<u>PES UNIVERSITY</u>

<u>DBMS: VIRTUAL MONEY TRANSFER</u>

Submitted By:
Name: ANSHIKA PAL
SRN: PES1UG20CS062
V Semester Section B

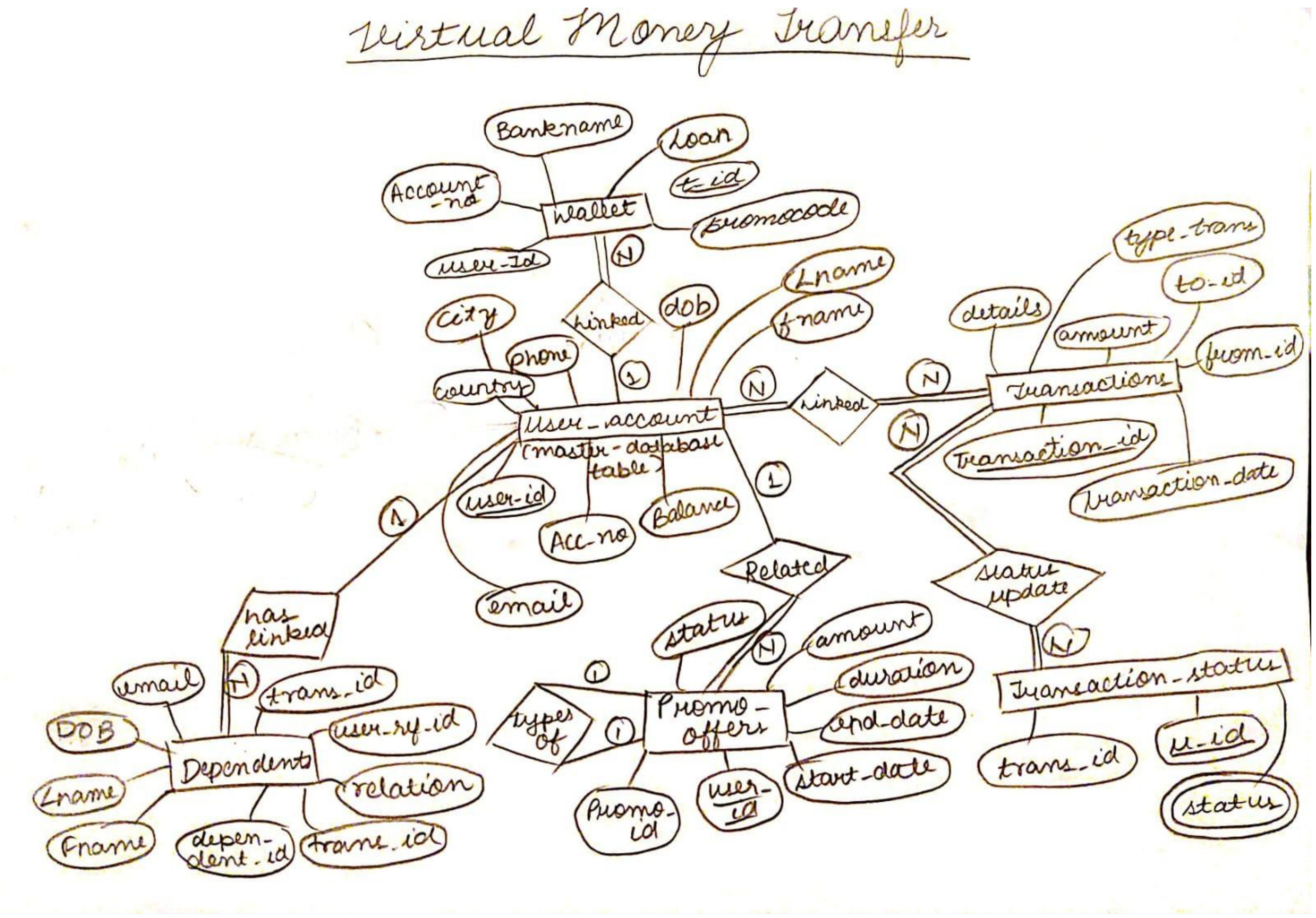
DESCRIPTION:

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:

