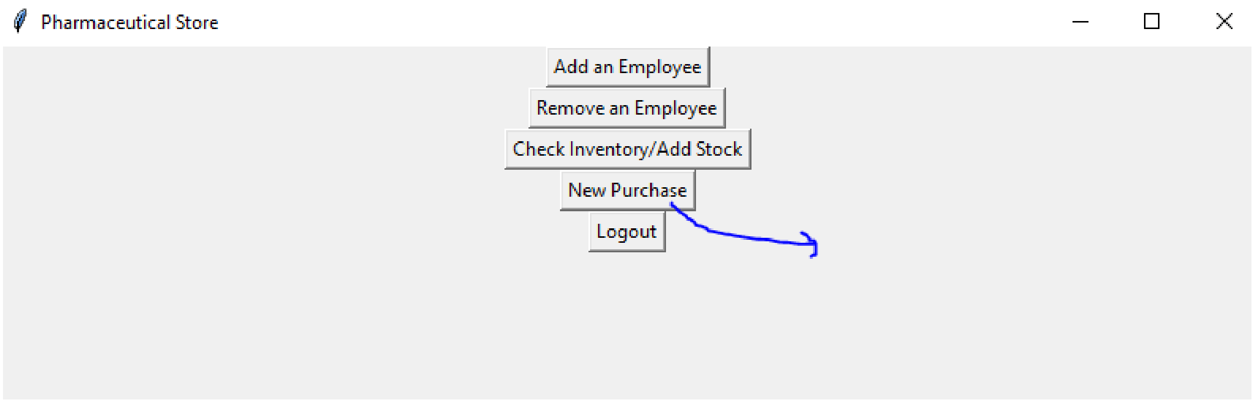
```
cur.callproc('terminate_transaction',[transactionId,]);
```

Result:- Current transaction is terminated and the user comes back from the 'new purchase' fame to the post-login frame.



Customer Phone Number 9978948992

Total bill 0



8. Insert purchase:-

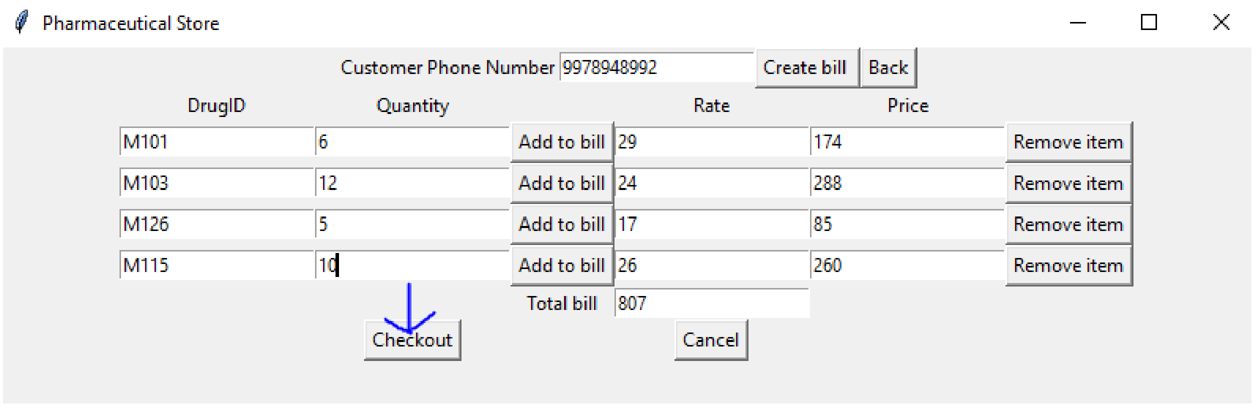
Code:-

```
create or replace function insert_purchase(_purchaseid integer, _transactionid
integer, _drugid text, _qty integer)
RETURNS void
AS $$
     begin
     INSERT INTO purchase VALUES(_purchaseid,_transactionid,_drugid,_qty);
     end;
$$ LANGUAGE plpgsql;
```

Statement in python to call the function on front end:-

```
cur.callproc('insert_purchase',[purchaseId,transactionId,drug[i],qty[i],]);
```

Result:- All the individual purchases are added into the purchase table stored in the database.



