**PES UNIVERSITY**

**DBMS: VIRTUAL MONEY TRANSFER**



**Submitted By:**

**Name: ANSHIKA PAL**

**SRN: PES1UG20CS062**

**V Semester Section B**

**DESCRIPTION:**

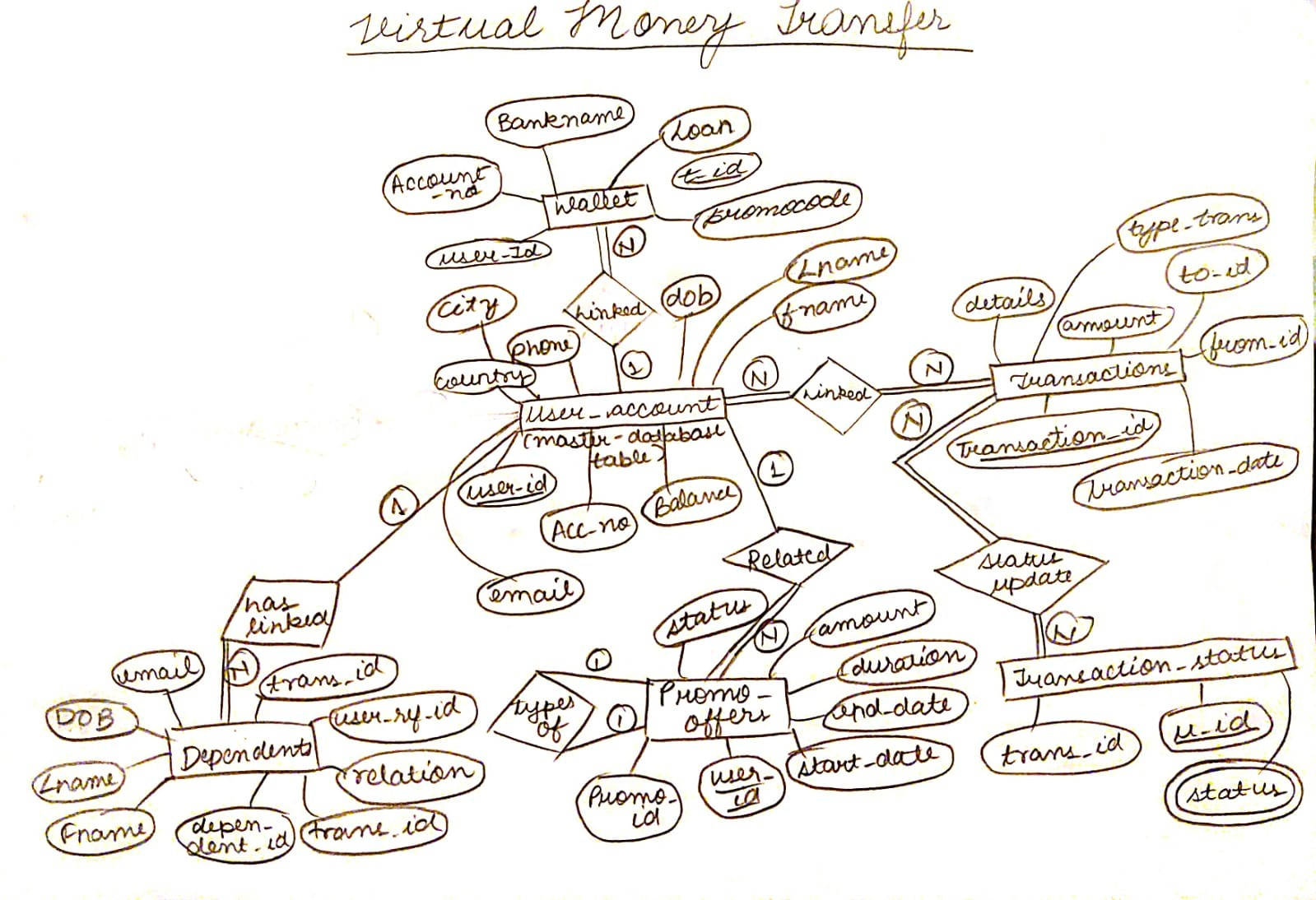
Virtual Money Transfer(E- Wallet) online payment portal, created using mysql and streamlit(python) with a MySQL database, where users can transfer money through virtual wallets. Transactions involving virtual currencies occur through secure, dedicated networks or over the Internet.

It allows to send money from one person to another and also keeps the history about all the previous transactions. You are also given Promo Offers which one can use and get rewards. This also allows people to give referral to other people and can claim offers. It will also show the loan on a person.

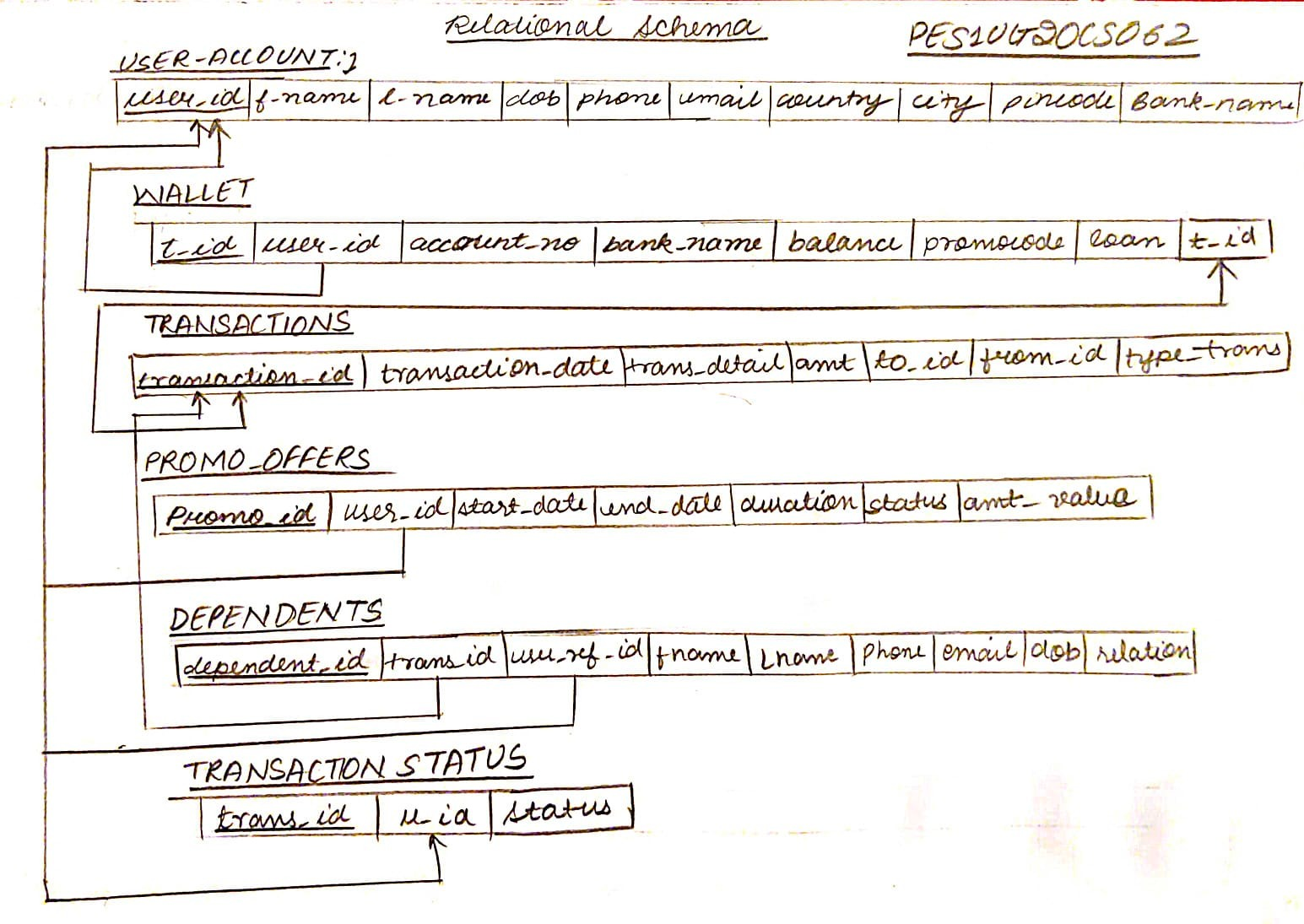
**SCOPE:**

1. Customers can perform financial transactions like transfer funds online, pay bills, apply for loans and open a savings account among various other debit card transactions.
2. You can transfer the funds 24/7 without going to the physical bank.
3. Check account Balance and statements.
4. One can apply for loans.
5. Make investments.
6. Bill payments and recharge.
7. Depositing and withdrawing money.

**ER Diagram:**



**Relational Schema:**



**DDL statements - Building the database:**

**create\_databse.sql**

create table user\_account(user\_id varchar(10),

fname varchar(15) NOT NULL,

lname varchar(15) NOT NULL,

dob varchar(30) NOT NULL,

phone varchar(10) NOT NULL ,

email varchar(40) NOT NULL,

country varchar(20) NOT NULL,

city varchar(30),

pincode int(6),

bank\_name varchar(20) NOT NULL,

primary key(user\_id));

create table wallet(t\_id varchar(20) NOT NULL,

user\_id varchar(20) NOT NULL,

account\_no varchar(20) NOT NULL,

bank\_name varchar(30) NOT NULL,

balance int check(balance > 0),

promocode varchar(20) default NULL,

loan varchar(10) default NULL,

primary key(t\_id,user\_id,promocode));

create table transactions(transaction\_id varchar(10),

transaction\_date varchar(20),

transaction\_detail varchar(30),

amount int check (amount>100),

to\_id varchar(20),

from\_id varchar(20),

type\_trans varchar(20),

primary key(transaction\_id,to\_id,from\_id));

create table promo\_offers(promo\_id varchar(10),

user\_id varchar(10),

start\_date varchar(20) NOT NULL,

end\_date varchar(20) NOT NULL,

duration varchar(20) NOT NULL,

status varchar(20) NOT NULL,

amount\_value varchar(50) NOT NULL,

primary key(promo\_id,user\_id),

FOREIGN KEY (user\_id) REFERENCES user\_account(user\_id) ON DELETE CASCADE);

create table dependents(dependent\_id varchar(10) NOT NULL,

trans\_id varchar(10) UNIQUE,

user\_ref\_id varchar(20),

fname varchar(20) NOT NULL,

lname varchar(20) NOT NULL,

phone varchar(10) NOT NULL,

email varchar(50) NOT NULL,

dob varchar(30) NOT NULL,

relation varchar(20) NOT NULL UNIQUE,

primary key(dependent\_id,trans\_id,user\_ref\_id));

create table transaction\_status(trans\_id varchar(10),

u\_id varchar(10),

primary key (trans\_id,u\_id));

ALTER TABLE wallet ADD CONSTRAINT FK\_t\_id FOREIGN KEY (t\_id) REFERENCES transactions(transaction\_id) ON DELETE CASCADE;

ALTER TABLE transaction\_status

ADD CONSTRAINT FK\_u\_id

FOREIGN KEY (u\_id) REFERENCES user\_account(user\_id)

ON DELETE CASCADE;

ALTER TABLE transaction\_status

ADD status varchar(15);

**Populating the Database:**

**insert\_data.sql**

insert into user\_account values ('VB201701','Vishal','Khanna','15-03-1979','9912121212','vishalkhanna@yahoo.com','India','Chandigarh','123456','State Bank Of India');

insert into user\_account values ('VB201702','Anuj','Bhushan','05-06-1996','8197765465','anujbhushan5@gmail.com','India','Bangalore','123890','State Bank Of India');

insert into user\_account values ('VB201703','Aniket','Bharati','11-01-1997','8147364941','bharatianiket@gmail.com','India','Bangalore','123890','Union Bank Of India');

insert into user\_account values ('VB201704','Ankit','Reddy','11-02-1997','8179949418','hdreddy97@gmail.com','India','Bangalore','123890','ICICI Bank');

insert into user\_account values ('VB201705','Shreya','Narayan','11-09-1996','9913131313','narayanshreya@yahoo.com','India','Mangalore','123476','Axis Bank');

insert into user\_account values ('VB201706','Dilip','Joshi','01-01-1980','9914141414','dilipjoshi@yahoo.com','India','Delhi','123019','Axis Bank');

/\*insert into wallet values('T-220001','VB201701','SBIN201701','State Bank Of India',7500,'','');\*/

insert into wallet values('T-220002','VB201702','SBIN201702','State Bank Of India',45000,'VB\_MAR','');

insert into wallet values('T-220003','VB201703','UBIN201703','State Bank Of India',6000,'VB\_MAR','1200');

insert into wallet values('T-220004','VB201704','ICIN201704','State Bank Of India',50000,'VB\_MAR','');

insert into wallet values('T-220005','VB201705','AXIN201705','State Bank Of India',9500,'','5000');

insert into wallet values('T-220006','VB201706','AXIN201706','State Bank Of India',1000,'','2000');

insert into transactions values('T-220002','22-03-2017','Transfer to Wife','50000','VB201705','VB201702','Wallet Transfer');

insert into transactions values('T-220003','22-03-2017','Transfer to Friend','20000','VB201704','VB201703','NEFT Transfer');

insert into transaction\_status values('T-220002','VB201702',’Completed’);

insert into transaction\_status values('T-220003','VB201703',’Failed’);

insert into promo\_offers values('VB\_MAR','VB201702','01-03-2017','31-03-2017','30 days','Active','Rs. 1000 V\_Credit');

insert into promo\_offers values('VB\_MAR','VB201703','01-03-2017','31-03-2017','30 days','Active','Rs. 1000 V\_Credit');

insert into promo\_offers values('VB\_MAR','VB201704','01-03-2017','31-03-2017','30 days','Active','Rs. 1000 V\_Credit');

insert into dependents values('VBDP001','T-210002','T-220002','Virat','Kohli','8812121212','vkohli87@gmail.com','20-01-1987','Brother');

insert into dependents values('VBDP002','T-210004','T-220004','Neha','Krishna','8814141414','krihna\_neha97@rediffmail.com','20-02-1987','Wife');

insert into dependents values('VBDP003','T-210005','T-220005','Vaibhav','Bhushan','8815151515','vaibhav\_born99@gmail.com','20-03-1980','Brother In Law');