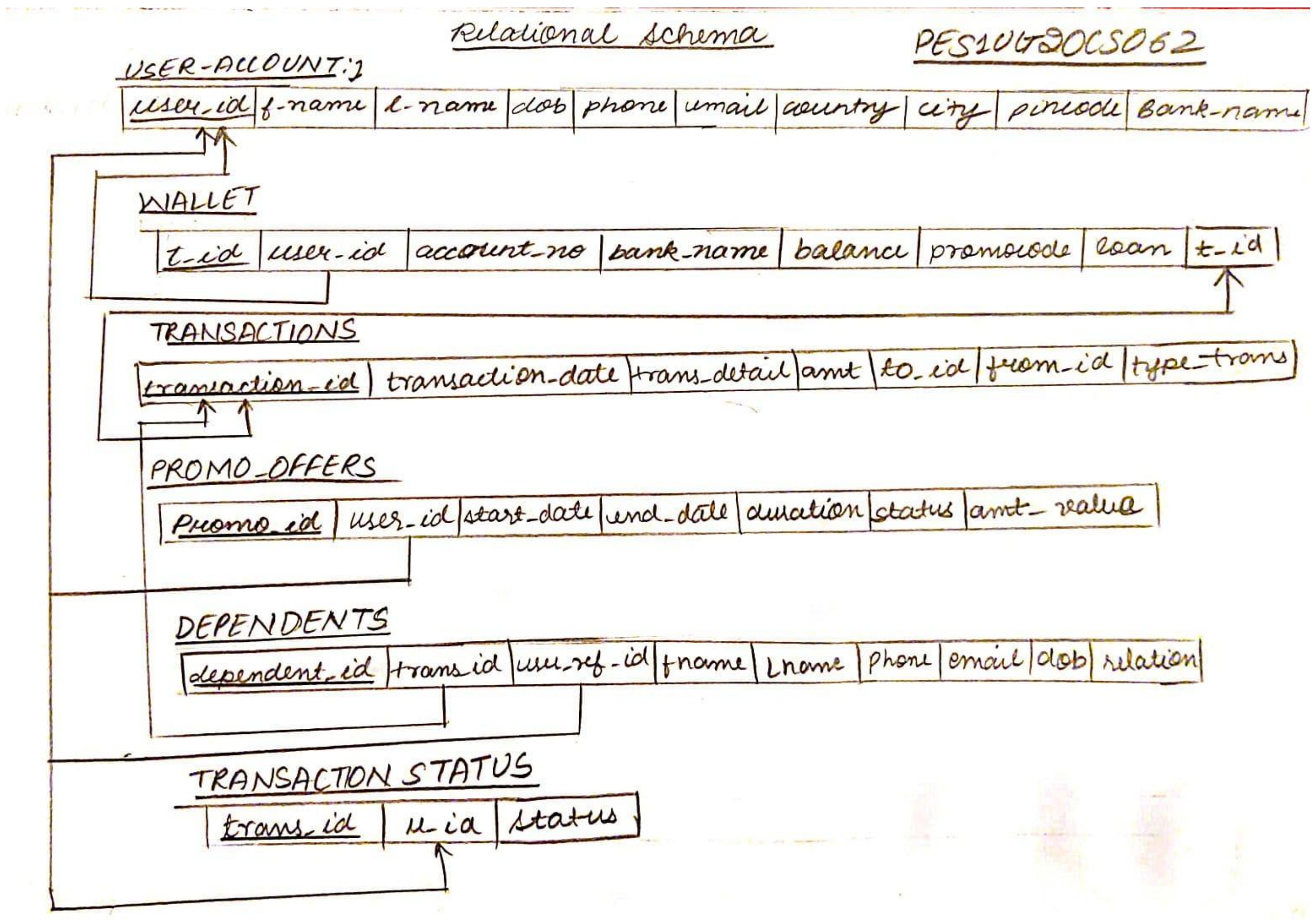
1. Customers can perform financial transactions like transfer funds online, pay bills, apply for loans and open a savings account among various other debit card transactions.
2. You can transfer the funds 24/7 without going to the physical bank.
3. Check account Balance and statements.
4. One can apply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

ER Diagram:



Virtual Money Transfer
Bankname
Loan
t-id
Account-no
wallet
promocode
user-Id
N
linked
1
Lname
fname
city
dob
phone
country
User_account
(master-database table)
N
linked
N
Transactions
type-trans
to-id
details
amount
from-id
N
Transaction_id
Transaction_date
user-id
Acc-no
Balance
1
1
email
Related
status update
has linked
N
status
N
amount
duration
end-date
start-date
Promo-offers
types of
1
1
user-id
Promo-id
email
trans_id
DOB
Dependents
user-ry-id
Lname
relation
Fname
dependent-id
frans-id
Transaction-status
trans-id
u-id
status

Relational Schema:


Relational Schema
PES1UG20CS062
USER-ACCOUNT:
user-id | f-name | l-name | dob | phone | email | country | city | pincode | Bank-name
WALLET
t-id | user-id | account_no | bank-name | balance | promocode | loan | t-id
TRANSACTIONS
transaction-id | transaction-date | trans_detail | amt | to_id | from-id | type-trans
PROMO_OFFERS
Promo_id | user-id | start-date | end-date | duration | status | amt-value
DEPENDENTS
dependent_id | trans id | user-ref-id | fname | Lname | Phone | email | dob | relation
TRANSACTION STATUS
trans_id | u-id | status

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));
```

```sql
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));
```

```sql
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));
```

```sql
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);
```

```sql
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));
```

```sql
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','I
CICI Bank');
```

```sql
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');
```

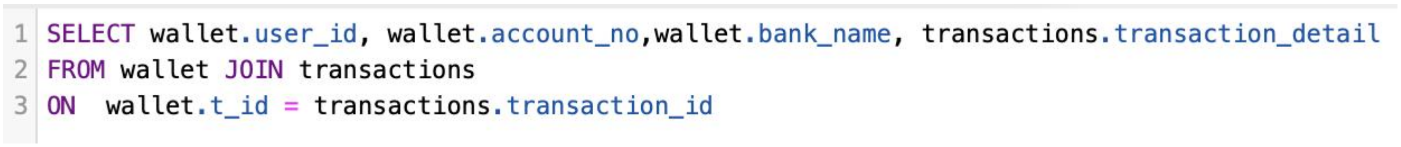
```sql
/*insert into wallet values('T-220001','VB201701','SBIN201701','State Bank Of India',7500,'','');*/
insert into wallet values('T-220002','VB201702','SBIN201702','State Bank Of India',45000,'VB_MAR','');
insert into wallet values('T-220003','VB201703','UBIN201703','State Bank Of India',6000,'VB_MAR','1200');
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');
```

```sql
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:

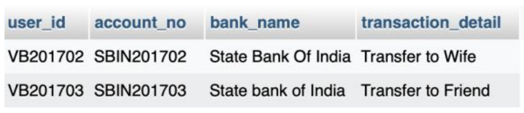
1) Displaying user_id, account_no, bank_name and transaction details of a person sending money to relative.

QUERY:



```
SELECT wallet.user_id, wallet.account_no,wallet.bank_name, transactions.transaction_detail
FROM wallet JOIN transactions
ON  wallet.t_id = transactions.transaction_id
```

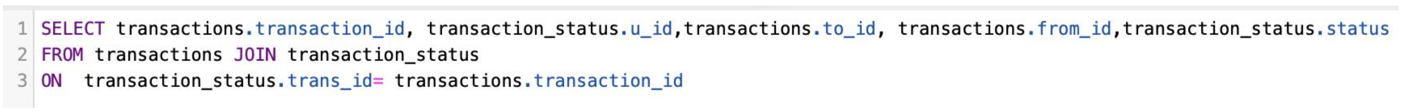
RESULT:



user_id	account_no	bank_name	transaction_detail
VB201702	SBIN201702	State Bank Of India	Transfer to Wife
VB201703	SBIN201703	State bank of India	Transfer to Friend

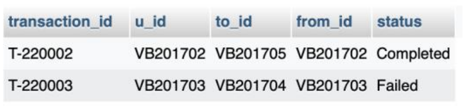
2) SELECT transaction_id,u_id,to_id,from_id, status from joining tables transactions and transaction_status on transaction id. (Showing if the transaction is completed or not).