Customer Phone Number 9978948992

DrugID	Quantity	Rate	Price
M101	6	29	174
M103	12	24	288
M126	5	17	85
M115	10	26	260

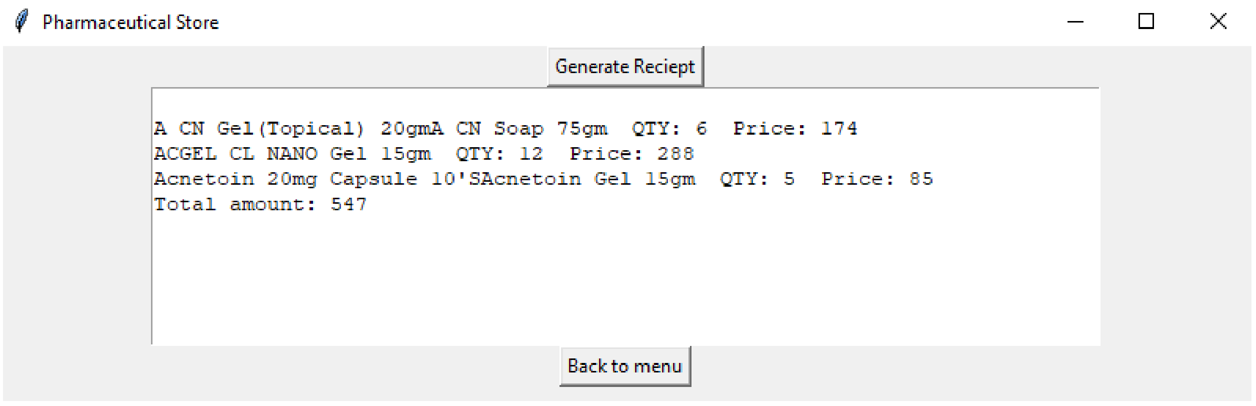
Total bill 807

9. Finish transaction :-

Code:-

```
create or replace function finish_transaction(_transactionid
integer,_billamount integer)
RETURNS void
AS $$
    begin
    UPDATE transaction
    set bill_amount=_billamount
    WHERE transaction_id=_transactionid;
    end;
$$ LANGUAGE plpgsql;
cur.callproc('finish_transaction',[transactionId,totalAmt,]);
```

Result:- Total bill amount is updated in the current transaction, and receipt is displayed on the front-end.



```
A CN Gel(Topical) 20gmA CN Soap 75gm  QTY: 6  Price: 174
ACGEL CL NANO Gel 15gm  QTY: 12  Price: 288
Acnetoin 20mg Capsule 10'SAcnetoin Gel 15gm  QTY: 5  Price: 85
Total amount: 547
```