**Join Queries:**

1. **Displaying user\_id, account\_no, bank\_name and transaction details of a person sending money to relative.**

**QUERY:**

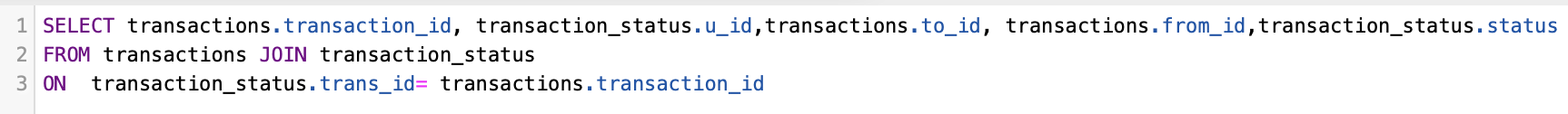
****

**RESULT:**

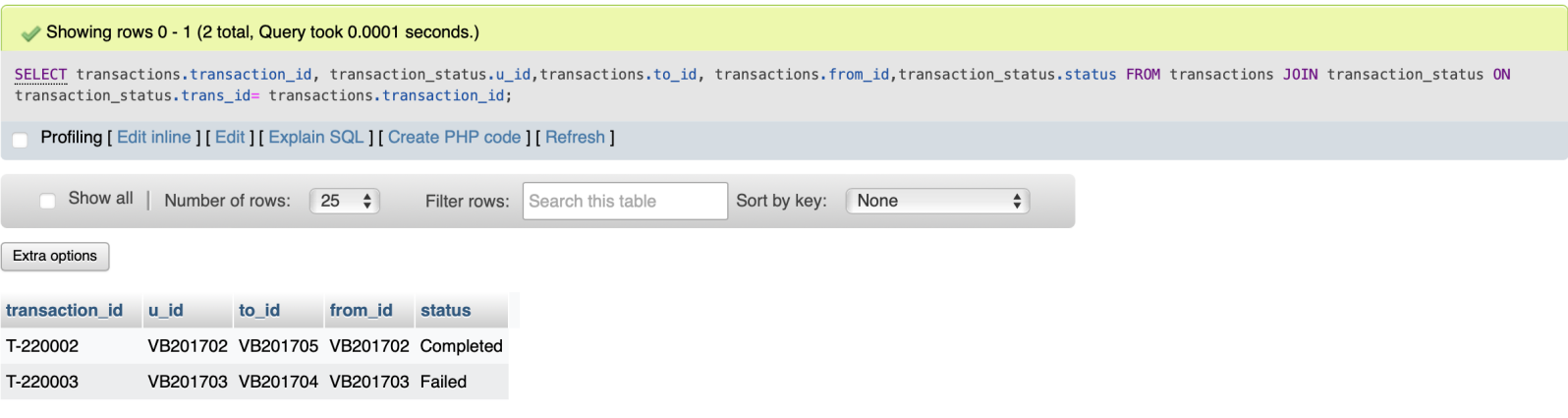
****

1. **SELECT transaction\_id,u\_id,to\_id,from\_id, status from joining tables transactions and transaction\_status on transaction id. (Showing if the transaction is completed or not).**

**QUERY:**

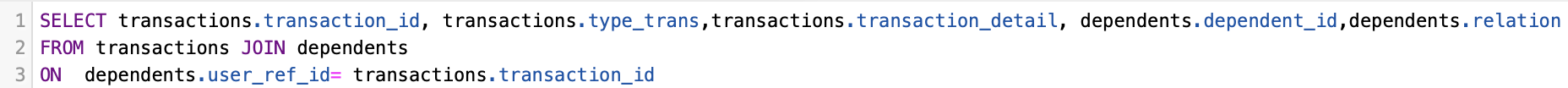
****

**RESULT:**

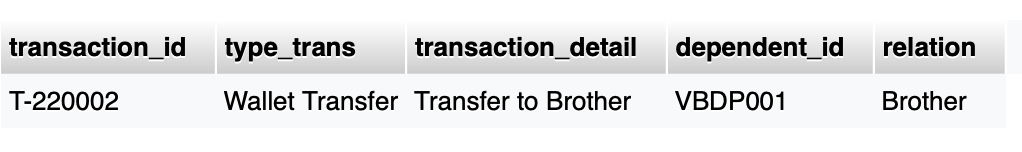
****

1. **Selecting transaction\_id, dependent\_id, relation, transaction\_typ and relation by joining tables transactions and dependents.**

**QUERY:**

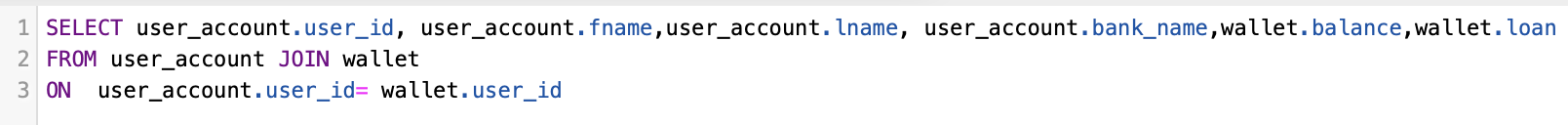
****

**RESULT:**

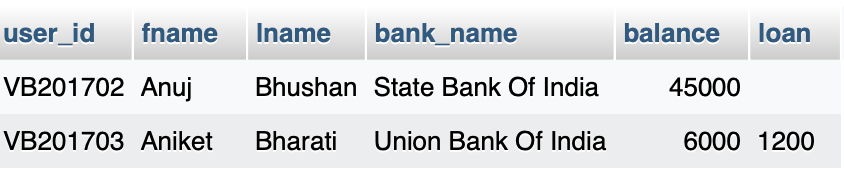
****

1. **Show the user\_id with it’s name and bank number and how much balance and loan is there on a person.**

**QUERY:**

****

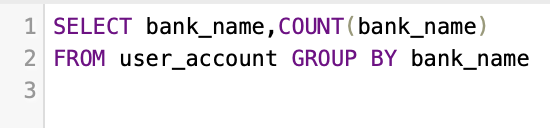
**RESULT:**

****

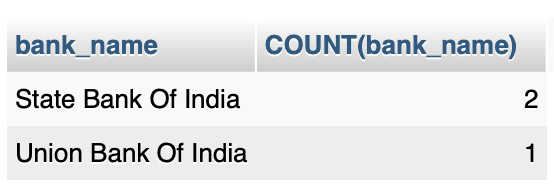
**AGGREGATE FUNCTIONS:**

1. **Counting the number of banks that user is associated with.**

**QUERY:**

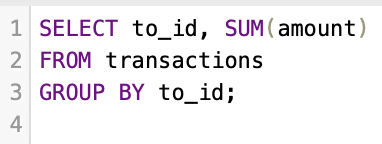
****

**RESULT:**

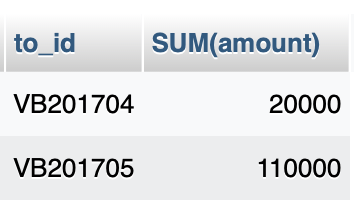
****

1. **Show total sum sent to a person.**

**QUERY:**

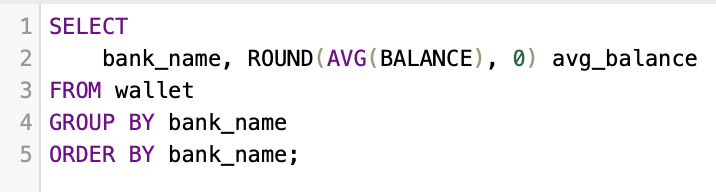
****

**RESULT:**

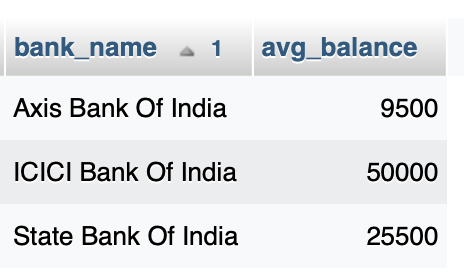
****

1. **Give average values of balance on which the transactions takes place of all the banks.**

**QUERY:**

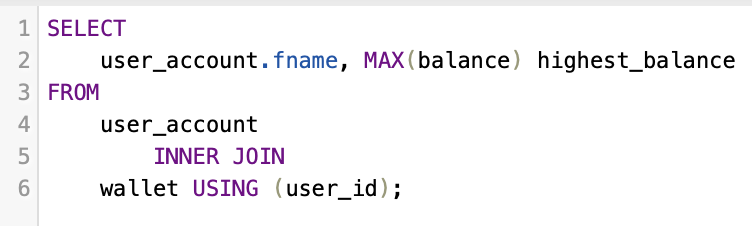
****

**RESULT:**

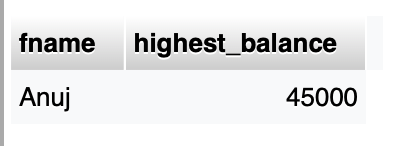
****

1. **Highest balance of all the users in the wallet.**

**QUERY:**

****

**RESULT:**

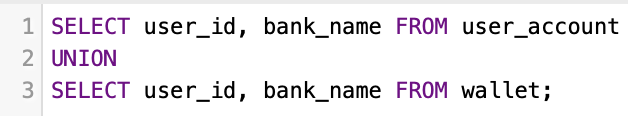
****

**#ANUJ HAS HIGHEST BALANCE**

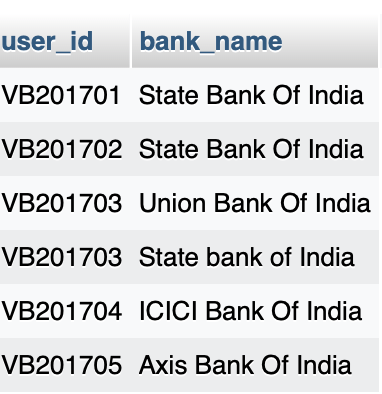
**Set Operations:**

1. **Perform union function on user\_id and bank\_name on user\_account and wallet columns.**

**QUERY:**

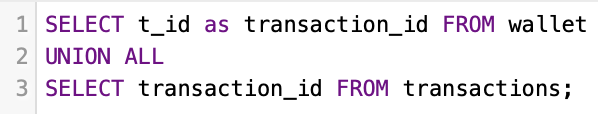
****

**RESULT:**

****

1. **UNION ALL on transaction id of wallet and transactions.**

**QUERY:**

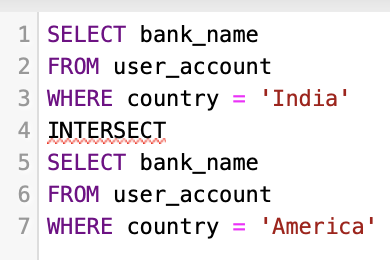
****

**RESULT:**

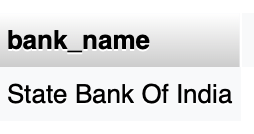
****

1. **Bank name tp which a person registered while living in different countries (i.e. India and America).**