QUERY:



```
SELECT transactions.transaction_id, transaction_status.u_id,transactions.to_id, transactions.from_id,transaction_status.status
FROM transactions JOIN transaction_status
ON  transaction_status.trans_id= transactions.transaction_id
```

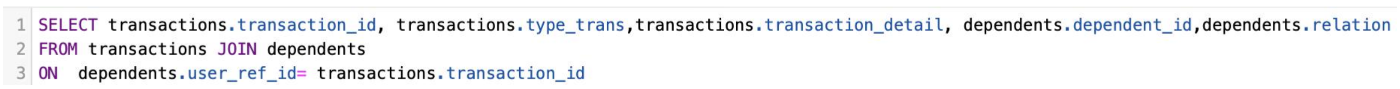
RESULT:



transaction_id	u_id	to_id	from_id	status
T-220002	VB201702	VB201705	VB201702	Completed
T-220003	VB201703	VB201704	VB201703	Failed

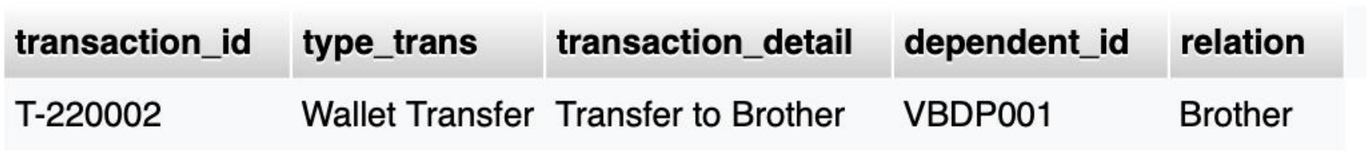
3) Selecting transaction_id, dependent_id, relation, transaction_typ and relation by joining tables transactions and dependents.

QUERY:



```
SELECT transactions.transaction_id, transactions.type_trans,transactions.transaction_detail, dependents.dependent_id,dependents.relation
FROM transactions JOIN dependents
ON  dependents.user_ref_id= transactions.transaction_id
```

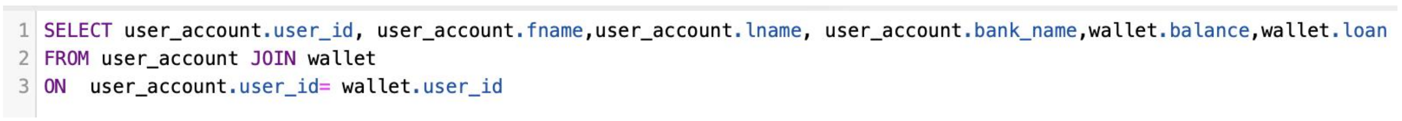
RESULT:



transaction_id	type_trans	transaction_detail	dependent_id	relation
T-220002	Wallet Transfer	Transfer to Brother	VBDP001	Brother

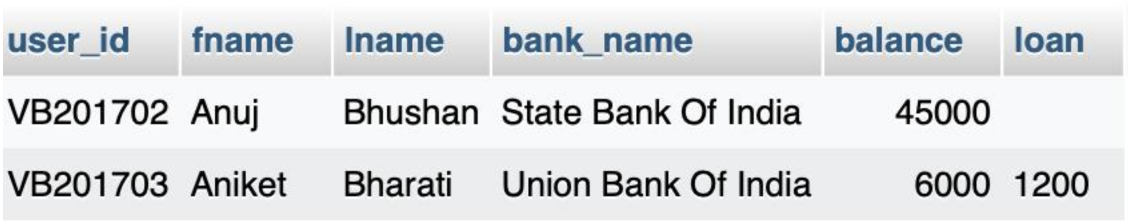
4) Show the user_id with it's name and bank number and how much balance and loan is there on a person.