Tiggers:-

1.Employee Salary Check

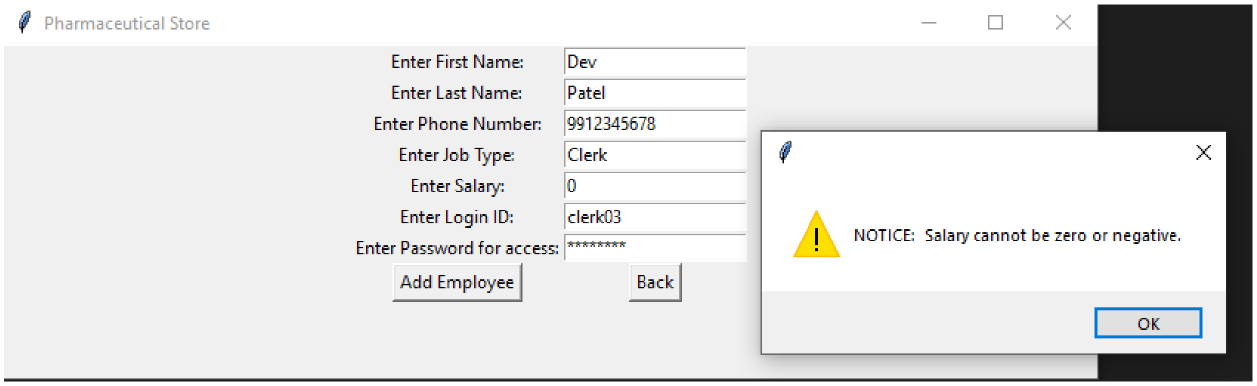
Code:-

```
CREATE TRIGGER empl_sal_check
BEFORE INSERT OR UPDATE
ON employee
FOR EACH ROW
when(NEW.salary <= 0)
EXECUTE PROCEDURE raise_sal_error();
```

Trigger function:-

```
create or replace function raise_sal_error()
RETURNS trigger
AS $$
     begin
     RAISE NOTICE 'Salary cannot be zero or negative.';
     end;
$$ LANGUAGE plpgsql;
```

Screenshot:-



Enter First Name:	Dev
Enter Last Name:	Patel
Enter Phone Number:	9912345678
Enter Job Type:	Clerk
Enter Salary:	0
Enter Login ID:	clerk03
Enter Password for access:	********

NOTICE: Salary cannot be zero or negative.

2.Employee phone number check:-

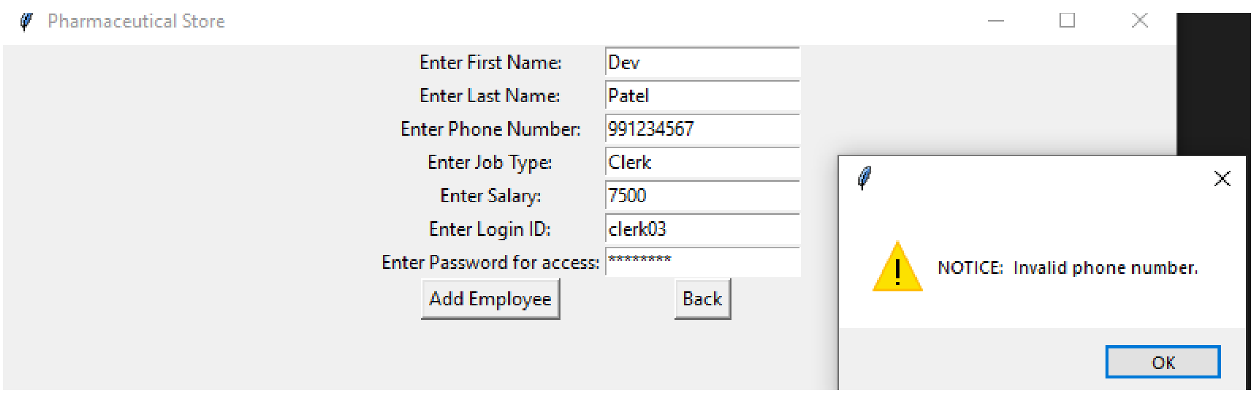
Code:-

```
CREATE TRIGGER phno_validity_check
BEFORE INSERT OR UPDATE
ON employee
FOR EACH ROW
when(length(NEW.phone_no)<>10)
  EXECUTE PROCEDURE raise_phno_error();
```

Trigger function-

```
create or replace function raise_phno_error()
RETURNS trigger
AS $$
     begin
     RAISE NOTICE 'Invalid phone number.';
     end;
$$ LANGUAGE plpgsql;
```

Screenshot:-



Pharmaceutical Store

Enter First Name:	Dev
Enter Last Name:	Patel
Enter Phone Number:	991234567
Enter Job Type:	Clerk
Enter Salary:	7500
Enter Login ID:	clerk03
Enter Password for access:	********

NOTICE: Invalid phone number.