**QUERY:**

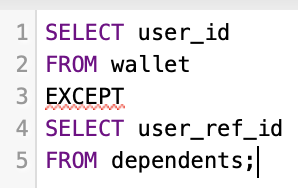
****

**RESULT:**

****

1. **Users which didnot give refferal to any person (or have any dependents).**

**QUERY:**

****

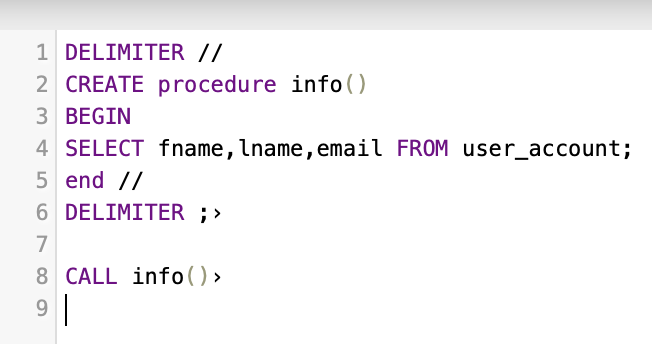
**RESULT:**

****

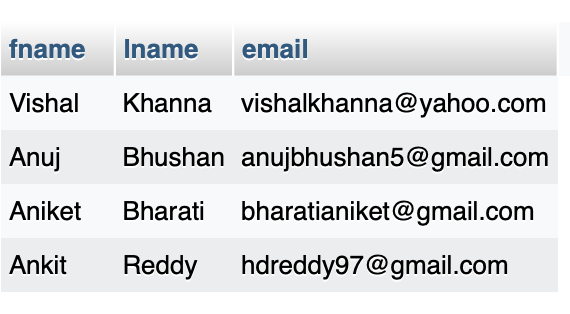
**Functions and Procedures:**

**PROCEDURE:**

**QUERY:**

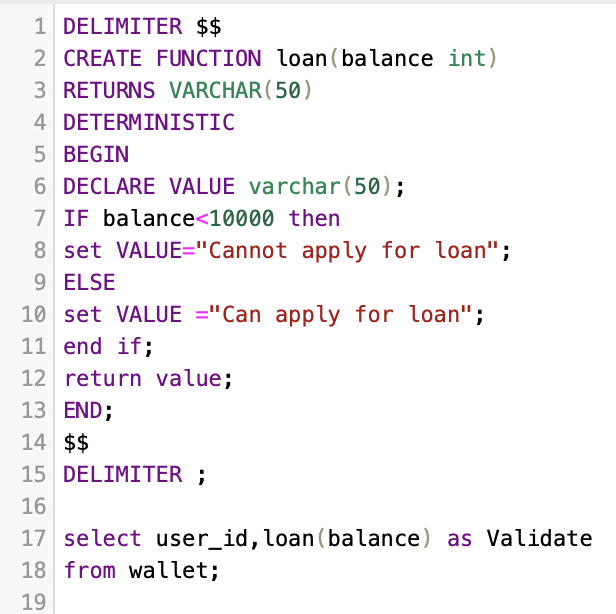


**RESULTS:**

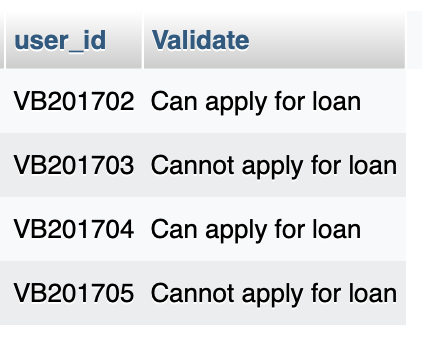
****

**FUNCTION:**

**QUERY:**

****

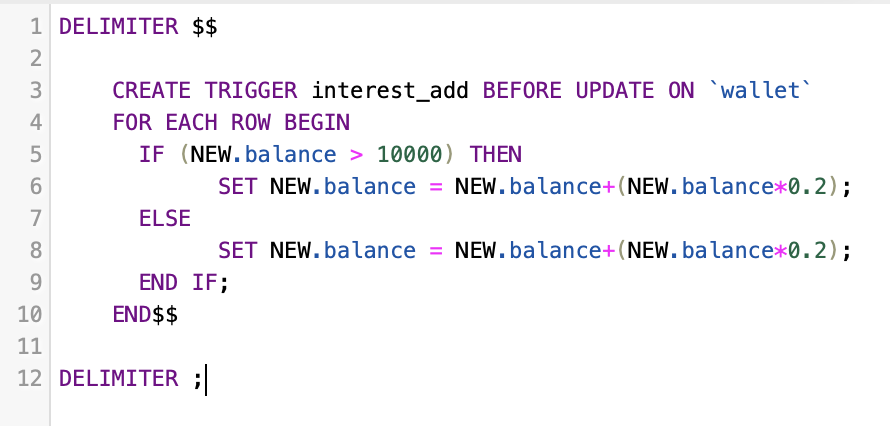
**RESULTS:**

****

**TRIGGERS AND CURSORS:**

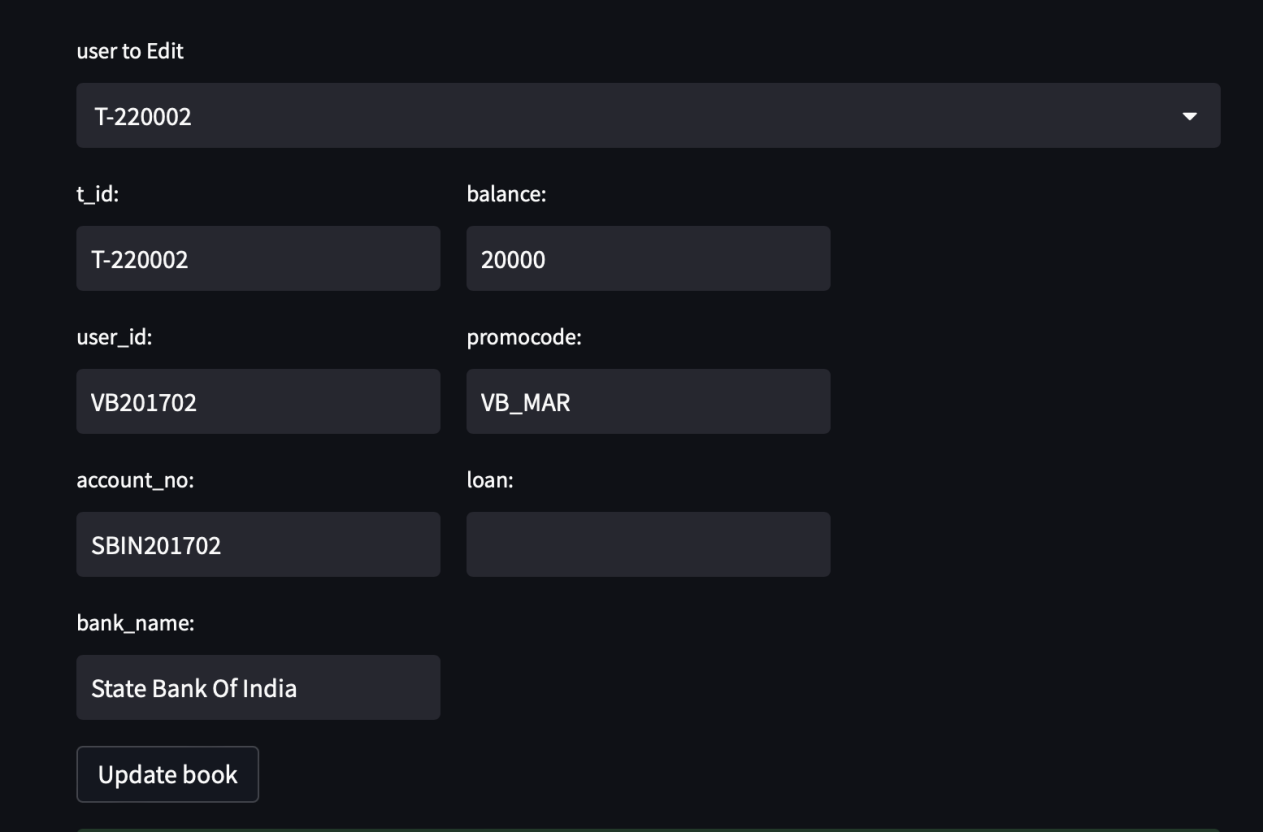