QUERY:



```sql
SELECT user_account.user_id, user_account.fname,user_account.lname, user_account.bank_name,wallet.balance,wallet.loan
FROM user_account JOIN wallet
ON  user_account.user_id= wallet.user_id
```

RESULT:

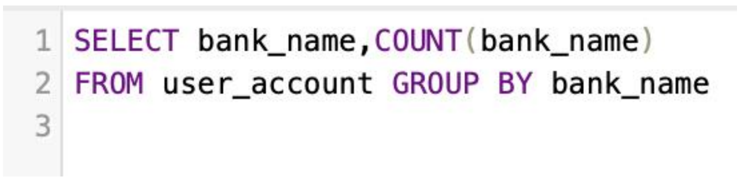


user_id	fname	lname	bank_name	balance	loan
VB201702	Anuj	Bhushan	State Bank Of India	45000	
VB201703	Aniket	Bharati	Union Bank Of India	6000	1200

AGGREGATE FUNCTIONS:

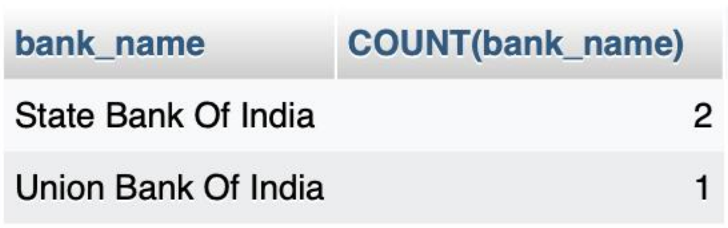
1) Counting the number of banks that user is associated with.

QUERY:



```sql
SELECT bank_name,COUNT(bank_name)
FROM user_account GROUP BY bank_name
```

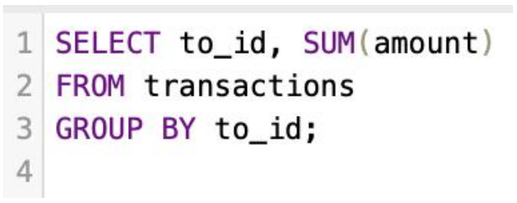
RESULT:



bank_name	COUNT(bank_name)
State Bank Of India	2
Union Bank Of India	1

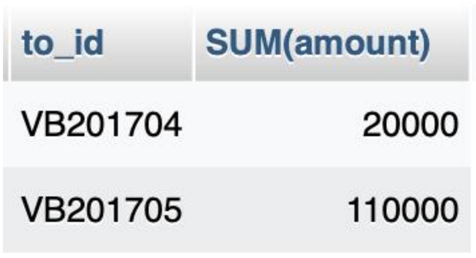
2) Show total sum sent to a person.

QUERY:



```
SELECT to_id, SUM(amount)
FROM transactions
GROUP BY to_id;
```

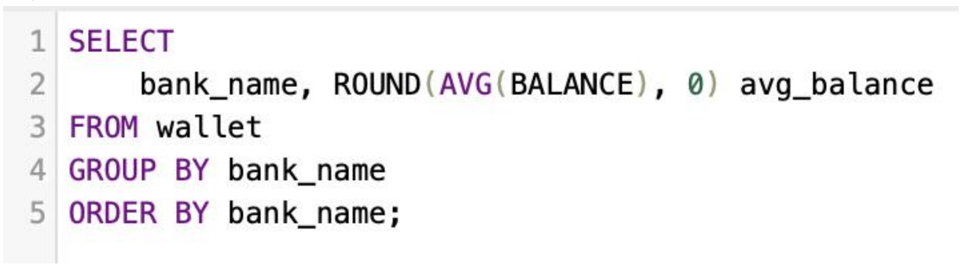
RESULT:



to_id	SUM(amount)
VB201704	20000
VB201705	110000

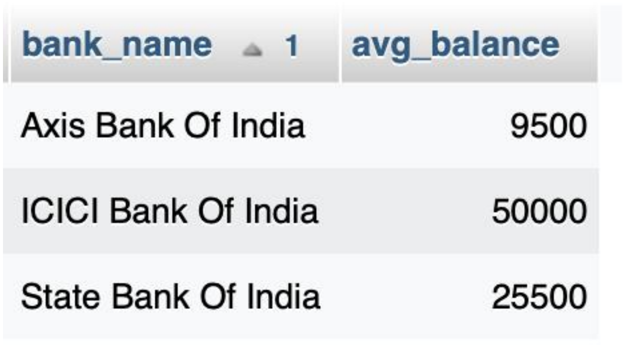
3) Give average values of balance on which the transactions takes place of all the banks.