3.Employee job type check:-

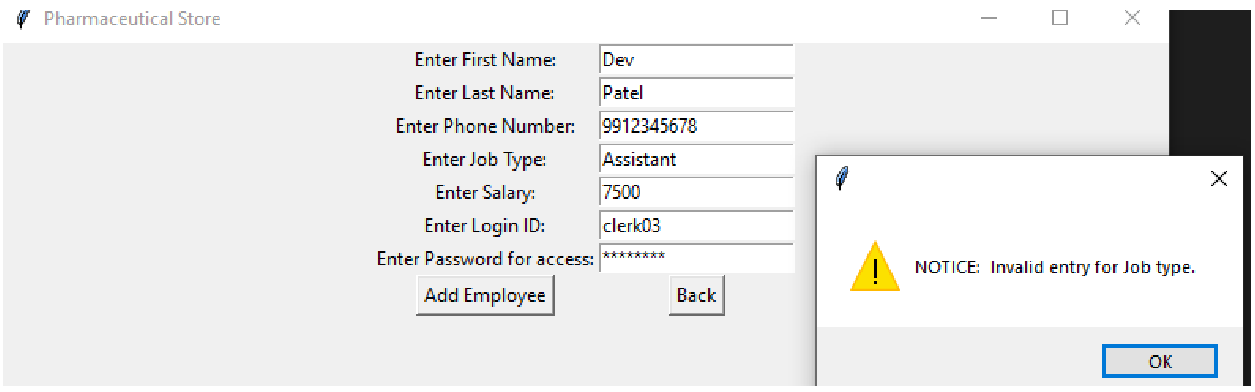
Code:-

```
CREATE TRIGGER jobtype_validity_check
BEFORE INSERT OR UPDATE
ON employee
FOR EACH ROW
when(lower(NEW.job_type) not in ('manager','clerk'))
  EXECUTE PROCEDURE raise_jobtype_error();
```

Trigger function:-

```
create or replace function raise_jobtype_error()
RETURNS trigger
AS $$
     begin
     RAISE NOTICE 'Invalid entry for Job type.';
     end;
$$ LANGUAGE plpgsql;
```

Screenshot:-



Pharmaceutical Store

Enter First Name:	Dev
Enter Last Name:	Patel
Enter Phone Number:	9912345678
Enter Job Type:	Assistant
Enter Salary:	7500
Enter Login ID:	clerk03
Enter Password for access:	********

NOTICE: Invalid entry for Job type.

4.Employee password check:-

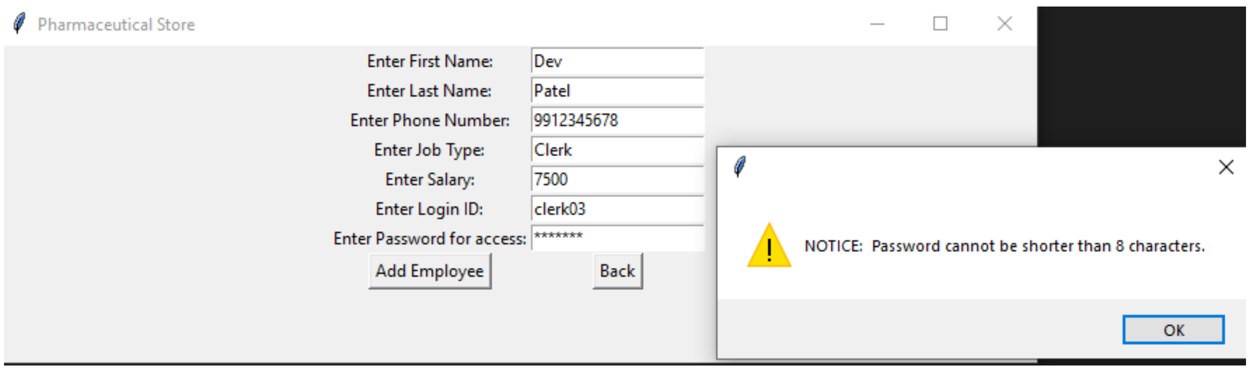
Code:-

```
CREATE TRIGGER password_validity_check
BEFORE INSERT OR UPDATE
ON employee
FOR EACH ROW
when(length(NEW.login_password)<8)
  EXECUTE PROCEDURE raise_password_error();
```