**TRIGGERS:**

**QUERY:**

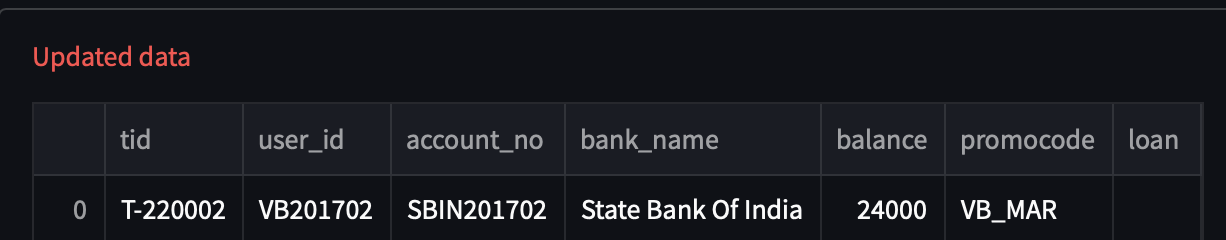
****

**RESULT:**

**UPDATED THE BALANCE TO 20000 WHICH IS > 10000:**

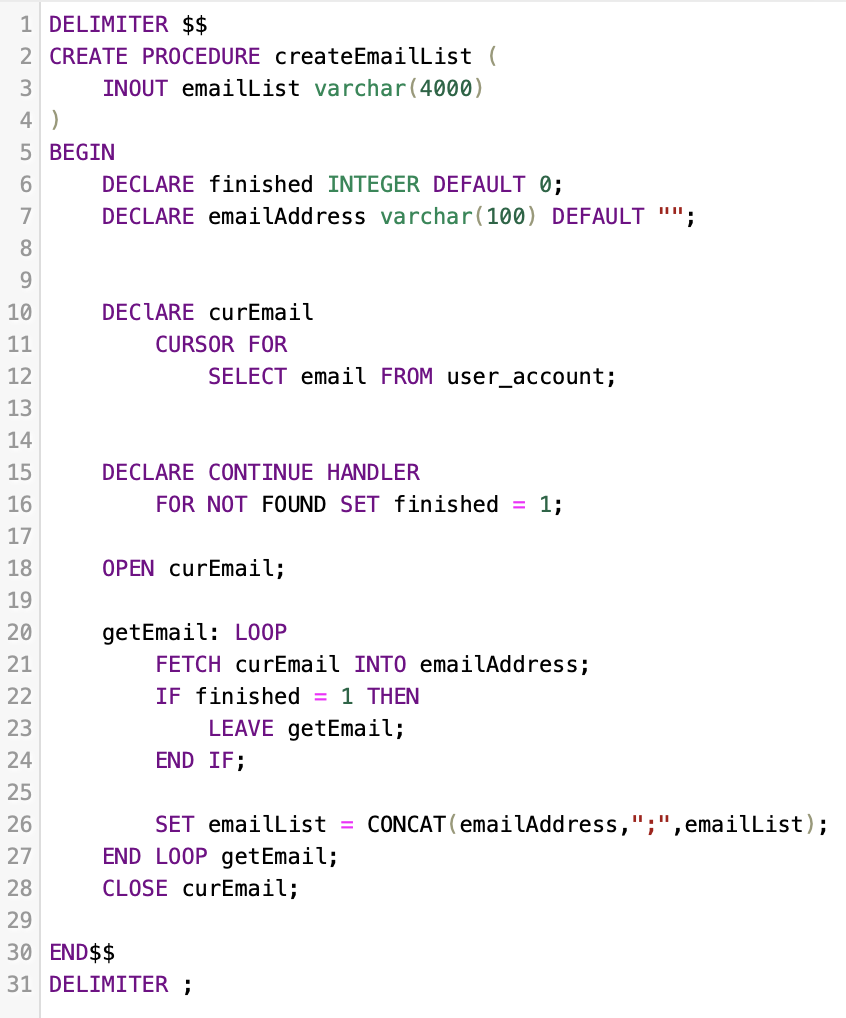
****

**AFTER UPDATION INTEREST OF 0.2 GOT ADDED ACC. TO CONDITION**

****

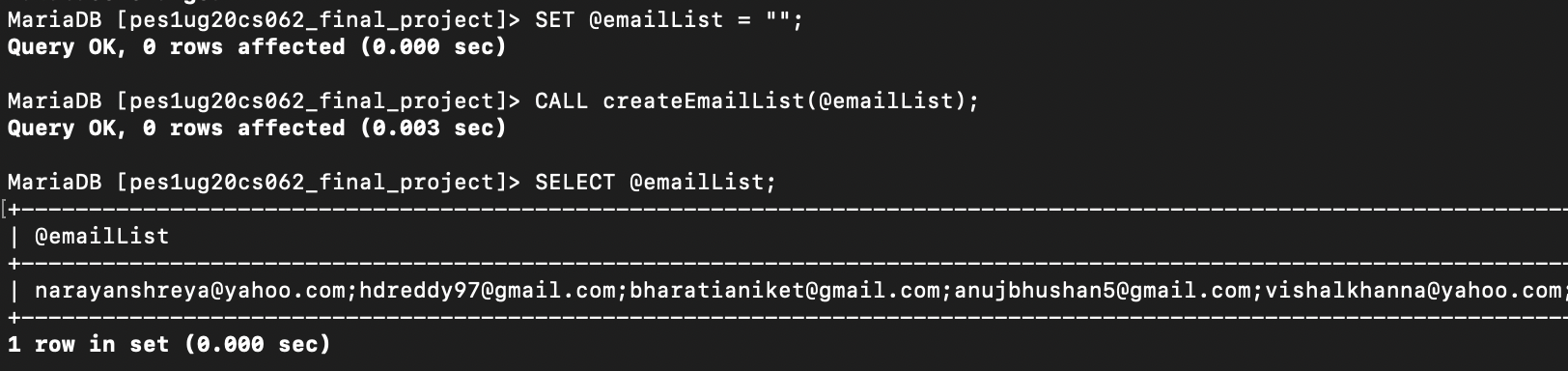
**CURSORS:**

**QUERY:**

****

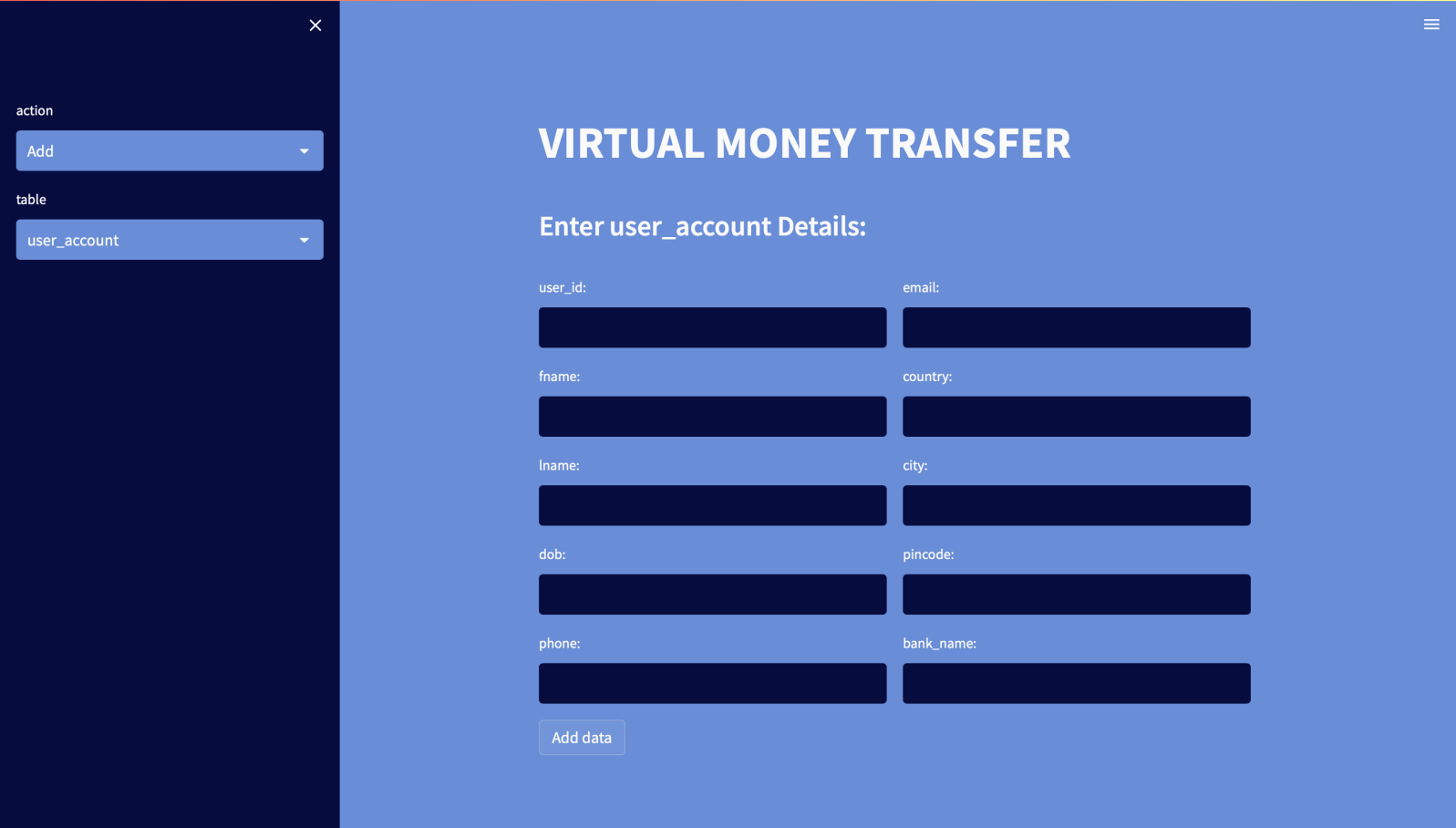
**RESULT:**

**LIST OF ALL EMAILS:**

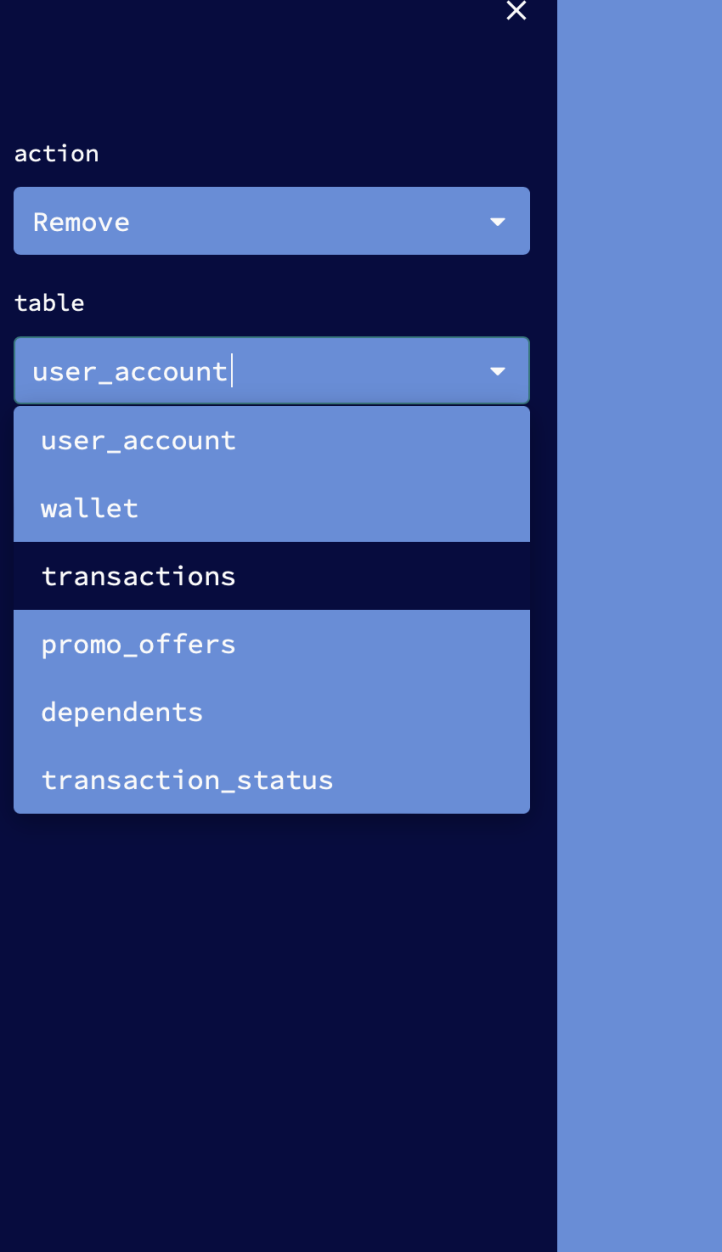
****

**Developing a Frontend:**

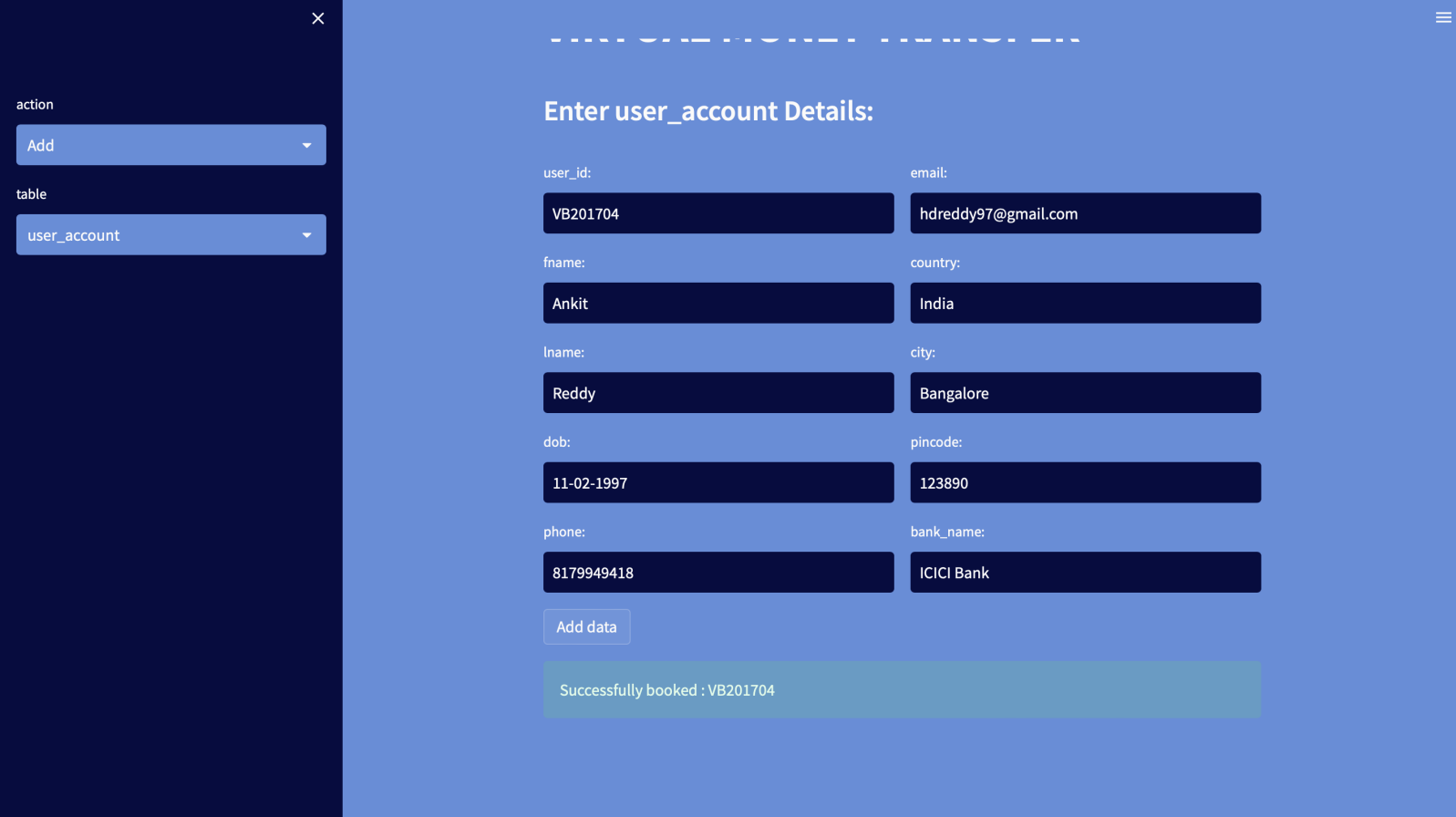
**MAIN PAGE:**

****

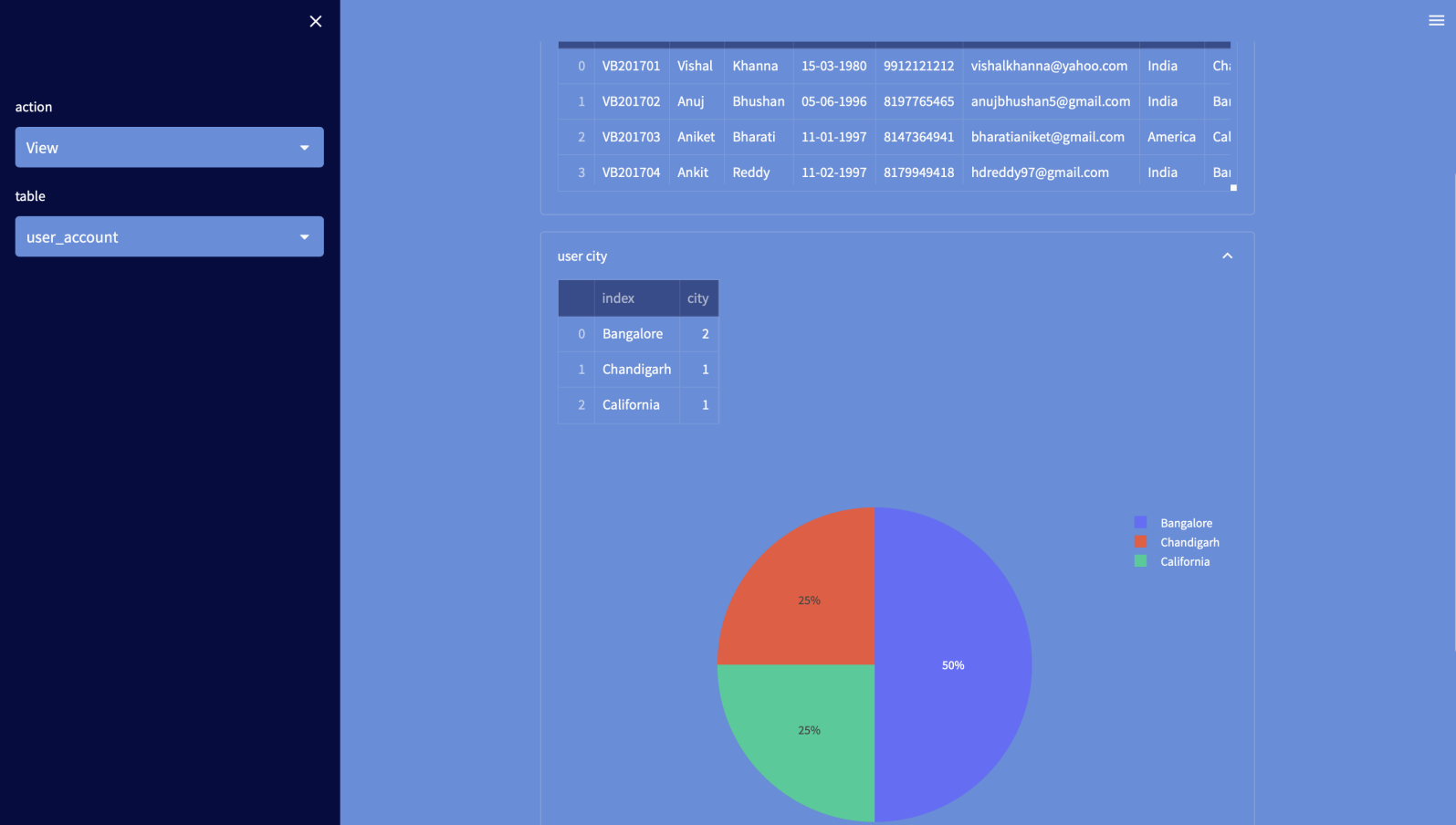
**NAVIGATE THROUGH TABLES:**

****

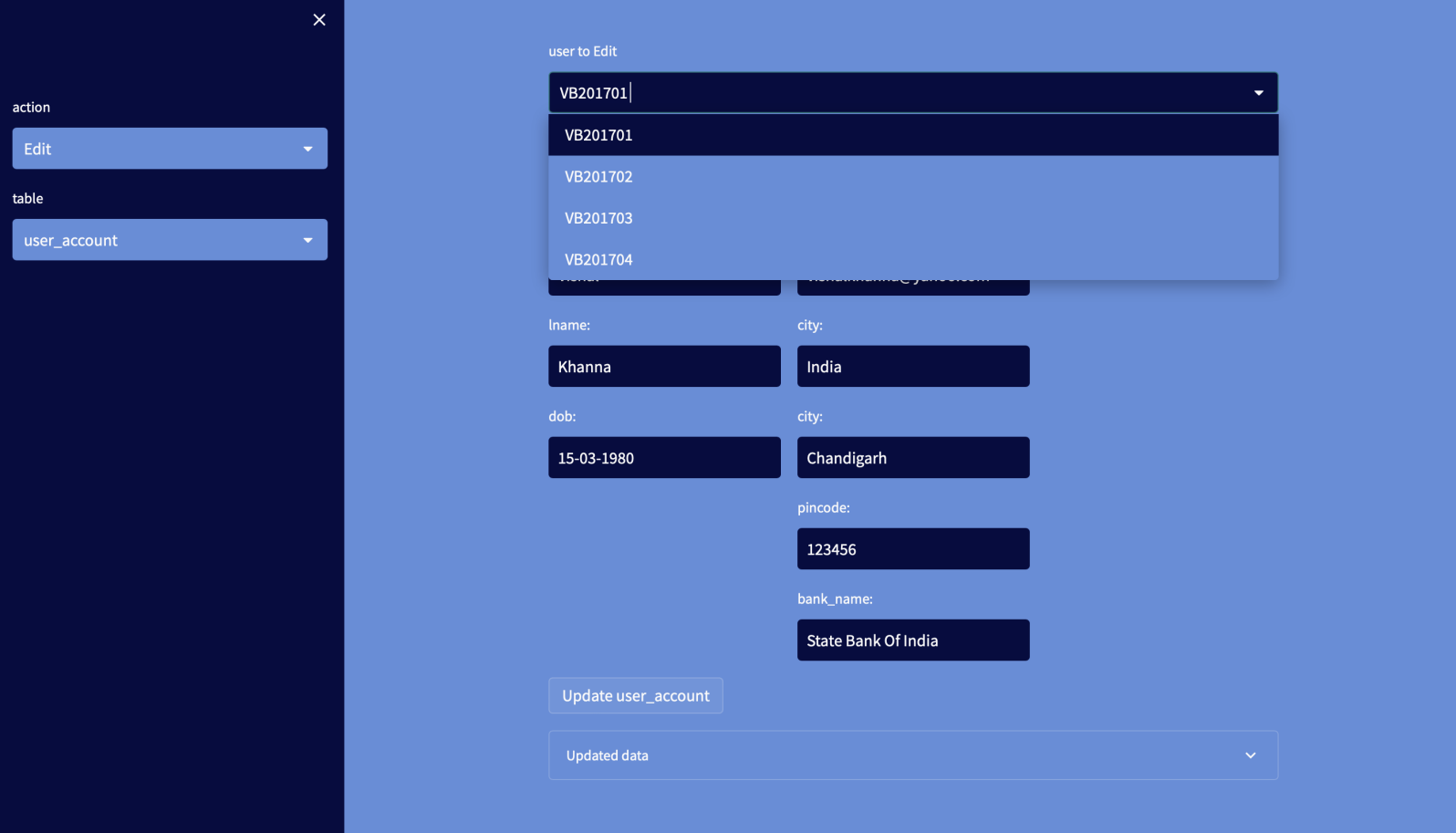