QUERY:



```
SELECT
    bank_name, ROUND(AVG(BALANCE), 0) avg_balance
FROM wallet
GROUP BY bank_name
ORDER BY bank_name;
```

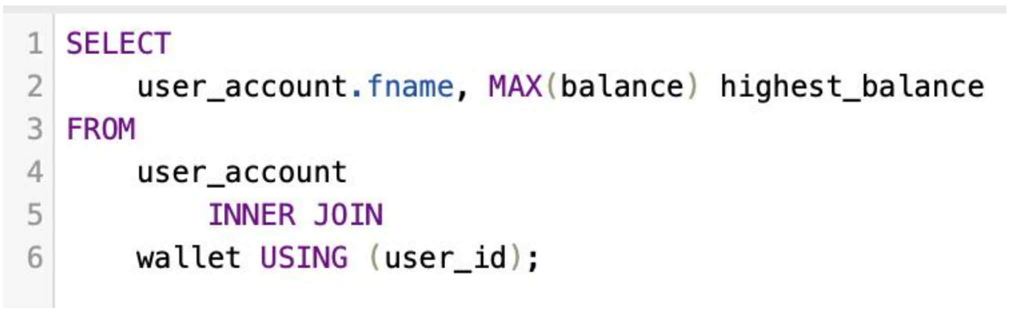
RESULT:



bank_name ▲ 1	avg_balance
Axis Bank Of India	9500
ICICI Bank Of India	50000
State Bank Of India	25500

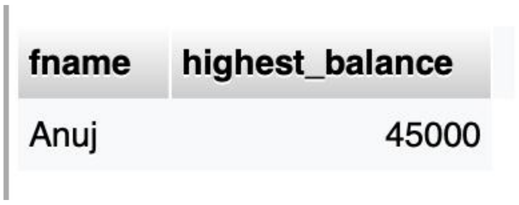
4) Highest balance of all the users in the wallet.

QUERY:



```
SELECT
    user_account.fname, MAX(balance) highest_balance
FROM
    user_account
        INNER JOIN
    wallet USING (user_id);
```

RESULT:



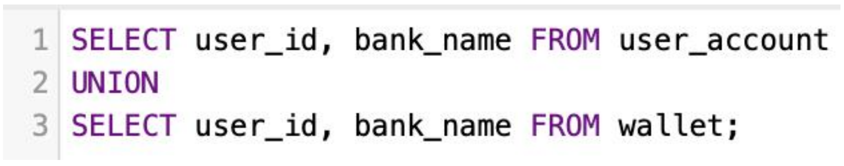
fname	highest_balance
Anuj	45000

#ANUJ HAS HIGHEST BALANCE

Set Operations:

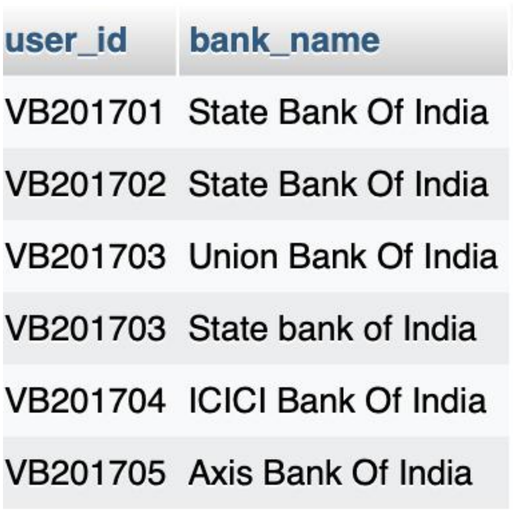
1) Perform union function on user_id and bank_name on user_account and wallet columns.

QUERY:



```
SELECT user_id, bank_name FROM user_account
UNION
SELECT user_id, bank_name FROM wallet;
```

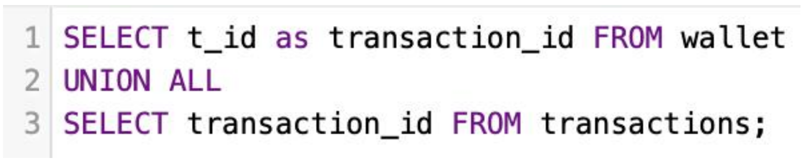
RESULT:



user_id	bank_name
VB201701	State Bank Of India
VB201702	State Bank Of India
VB201703	Union Bank Of India
VB201703	State bank of India
VB201704	ICICI Bank Of India
VB201705	Axis Bank Of India

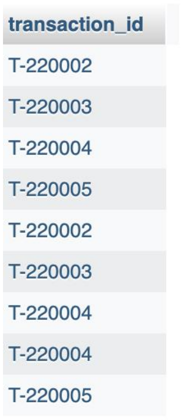
2) UNION ALL on transaction id of wallet and transactions.