Trigger function:-

```
create or replace function raise_password_error()
RETURNS trigger
AS $$
     begin
     RAISE NOTICE 'Password cannot be shorter than 8 characters.';
     end;
$$ LANGUAGE plpgsql;
```

Screenshot:-



Enter First Name:	Dev
Enter Last Name:	Patel
Enter Phone Number:	9912345678
Enter Job Type:	Clerk
Enter Salary:	7500
Enter Login ID:	clerk03
Enter Password for access:	*******

NOTICE: Password cannot be shorter than 8 characters.

5.Time check:-

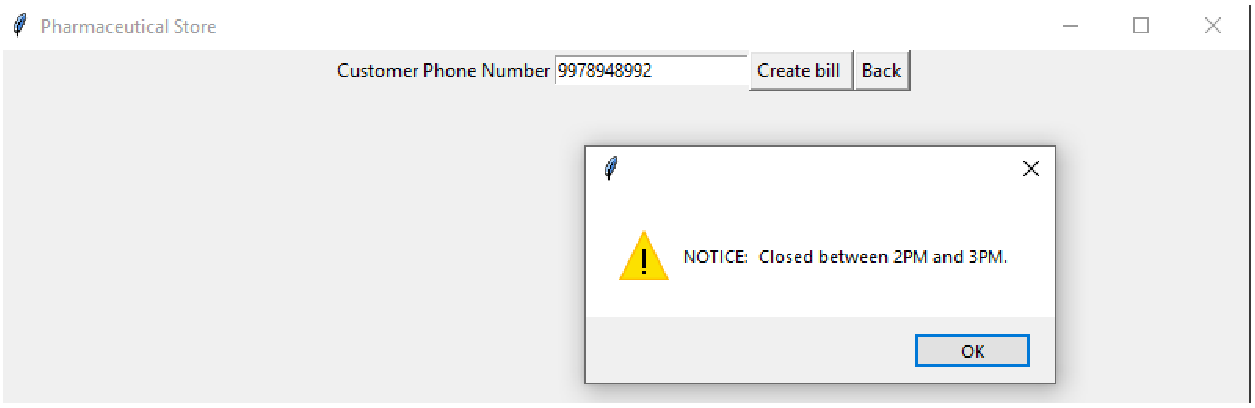
Code:-

```
CREATE TRIGGER time_check
BEFORE INSERT
ON transaction
FOR EACH ROW
when(date_part('hour', (current_timestamp))=14)
  EXECUTE PROCEDURE raise_time_error();
```

Trigger function:-

```
create or replace function raise_time_error()
RETURNS trigger
AS $$
     begin
     RAISE NOTICE 'Closed between 2PM and 3PM.';
     end;
$$ LANGUAGE plpgsql;
```

Screenshot:-



9978948992

NOTICE: Closed between 2PM and 3PM.

CONCLUSION

Designed and developed an E-R Diagram for a pharmaceutical inventory database management system, streamlining the workflow of purchase and sales operations for drugs. Increased efficiency by 40%.Implemented a Python script to automate data entry and retrieval in the pharmaceutical store management system, reducing manual effort by 75%. Utilized SQL queries to generate comprehensive reports on product performance, leading to a 20% increase in revenue.