**ADDING VALUES:**

****

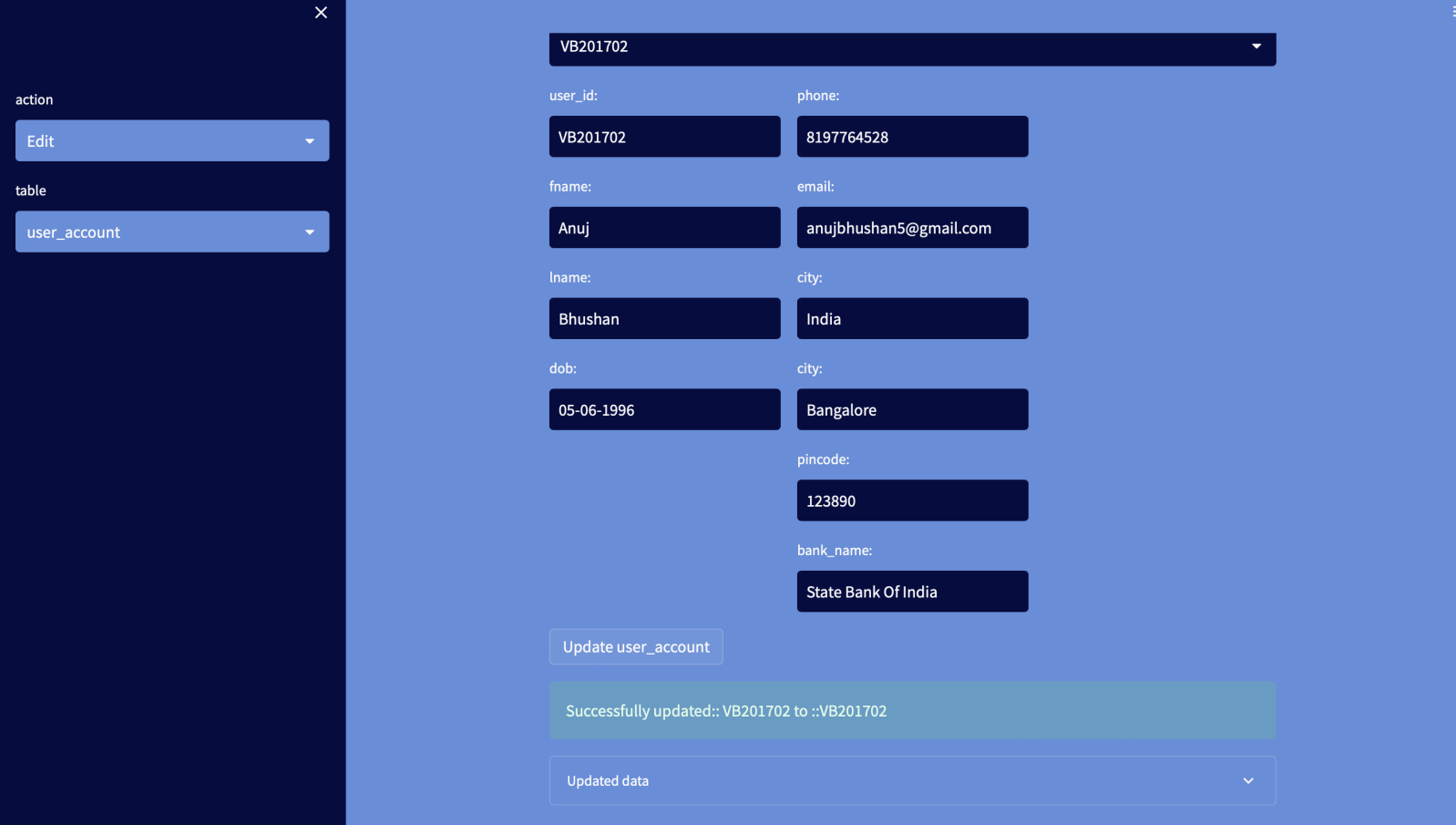
**VIEWING VALUES:**

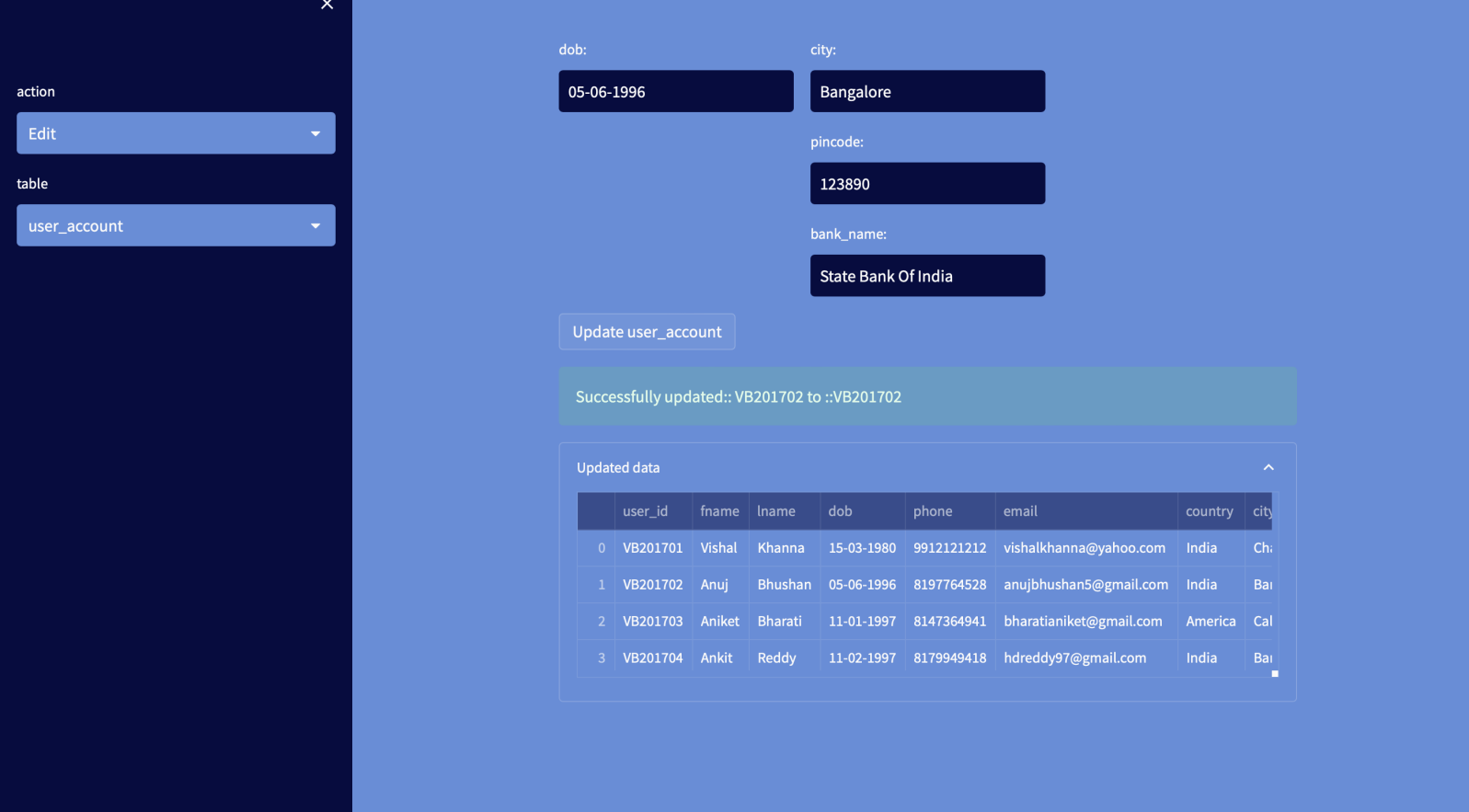
****

**DROP DOWN MENU TO CHOOSE ON WHICH VALUE TO EDIT:**

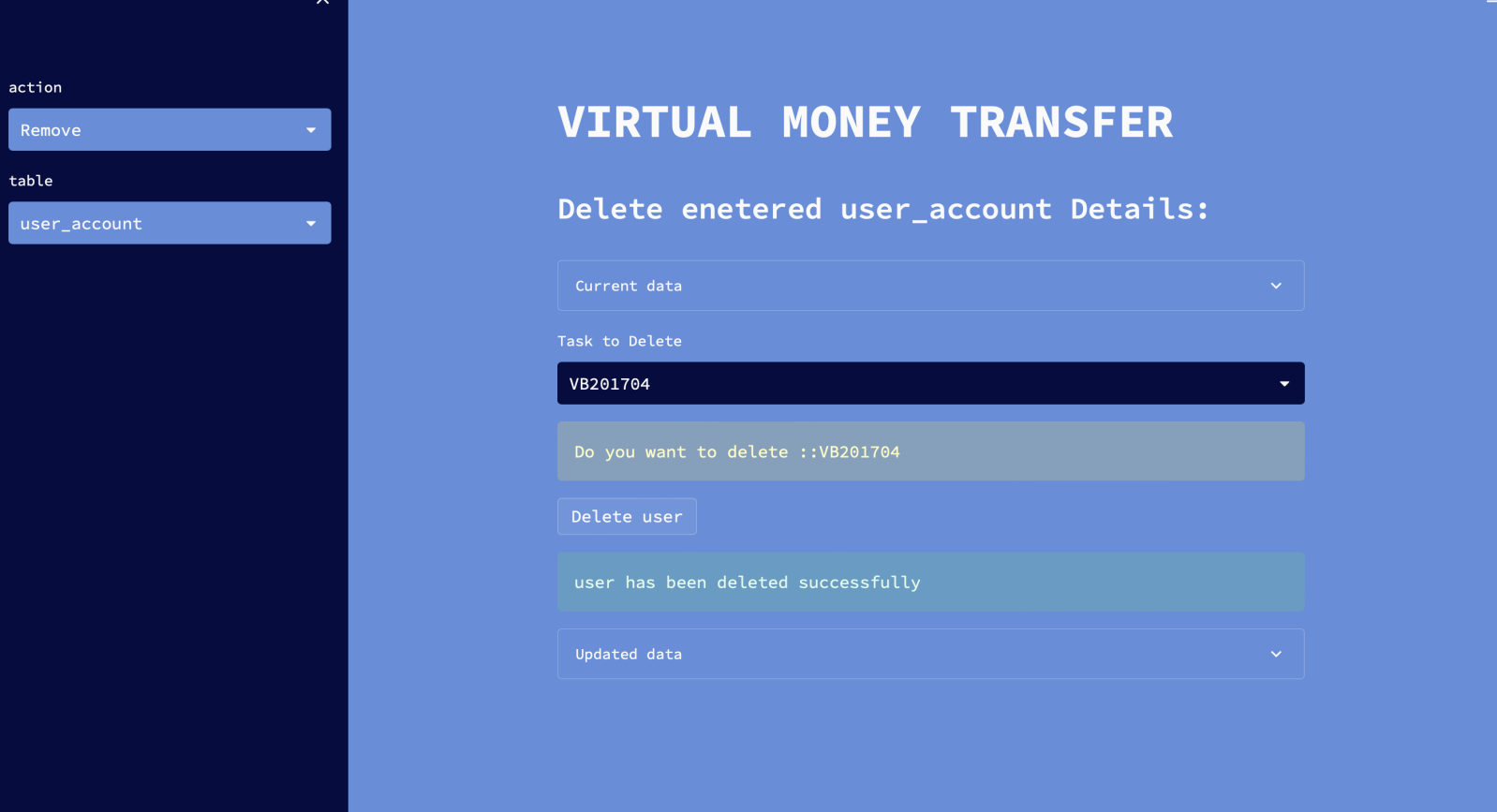
****

**UPDATING THE VALUES:**

****

****

**DELETING THE VALUES:**

****

**CODE:**

**app.py**

# Importing pakages

import streamlit as st

import mysql.connector

from create import create

from database import create\_table

from delete import delete

from read import read

from update import update

mydb = mysql.connector.connect(

host="localhost",

user="root"

)

c = mydb.cursor()

# c.execute("CREATE DATABASE pes1ug20cs062\_final\_project")

c.execute("use pes1ug20cs062\_final\_project")

def main():

st.title("VIRTUAL MONEY TRANSFER")

menu = ["Add", "View", "Edit", "Remove"]

table\_names=["user\_account","wallet","transactions","promo\_offers","dependents","transaction\_status"]

choice = st.sidebar.selectbox("action", menu)

table=st.sidebar.selectbox("table", table\_names)

create\_table(table)

if choice == "Add":

if table=='user\_account':

st.subheader("Enter user\_account Details:")

create(table)

elif table=='wallet':

st.subheader("Enter wallet Details:")

create(table)

elif table=='transactions':

st.subheader("Enter transactions Details:")

create(table)

elif table=='promo\_offers':

st.subheader("Enter promo\_offers Details:")

create(table)

elif table=='dependents':

st.subheader("Enter dependents Details:")

create(table)

elif table=='transaction\_status':

st.subheader("Enter transaction\_status Details:")

create(table)

if choice == "View":

if table=='user\_account':

st.subheader("View entered user\_account Details:")

read(table)

elif table=='wallet':

st.subheader("View entered wallet Details:")

read(table)

elif table=='transactions':

st.subheader("View entered transactions Details:")

read(table)

elif table=='promo\_offers':

st.subheader("View entered promo\_offers Details:")

read(table)

elif table=='dependents':

st.subheader("View entered dependents Details:")

read(table)

elif table=='transaction\_status':

st.subheader("View entered transaction\_status Details:")

read(table)

if choice == "Remove":

if table=='user\_account':

st.subheader("Delete enetered user\_account Details:")

delete(table)

elif table=='wallet':

st.subheader("Delete entered wallet Details:")

delete(table)

elif table=='transactions':

st.subheader("Delete entered transactions Details:")

delete(table)

elif table=='promo\_offers':

st.subheader("Delete entered promo\_offers Details:")

delete(table)

elif table=='dependents':

st.subheader("Delete entered dependents Details:")

delete(table)

elif table=='transaction\_status':

st.subheader("Delete entered transaction\_status Details:")

delete(table)

if choice == "Edit":

if table=='user\_account':

st.subheader("Update entered user\_account Details:")

update(table)

elif table=='wallet':

st.subheader("Update entered wallet Details:")

update(table)

elif table=='transactions':

st.subheader("Update entered transactions Details:")

update(table)

elif table=='promo\_offers':

st.subheader("Update entered promo\_offers Details:")

update(table)

elif table=='dependents':

st.subheader("Update entered dependents Details:")

update(table)

elif table=='transaction\_status':

st.subheader("Update entered transaction\_status Details:")

update(table)

if \_\_name\_\_ == '\_\_main\_\_':

main()

**create.py:**

import streamlit as st

from database import add\_data\_user\_account

from database import add\_data\_wallet

from database import add\_data\_transactions

from database import add\_data\_promo\_offers

from database import add\_data\_dependents

from database import add\_data\_transaction\_status

def create(table):

if table=='user\_account':

col1, col2 = st.columns(2)

with col1:

user\_id = st.text\_input("user\_id:")

fname = st.text\_input("fname:")

lname = st.text\_input("lname:")

dob = st.text\_input("dob:")

phone = st.text\_input("phone:")

with col2:

email = st.text\_input("email:")

country = st.text\_input("country:")

city = st.text\_input("city:")

pincode= st.text\_input("pincode:")

bank\_name = st.text\_input("bank\_name:")

if st.button("Add data"):

add\_data\_user\_account(user\_id,fname,lname ,dob ,phone ,email ,country ,city ,pincode,bank\_name)

st.success("Successfully booked : {}".format(user\_id))

elif table=='wallet':

col1, col2 = st.columns(2)

with col1:

t\_id = st.text\_input("t\_id:")

user\_id = st.text\_input("user\_id:")

account\_no = st.text\_input("account\_no:")

bank\_name = st.text\_input("bank\_name:")

with col2:

balance = st.text\_input("balance:")

promocode = st.text\_input("promocode:")

loan = st.text\_input("loan:")

if st.button("Add data"):

add\_data\_wallet(t\_id,user\_id,account\_no, bank\_name,balance,promocode,loan)

st.success("Successfully added : {}".format(t\_id))

elif table == 'transactions':

col1, col2 = st.columns(2)

with col1:

transaction\_id = st.text\_input("transaction\_id:")

transaction\_date = st.text\_input("transaction\_date:")

transaction\_detail = st.text\_input("transaction\_detail:")

amount = st.text\_input("amount:")

with col2:

to\_id = st.text\_input("to\_id:")

from\_id = st.text\_input("from\_id:")

type\_trans = st.text\_input("type\_trans:")

if st.button("Add data"):

add\_data\_transactions(transaction\_id, transaction\_date,transaction\_detail, amount,to\_id,from\_id,type\_trans)

st.success("Successfully added : {}".format(transaction\_id))

elif table == 'promo\_offers':

col1, col2 = st.columns(2)

with col1:

promo\_id = st.text\_input("promo\_id:")

user\_id = st.text\_input("user\_id:")

start\_date = st.text\_input("start\_date:")

end\_date = st.text\_input("end\_date:")

with col2:

duration = st.text\_input("duration:")

status = st.text\_input("status:")

amount\_value = st.text\_input("amount\_value:")

if st.button("Add data"):

add\_data\_promo\_offers(promo\_id,user\_id,start\_date,end\_date,duration,status, amount\_value)

st.success("Successfully added : {}".format(promo\_id))

elif table == 'dependents':

col1, col2 = st.columns(2)

with col1:

dependent\_id = st.text\_input("dependent\_id:")

trans\_id = st.text\_input("trans\_id:")

user\_ref\_id = st.text\_input("user\_ref\_id:")

fname = st.text\_input("fname:")

lname = st.text\_input("lname:")

with col2:

phone = st.text\_input("phone:")

email = st.text\_input("email:")

dob = st.text\_input("dob:")

relation = st.text\_input("relation:")

if st.button("Add data"):

add\_data\_dependents(dependent\_id,trans\_id,user\_ref\_id,fname,lname,phone,email,dob,relation)

st.success("Successfully added : {}".format(dependent\_id))

elif table == 'transaction\_status':

col1, col2 = st.columns(2)

with col1:

trans\_id = st.text\_input("trans\_id:")

u\_id = st.text\_input("uid:")

status = st.text\_input("status:")

if st.button("Add data"):

add\_data\_transaction\_status(trans\_id,u\_id,status)

st.success("Successfully added : {}".format(trans\_id))

**database.py**

# pip install mysql-connector-python

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="root",

database="pes1ug20cs062\_final\_project"

)

c = mydb.cursor()

def create\_table(table):

if table=='uder\_account':

c.execute('CREATE TABLE IF NOT EXISTS user\_account(user\_id TEXT,fname TEXT,lname TEXT,dob TEXT,phone TEXT,email TEXT,country TEXT,city TEXT,pincode TEXT,bank\_name TEXT)')

elif table=='wallet':

c.execute('CREATE TABLE IF NOT EXISTS wallet(t\_id TEXT ,user\_id TEXT,account\_no TEXT, bank\_name TEXT,balance TEXT,promocode TEXT,loan TEXT)')

elif table=='transactions':

c.execute('CREATE TABLE IF NOT EXISTS transactions(transaction\_id TEXT, transaction\_date TEXT,transaction\_detail TEXT, amount TEXT,to\_id TEXT,from\_id TEXT,type\_trans TEXT)')

elif table=='promo\_offers':

c.execute('CREATE TABLE IF NOT EXISTS promo\_offers(promo\_id TEXT,user\_id TEXT,start\_date TEXT,end\_date TEXT,duration TEXT,status TEXT, amount\_value TEXT)')

elif table=='dependents':

c.execute('CREATE TABLE IF NOT EXISTS dependents(dependent\_id TEXT,trans\_id TEXT,user\_ref\_id TEXT,fname TEXT,lname TEXT,phone TEXT,email TEXT,dob TEXT,relation TEXT)')

elif table=='transaction\_status':

c.execute('CREATE TABLE IF NOT EXISTS transaction\_status(trans\_id TEXT,u\_id TEXT,status TEXT)')

def add\_data\_user\_account(user\_id,fname,lname ,dob ,phone ,email ,country ,city ,pincode,bank\_name):

c.execute('INSERT INTO user\_account(user\_id,fname,lname ,dob ,phone ,email ,country ,city ,pincode,bank\_name) VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)',

(user\_id,fname,lname ,dob ,phone ,email ,country ,city ,pincode,bank\_name))

mydb.commit()

def add\_data\_wallet(t\_id,user\_id,account\_no, bank\_name,balance,promocode,loan):

c.execute('INSERT INTO wallet(t\_id,user\_id,account\_no, bank\_name,balance,promocode,loan) VALUES (%s,%s,%s,%s,%s,%s,%s)',

(t\_id,user\_id,account\_no, bank\_name,balance,promocode,loan))

mydb.commit()

def add\_data\_transactions(transaction\_id, transaction\_date,transaction\_detail, amount,to\_id,from\_id,type\_trans):

c.execute('INSERT INTO transactions(transaction\_id, transaction\_date,transaction\_detail, amount,to\_id,from\_id,type\_trans) VALUES (%s,%s,%s,%s,%s,%s,%s)',

(transaction\_id, transaction\_date,transaction\_detail, amount,to\_id,from\_id,type\_trans))

mydb.commit()

def add\_data\_promo\_offers(promo\_id,user\_id,start\_date,end\_date,duration,status, amount\_value):

c.execute('INSERT INTO promo\_offers(promo\_id,user\_id,start\_date,end\_date,duration,status, amount\_value) VALUES (%s,%s,%s,%s,%s,%s,%s)',

(promo\_id,user\_id,start\_date,end\_date,duration,status, amount\_value))

mydb.commit()

def add\_data\_dependents(dependent\_id,trans\_id,user\_ref\_id,fname,lname,phone ,email ,dob,relation):

c.execute('INSERT INTO dependents(dependent\_id,trans\_id,user\_ref\_id,fname,lname,phone ,email ,dob,relation) VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s)',

(dependent\_id,trans\_id,user\_ref\_id,fname,lname,phone ,email ,dob,relation))

mydb.commit()

def add\_data\_transaction\_status(trans\_id,u\_id,status):

c.execute('INSERT INTO transaction\_status(trans\_id,u\_id,status) VALUES (%s,%s,%s)',

(trans\_id,u\_id,status))

mydb.commit()

#view tables

def view\_all\_user\_account():

c.execute('SELECT \* FROM user\_account')

data = c.fetchall()

return data

def view\_all\_wallet():

c.execute('SELECT \* FROM wallet')

data = c.fetchall()

return data

def view\_all\_data\_transactions():

c.execute('SELECT \* FROM transactions')

data = c.fetchall()

return data

def view\_all\_data\_promo\_offers():

c.execute('SELECT \* FROM promo\_offers')

data = c.fetchall()

return data

def view\_all\_data\_dependents():

c.execute('SELECT \* FROM dependents')

data = c.fetchall()

return data

def view\_all\_data\_transaction\_status():

c.execute('SELECT \* FROM transaction\_status')

data = c.fetchall()

return data

#viewonly tables

def view\_only\_user\_account():

c.execute('SELECT user\_id FROM user\_account')

data = c.fetchall()

return data

def view\_only\_wallet():

c.execute('SELECT t\_id FROM wallet')

data = c.fetchall()

return data

def view\_only\_data\_transactions():

c.execute('SELECT transaction\_id FROM transactions')

data = c.fetchall()

return data

def view\_only\_data\_promo\_offers():

c.execute('SELECT promo\_id FROM promo\_offers')

data = c.fetchall()

return data

def view\_only\_data\_dependents():

c.execute('SELECT dependent\_id FROM dependents')

data = c.fetchall()

return data

def view\_only\_data\_transaction\_status():

c.execute('SELECT trans\_id FROM transaction\_status')

data = c.fetchall()

return data

#getting

def get\_user\_id(user\_id):

c.execute('SELECT \* FROM user\_account WHERE user\_id="{}"'.format(user\_id))

data = c.fetchall()

return data

def get\_tid(t\_id):

c.execute('SELECT \* FROM wallet WHERE t\_id="{}"'.format(t\_id))

data = c.fetchall()

return data

def get\_transaction\_id(transaction\_id):

c.execute('SELECT \* FROM transactions WHERE transaction\_id="{}"'.format(transaction\_id))

data = c.fetchall()

return data

def get\_promo\_id(promo\_id):

c.execute('SELECT \* FROM promo\_offers WHERE promo\_id="{}"'.format(promo\_id))

data = c.fetchall()

return data

def get\_dependent\_id(dependent\_id):

c.execute('SELECT \* FROM dependents WHERE dependent\_id="{}"'.format(dependent\_id))

data = c.fetchall()

return data

def get\_trans\_id(trans\_id):

c.execute('SELECT \* FROM transaction\_status WHERE trans\_id="{}"'.format(trans\_id))

data = c.fetchall()

return data

#editig

def edit\_user\_account\_data(new\_user\_id,new\_fname,new\_lname ,new\_dob ,new\_phone ,new\_email ,new\_country ,new\_city ,new\_pincode,new\_bank\_name,user\_id,fname,lname ,dob ,phone ,email ,country ,city ,pincode,bank\_name):

c.execute("UPDATE user\_account SET user\_id=%s,fname=%s, lname=%s, dob=%s, phone=%s,email=%s,country=%s,city=%s,pincode=%s,bank\_name=%s WHERE "

"user\_id=%s and fname=%s and lname=%s and dob=%s and phone=%s and email=%s and country=%s and city=%s and pincode=%s and bank\_name=%s",(new\_user\_id,new\_fname,new\_lname ,new\_dob ,new\_phone ,new\_email ,new\_country ,new\_city ,new\_pincode,new\_bank\_name,user\_id,fname,lname ,dob ,phone ,email ,country ,city ,pincode,bank\_name))

mydb.commit()

data = c.fetchall()

return data

def edit\_wallet\_data(new\_t\_id,new\_user\_id,new\_account\_no, new\_bank\_name,new\_balance,new\_promocode,new\_loan,t\_id,user\_id,account\_no, bank\_name,balance,promocode,loan):

c.execute("UPDATE wallet SET t\_id=%s, user\_id=%s, account\_no=%s, bank\_name=%s,balance=%s,promocode=%s,loan=%s WHERE "

"t\_id=%s and user\_id=%s and account\_no=%s and bank\_name=%s and balance=%s and promocode=%s and loan=%s", (new\_t\_id,new\_user\_id,new\_account\_no, new\_bank\_name,new\_balance,new\_promocode,new\_loan,t\_id,user\_id,account\_no, bank\_name,balance,promocode,loan))

mydb.commit()

data = c.fetchall()

return data

def edit\_transactions\_data(new\_transaction\_id, new\_transaction\_date,new\_transaction\_detail, new\_amount,new\_to\_id,new\_from\_id,new\_type\_trans,transaction\_id, transaction\_date,transaction\_detail, amount,to\_id,from\_id,type\_trans):

c.execute("UPDATE transactions SET transaction\_id=%s, transaction\_date=%s, transaction\_detail=%s, amount=%s, to\_id=%s, from\_id=%s, type\_trans=%s WHERE "

"transaction\_id=%s and transaction\_date=%s and transaction\_detail=%s and amount=%s and to\_id=%s and from\_id=%s and type\_trans=%s ", (new\_transaction\_id, new\_transaction\_date,new\_transaction\_detail, new\_amount,new\_to\_id,new\_from\_id,new\_type\_trans,transaction\_id, transaction\_date,transaction\_detail, amount,to\_id,from\_id,type\_trans))

mydb.commit()

data = c.fetchall()

return data

def edit\_promo\_offers\_data(new\_promo\_id,new\_user\_id,new\_start\_date,new\_end\_date,new\_duration,new\_status, new\_amount\_value,promo\_id,user\_id,start\_date,end\_date,duration,status, amount\_value):

c.execute("UPDATE promo\_offers SET promo\_id=%s, user\_id=%s, start\_date=%s,end\_date=%s, duration=%s, status=%s, amount\_value=%s WHERE "

"promo\_id=%s and user\_id=%s and start\_date=%s and end\_date=%s and duration=%s and status=%s and amount\_value=%s ", (new\_promo\_id,new\_user\_id,new\_start\_date,new\_end\_date,new\_duration,new\_status, new\_amount\_value,promo\_id,user\_id,start\_date,end\_date,duration,status, amount\_value))

mydb.commit()

data = c.fetchall()

return data

def edit\_dependents\_data(new\_dependent\_id,new\_trans\_id,new\_user\_ref\_id,new\_fname,new\_lname,new\_phone,new\_email,new\_dob,new\_relation,dependent\_id,trans\_id,user\_ref\_id,fname,lname,phone,email,dob,relation):

c.execute("UPDATE dependents SET dependent\_id=%s, trans\_id=%s, user\_ref\_id=%s, fname=%s,lname=%s, phone=%s, email=%s, dob=%s, relation=%s WHERE "

"dependent\_id=%s and trans\_id=%s and user\_ref\_id=%s and fname=%s and lname=%s and phone=%s and email=%s and dob=%s and relation=%s ", (new\_dependent\_id,new\_trans\_id,new\_user\_ref\_id,new\_fname,new\_lname,new\_phone,new\_email,new\_dob,new\_relation,dependent\_id,trans\_id,user\_ref\_id,fname,lname,phone,email,dob,relation))

mydb.commit()

data = c.fetchall()

return data

def edit\_transaction\_status\_data(new\_trans\_id,new\_u\_id,new\_status,trans\_id,u\_id,status):

c.execute("UPDATE transaction\_status SET trans\_id=%s, u\_id=%s, status=%s WHERE "

"trans\_id=%s and u\_id=%s and status=%s ", (new\_trans\_id,new\_u\_id,new\_status,trans\_id,u\_id,status))

mydb.commit()

data = c.fetchall()

return data

#delete

def delete\_user\_account(user\_id):

c.execute('DELETE FROM user\_account WHERE user\_id="{}"'.format(user\_id))

mydb.commit()

def delete\_wallet(t\_id):

c.execute('DELETE FROM wallet WHERE t\_id="{}"'.format(t\_id))

mydb.commit()

def delete\_transactions(transaction\_id):

c.execute('DELETE FROM transactions WHERE transaction\_id="{}"'.format(transaction\_id))

mydb.commit()

def delete\_promo\_offers(promo\_id):

c.execute('DELETE FROM promo\_offers WHERE promo\_id="{}"'.format(promo\_id))

mydb.commit()

def delete\_dependents(dependent\_id):

c.execute('DELETE FROM dependents WHERE dependent\_id="{}"'.format(dependent\_id))

mydb.commit()

def delete\_transaction\_status(trans\_id):

c.execute('DELETE FROM transaction\_status WHERE trans\_id="{}"'.format(trans\_id))

mydb.commit()

**update.py**

import datetime

import pandas as pd

import streamlit as st

from database import view\_all\_user\_account

from database import view\_all\_wallet

from database import view\_all\_data\_transactions

from database import view\_all\_data\_promo\_offers

from database import view\_all\_data\_dependents

from database import view\_all\_data\_transaction\_status

from database import view\_only\_user\_account

from database import view\_only\_wallet

from database import view\_only\_data\_transactions

from database import view\_only\_data\_promo\_offers

from database import view\_only\_data\_dependents

from database import view\_only\_data\_transaction\_status

from database import get\_user\_id

from database import get\_tid

from database import get\_transaction\_id

from database import get\_promo\_id

from database import get\_dependent\_id

from database import get\_trans\_id

from database import edit\_user\_account\_data

from database import edit\_wallet\_data

from database import edit\_transactions\_data

from database import edit\_promo\_offers\_data

from database import edit\_dependents\_data

from database import edit\_transaction\_status\_data

def update(table):

if table=='user\_account':

result = view\_all\_user\_account()

# st.write(result)

df = pd.DataFrame(result, columns=['user\_id','fname','lname' ,'dob' ,'phone' ,'email' ,'country' ,'city' ,'pincode','bank\_name'])

with st.expander("Current user\_accounts"):

st.dataframe(df)

list\_of\_dealers = [i[0] for i in view\_only\_user\_account()]

selected\_dealer = st.selectbox("user to Edit", list\_of\_dealers)

selected\_result = get\_user\_id(selected\_dealer)

# st.write(selected\_result)

if selected\_result:

user\_id = selected\_result[0][0]

fname = selected\_result[0][1]

lname = selected\_result[0][2]

dob = selected\_result[0][3]

phone = selected\_result[0][4]

email = selected\_result[0][5]

country = selected\_result[0][6]

city = selected\_result[0][7]

pincode = selected\_result[0][8]

bank\_name = selected\_result[0][9]

# Layout of Create

col1, col2 ,col3= st.columns(3)

with col1:

new\_user\_id = st.text\_input("user\_id:",user\_id)

new\_fname = st.text\_input("fname:", fname)

new\_lname = st.text\_input("lname:", lname)

new\_dob = st.text\_input("dob:", dob)

with col2:

new\_phone = st.text\_input("phone:",phone)

new\_email = st.text\_input("email:",email)

new\_country = st.text\_input("city:",country)

new\_city = st.text\_input("city:",city)

new\_pincode = st.text\_input("pincode:",pincode)

new\_bank\_name = st.text\_input("bank\_name:",bank\_name)

if st.button("Update user\_account"):

edit\_user\_account\_data(new\_user\_id,new\_fname,new\_lname ,new\_dob ,new\_phone ,new\_email ,new\_country ,new\_city ,new\_pincode,new\_bank\_name,user\_id,fname,lname ,dob ,phone ,email ,country ,city ,pincode,bank\_name)

st.success("Successfully updated:: {} to ::{}".format(user\_id, new\_user\_id))

result2 = view\_all\_user\_account()

df2 = pd.DataFrame(result2, columns=['user\_id','fname','lname' ,'dob' ,'phone' ,'email' ,'country' ,'city' ,'pincode','bank\_name'])

with st.expander("Updated data"):

st.dataframe(df2)

elif table=='wallet':

result = view\_all\_wallet()

# st.write(result)

df = pd.DataFrame(result, columns=['t\_id','user\_id','account\_no', 'bank\_name','balance','promocode','loan'])

with st.expander("Current wallets"):

st.dataframe(df)

list\_of\_dealers = [i[0] for i in view\_only\_wallet()]

selected\_dealer = st.selectbox("user to Edit", list\_of\_dealers)

selected\_result = get\_tid(selected\_dealer)

# st.write(selected\_result)

if selected\_result:

t\_id = selected\_result[0][0]

user\_id = selected\_result[0][1]

account\_no = selected\_result[0][2]

bank\_name = selected\_result[0][3]

balance = selected\_result[0][4]

promocode = selected\_result[0][5]

loan = selected\_result[0][6]

# Layout of Create

col1, col2 ,col3= st.columns(3)

with col1:

new\_t\_id = st.text\_input("t\_id:",t\_id)

new\_user\_id = st.text\_input("user\_id:", user\_id)

new\_account\_no = st.text\_input("account\_no:", account\_no)

new\_bank\_name = st.text\_input("bank\_name:", bank\_name)

with col2:

new\_balance = st.text\_input("balance:",balance)

new\_promocode = st.text\_input("promocode:",promocode)

new\_loan = st.text\_input("loan:",loan)

if st.button("Update book"):

edit\_wallet\_data(new\_t\_id,new\_user\_id,new\_account\_no, new\_bank\_name,new\_balance,new\_promocode,new\_loan,t\_id,user\_id,account\_no, bank\_name,balance,promocode,loan)

st.success("Successfully updated:: {} to ::{}".format(t\_id, new\_t\_id))

result2 = view\_all\_wallet()

df2 = pd.DataFrame(result2, columns=['tid','user\_id','account\_no', 'bank\_name','balance','promocode','loan'])

with st.expander("Updated data"):

st.dataframe(df2)

elif table=='transactions':

result = view\_all\_data\_transactions()

# st.write(result)

df = pd.DataFrame(result, columns=['transaction\_id', 'transaction\_date','transaction\_detail', 'amount','to\_id','from\_id','type\_trans'])

with st.expander("Current transactions"):

st.dataframe(df)

list\_of\_dealers = [i[0] for i in view\_only\_data\_transactions()]

selected\_dealer = st.selectbox("transactions to Edit", list\_of\_dealers)

selected\_result = get\_transaction\_id(selected\_dealer)

# st.write(selected\_result)

if selected\_result:

transaction\_id = selected\_result[0][0]

transaction\_date = selected\_result[0][1]

transaction\_detail = selected\_result[0][2]

amount = selected\_result[0][3]

to\_id = selected\_result[0][4]

from\_id = selected\_result[0][5]

type\_trans = selected\_result[0][6]

# Layout of Create

col1, col2 = st.columns(2)

with col1:

new\_transaction\_id = st.text\_input("transaction\_id:",transaction\_id)

new\_transaction\_date = st.text\_input("transaction\_date:", transaction\_date)

new\_transaction\_detail = st.text\_input("transaction\_detail:", transaction\_detail)

new\_amount = st.text\_input("amount:", amount)

with col2:

new\_to\_id = st.text\_input("to\_id:",to\_id)

new\_from\_id = st.text\_input("from\_id:",from\_id)

new\_type\_trans = st.text\_input("type\_trans:",type\_trans)

if st.button("Update book"):

edit\_transactions\_data(new\_transaction\_id, new\_transaction\_date,new\_transaction\_detail, new\_amount,new\_to\_id,new\_from\_id,new\_type\_trans,transaction\_id, transaction\_date,transaction\_detail, amount,to\_id,from\_id,type\_trans)

st.success("Successfully updated:: {} to ::{}".format(transaction\_id, new\_transaction\_id))

result2 = view\_all\_data\_transactions()

df2 = pd.DataFrame(result2, columns=['transaction\_id', 'transaction\_date','transaction\_detail', 'amount','to\_id','from\_id','type\_trans'])

with st.expander("Updated data"):

st.dataframe(df2)

elif table=='promo\_offers':

result = view\_all\_data\_promo\_offers()

# st.write(result)

df = pd.DataFrame(result, columns=['promo\_id','user\_id','start\_date','end\_date','duration','status', 'amount\_value'])

with st.expander("Current bookings"):

st.dataframe(df)

list\_of\_dealers = [i[0] for i in view\_only\_data\_promo\_offers()]

selected\_dealer = st.selectbox("user to Edit", list\_of\_dealers)

selected\_result = get\_promo\_id(selected\_dealer)

# st.write(selected\_result)

if selected\_result:

promo\_id = selected\_result[0][0]

user\_id = selected\_result[0][1]

start\_date = selected\_result[0][2]

end\_date = selected\_result[0][3]

duration = selected\_result[0][4]

status = selected\_result[0][5]

amount\_value = selected\_result[0][6]

# Layout of Create

col1, col2 ,col3= st.columns(3)

with col1:

new\_promo\_id = st.text\_input("promo\_id:",promo\_id)

new\_user\_id = st.text\_input("user\_id:", user\_id)

new\_start\_date = st.text\_input("start\_date:", start\_date)

with col2:

new\_end\_date = st.text\_input("end\_date:", end\_date)

new\_duration = st.text\_input("duration:",duration)

new\_status = st.text\_input("status:",status)

new\_amount\_value = st.text\_input("amount\_value:",amount\_value)

if st.button("Update book"):

edit\_promo\_offers\_data(new\_promo\_id,new\_user\_id,new\_start\_date,new\_end\_date,new\_duration,new\_status, new\_amount\_value,promo\_id,user\_id,start\_date,end\_date,duration,status, amount\_value)

st.success("Successfully updated:: {} to ::{}".format(promo\_id, new\_promo\_id))

result2 = view\_all\_data\_promo\_offers()

df2 = pd.DataFrame(result2, columns=['promo\_id','user\_id','start\_date','end\_date','duration','status', 'amount\_value'])

with st.expander("Updated data"):

st.dataframe(df2)

elif table=='dependents':

result = view\_all\_data\_dependents()

# st.write(result)

df = pd.DataFrame(result, columns=['dependent\_id','trans\_id','user\_ref\_id','fname','lname','phone' ,'email' ,'dob','relation'])

with st.expander("Current bookings"):

st.dataframe(df)

list\_of\_dealers = [i[0] for i in view\_only\_data\_dependents()]

selected\_dealer = st.selectbox("user to Edit", list\_of\_dealers)

selected\_result = get\_dependent\_id(selected\_dealer)

# st.write(selected\_result)

if selected\_result:

dependent\_id = selected\_result[0][0]

trans\_id = selected\_result[0][1]

user\_ref\_id = selected\_result[0][2]

fname = selected\_result[0][3]

lname = selected\_result[0][4]

phone = selected\_result[0][5]

email = selected\_result[0][6]

dob = selected\_result[0][7]

relation = selected\_result[0][8]

# Layout of Create

col1, col2 ,col3= st.columns(3)

with col1:

new\_dependent\_id = st.text\_input("dependent\_id:",dependent\_id)

new\_trans\_id = st.text\_input("trans\_id:", trans\_id)

new\_user\_ref\_id = st.text\_input("user\_ref\_id:", user\_ref\_id)

new\_fname = st.text\_input("fname:", fname)

with col2:

new\_lname = st.text\_input("lname:",lname)

new\_phone = st.text\_input("phone:",phone)

new\_email = st.text\_input("email:",email)

new\_dob = st.text\_input("dob:",dob)

new\_relation = st.text\_input("relation:",relation)

if st.button("Update book"):

edit\_dependents\_data(new\_dependent\_id,new\_trans\_id,new\_user\_ref\_id,new\_fname,new\_lname,new\_phone,new\_email,new\_dob,new\_relation,dependent\_id,trans\_id,user\_ref\_id,fname,lname,phone,email,dob,relation)

st.success("Successfully updated:: {} to ::{}".format(dependent\_id, new\_dependent\_id))

result2 = view\_all\_data\_dependents()

df2 = pd.DataFrame(result2, columns=['dependent\_id','trans\_id','user\_ref\_id','fname','lname','phone' ,'email' ,'dob','relation'])

with st.expander("Updated data"):

st.dataframe(df2)

elif table=='transaction\_status':

result = view\_all\_data\_transaction\_status()

# st.write(result)

df = pd.DataFrame(result, columns=['trans\_id','u\_id','status'])

with st.expander("Current bookings"):

st.dataframe(df)

list\_of\_dealers = [i[0] for i in view\_only\_data\_transaction\_status()]

selected\_dealer = st.selectbox("user to Edit", list\_of\_dealers)

selected\_result = get\_trans\_id(selected\_dealer)

# st.write(selected\_result)

if selected\_result:

trans\_id = selected\_result[0][0]

u\_id = selected\_result[0][1]

status = selected\_result[0][2]

# Layout of Create

col1, col2 = st.columns(2)

with col1:

new\_trans\_id = st.text\_input("trans\_id:",trans\_id)

new\_u\_id = st.text\_input("u\_id:",u\_id)

with col2:

new\_status = st.text\_input("status:",status)

if st.button("Update book"):

edit\_transaction\_status\_data(new\_trans\_id,new\_u\_id,new\_status,trans\_id,u\_id,status)

st.success("Successfully updated:: {} to ::{}".format(trans\_id, new\_trans\_id))

result2 = view\_all\_data\_transaction\_status()

df2 = pd.DataFrame(result2, columns=['trans\_id','u\_id','status'])

with st.expander("Updated data"):

st.dataframe(df2)

**delete.py**

import pandas as pd

import streamlit as st

from database import view\_only\_user\_account

from database import view\_only\_wallet

from database import view\_only\_data\_transactions

from database import view\_only\_data\_promo\_offers

from database import view\_only\_data\_dependents

from database import view\_only\_data\_transaction\_status

from database import view\_all\_user\_account

from database import view\_all\_wallet

from database import view\_all\_data\_transactions

from database import view\_all\_data\_promo\_offers

from database import view\_all\_data\_dependents

from database import view\_all\_data\_transaction\_status

from database import delete\_user\_account

from database import delete\_wallet

from database import delete\_transactions

from database import delete\_promo\_offers

from database import delete\_dependents

from database import delete\_transaction\_status

def delete(table):

if table=='user\_account':

result = view\_all\_user\_account()

df = pd.DataFrame(result, columns=['user\_id','fname','lname' ,'dob' ,'phone' ,'email' ,'country' ,'city' ,'pincode','bank\_name'])

with st.expander("Current data"):

st.dataframe(df)

list\_of\_user = [i[0] for i in view\_only\_user\_account()]

selected\_user = st.selectbox("Task to Delete", list\_of\_user)

st.warning("Do you want to delete ::{}".format(selected\_user))

if st.button("Delete user"):

delete\_user\_account(selected\_user)

st.success("user has been deleted successfully")

new\_result = view\_all\_user\_account()

df2 = pd.DataFrame(new\_result, columns=['user\_id','fname','lname' ,'dob' ,'phone' ,'email' ,'country' ,'city' ,'pincode','bank\_name'])

with st.expander("Updated data"):

st.dataframe(df2)

elif table=='wallet':

result = view\_all\_wallet()

df = pd.DataFrame(result, columns=['t\_id','user\_id','account\_no', 'bank\_name','balance','promocode','loan'])

with st.expander("Current data"):

st.dataframe(df)

list\_of\_user = [i[0] for i in view\_only\_wallet()]

selected\_user = st.selectbox("Task to Delete", list\_of\_user)

st.warning("Do you want to delete ::{}".format(selected\_user))

if st.button("Delete wallet"):

delete\_wallet(selected\_user)

st.success("Wallet has been deleted successfully")

new\_result = view\_all\_wallet()

df2 = pd.DataFrame(new\_result, columns=['t\_id','user\_id','account\_no', 'bank\_name','balance','promocode','loan'])

with st.expander("Updated data"):

st.dataframe(df2)

elif table=='transactions':

result = view\_all\_data\_transactions()

df = pd.DataFrame(result, columns=['transaction\_id', 'transaction\_date','transaction\_detail', 'amount','to\_id','from\_id','type\_trans'])

with st.expander("Current data"):

st.dataframe(df)

list\_of\_user = [i[0] for i in view\_only\_data\_transactions()]

selected\_user = st.selectbox("Task to Delete", list\_of\_user)

st.warning("Do you want to delete ::{}".format(selected\_user))

if st.button("Delete transactions"):

delete\_transactions(selected\_user)

st.success("Transaction has been deleted successfully")

new\_result = view\_all\_wallet()

df2 = pd.DataFrame(new\_result, columns=['transaction\_id', 'transaction\_date','transaction\_detail', 'amount','to\_id','from\_id','type\_trans'])

with st.expander("Updated data"):

st.dataframe(df2)

elif table=='promo\_offers':

result = view\_all\_data\_transactions()

df = pd.DataFrame(result, columns=['promo\_id','user\_id','start\_date','end\_date','duration','status', 'amount\_value'])

with st.expander("Current data"):

st.dataframe(df)

list\_of\_user = [i[0] for i in view\_only\_data\_promo\_offers()]

selected\_user = st.selectbox("Task to Delete", list\_of\_user)

st.warning("Do you want to delete ::{}".format(selected\_user))

if st.button("Delete promo\_offers"):

delete\_promo\_offers(selected\_user)

st.success("user has been deleted successfully")

new\_result = view\_all\_data\_promo\_offers()

df2 = pd.DataFrame(new\_result, columns=['promo\_id','user\_id','start\_date','end\_date','duration','status', 'amount\_value'])

with st.expander("Updated data"):

st.dataframe(df2)

elif table=='dependents':

result = view\_all\_data\_dependents()

df = pd.DataFrame(result, columns=['dependent\_id','trans\_id','user\_ref\_id','fname','lname','phone' ,'email' ,'dob','relation'])

with st.expander("Current data"):

st.dataframe(df)

list\_of\_user = [i[0] for i in view\_only\_data\_dependents()]

selected\_user = st.selectbox("Task to Delete", list\_of\_user)

st.warning("Do you want to delete ::{}".format(selected\_user))

if st.button("Delete dependents"):

delete\_dependents(selected\_user)

st.success("Dependent has been deleted successfully")

new\_result = view\_all\_data\_dependents()

df2 = pd.DataFrame(new\_result, columns=['dependent\_id','trans\_id','user\_ref\_id','fname','lname','phone' ,'email' ,'dob','relation'])

with st.expander("Updated data"):

st.dataframe(df2)

elif table=='transaction\_status':

result = view\_all\_data\_transaction\_status()

df = pd.DataFrame(result, columns=['trans\_id','u\_id','status'])

with st.expander("Current data"):

st.dataframe(df)

list\_of\_user = [i[0] for i in view\_only\_data\_transaction\_status()]

selected\_user = st.selectbox("Task to Delete", list\_of\_user)

st.warning("Do you want to delete ::{}".format(selected\_user))

if st.button("Delete status"):

delete\_transaction\_status(selected\_user)

st.success("status has been deleted successfully")

new\_result = view\_all\_data\_transaction\_status()

df2 = pd.DataFrame(new\_result, columns=['trans\_id','u\_id','status'])

with st.expander("Updated data"):

st.dataframe(df2)

**read.py**

import pandas as pd

import streamlit as st

import plotly.express as px

from database import view\_all\_user\_account

from database import view\_all\_wallet

from database import view\_all\_data\_transactions

from database import view\_all\_data\_promo\_offers

from database import view\_all\_data\_dependents

from database import view\_all\_data\_transaction\_status

def read(table):

if table=='user\_account':

result = view\_all\_user\_account()

# st.write(result)

df = pd.DataFrame(result, columns=['user\_id','fname','lname' ,'dob' ,'phone' ,'email' ,'country' ,'city' ,'pincode','bank\_name'])

with st.expander("View all user\_accounts"):

st.dataframe(df)

with st.expander("user city"):

task\_df = df['city'].value\_counts().to\_frame()

task\_df = task\_df.reset\_index()

st.dataframe(task\_df)

p1 = px.pie(task\_df, names='index', values='city')

st.plotly\_chart(p1)

elif table=='wallet':

result = view\_all\_wallet()

# st.write(result)

df = pd.DataFrame(result, columns=['t\_id','user\_id','account\_no', 'bank\_name','balance','promocode','loan'])

with st.expander("View all wallets"):

st.dataframe(df)

with st.expander("user Bank\_name"):

task\_df = df['bank\_name'].value\_counts().to\_frame()

task\_df = task\_df.reset\_index()

st.dataframe(task\_df)

p1 = px.pie(task\_df, names='index', values='bank\_name')

st.plotly\_chart(p1)

elif table=='transactions':

result = view\_all\_data\_transactions()

# st.write(result)

df = pd.DataFrame(result, columns=['transaction\_id', 'transaction\_date','transaction\_detail', 'amount','to\_id','from\_id','type\_trans'])

with st.expander("View all transactions"):

st.dataframe(df)

elif table=='promo\_offers':

result = view\_all\_data\_promo\_offers()

# st.write(result)

df = pd.DataFrame(result, columns=['promo\_id','user\_id','start\_date','end\_date','duration','status', 'amount\_value'])

with st.expander("View all promo\_offers"):

st.dataframe(df)

with st.expander("user promos"):

task\_df = df['user\_id'].value\_counts().to\_frame()

task\_df = task\_df.reset\_index()

st.dataframe(task\_df)

p1 = px.pie(task\_df, names='index', values='user\_id')

st.plotly\_chart(p1)

elif table=='dependents':

result = view\_all\_data\_dependents()

# st.write(result)

df = pd.DataFrame(result, columns=['dependent\_id','trans\_id','user\_ref\_id','fname','lname','phone' ,'email' ,'dob','relation'])

with st.expander("View all dependents"):

st.dataframe(df)

elif table=='transaction\_status':

result = view\_all\_data\_transaction\_status()

# st.write(result)

df = pd.DataFrame(result, columns=['trans\_id','u\_id','status'])

with st.expander("View all status"):

st.dataframe(df)