QUERY:



```sql
SELECT t_id as transaction_id FROM wallet
UNION ALL
SELECT transaction_id FROM transactions;
```

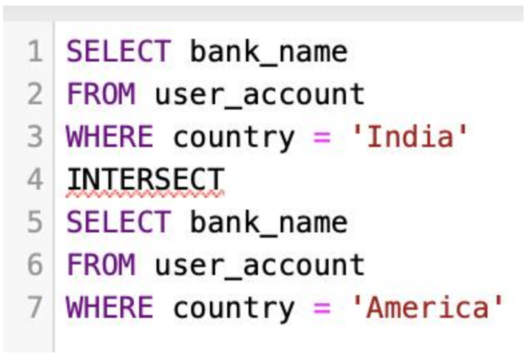
RESULT:



transaction_id
T-220002
T-220003
T-220004
T-220005
T-220002
T-220003
T-220004
T-220004
T-220005

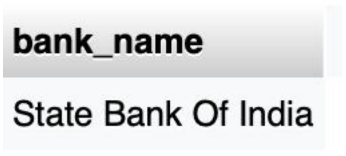
3) Bank name tp which a person registered while living in different countries (i.e. India and America).

QUERY:



```sql
SELECT bank_name
FROM user_account
WHERE country = 'India'
INTERSECT
SELECT bank_name
FROM user_account
WHERE country = 'America'
```

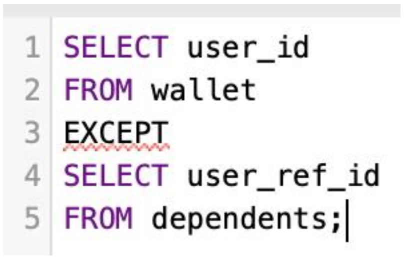
RESULT:



bank_name
State Bank Of India

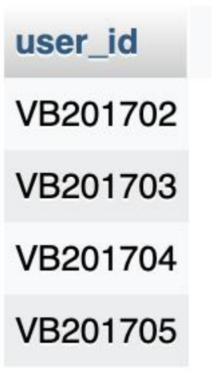
4) Users which didnot give refferal to any person (or have any dependents).

QUERY:



```
SELECT user_id
FROM wallet
EXCEPT
SELECT user_ref_id
FROM dependents;
```

RESULT:

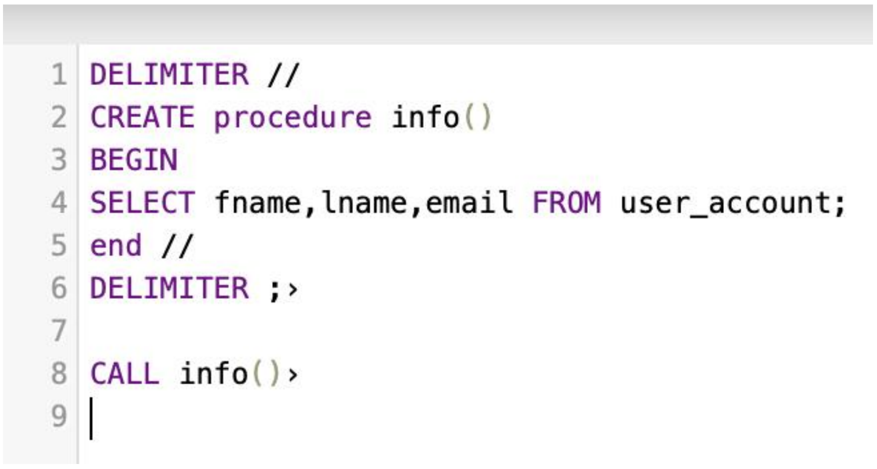


user_id
VB201702
VB201703
VB201704
VB201705

Functions and Procedures:

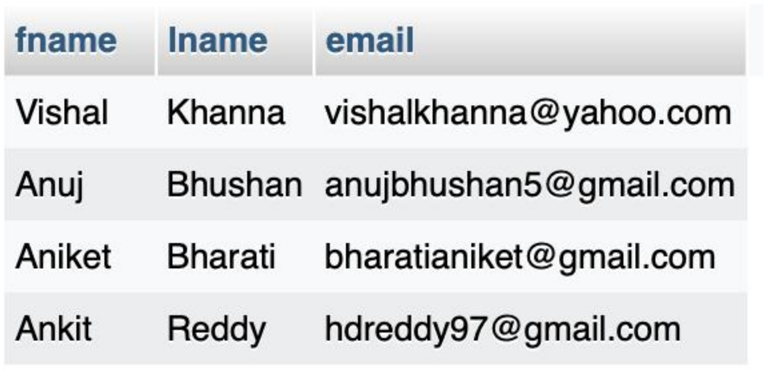
PROCEDURE:

QUERY:



```
DELIMITER //
CREATE procedure info()
BEGIN
SELECT fname,lname,email FROM user_account;
end //
DELIMITER ;›

CALL info()›

```

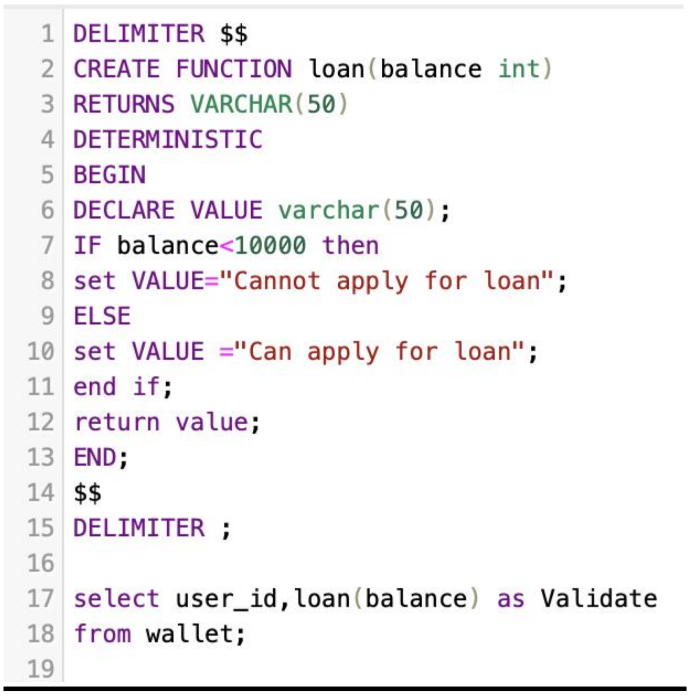
RESULTS:



fname	lname	email
Vishal	Khanna	vishalkhanna@yahoo.com
Anuj	Bhushan	anujbhushan5@gmail.com
Aniket	Bharati	bharatianiket@gmail.com
Ankit	Reddy	hdreddy97@gmail.com

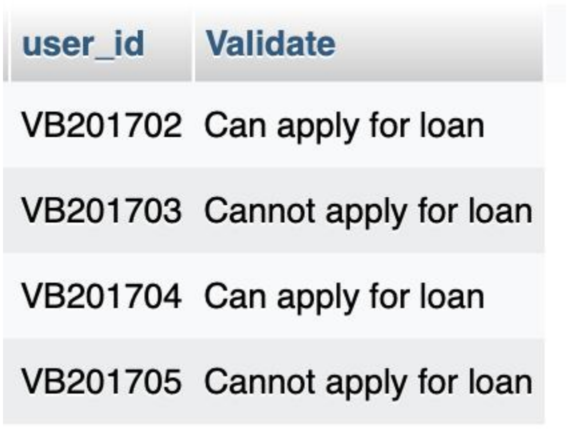
FUNCTION:

QUERY:



```sql
DELIMITER $$
CREATE FUNCTION loan(balance int)
RETURNS VARCHAR(50)
DETERMINISTIC
BEGIN
DECLARE VALUE varchar(50);
IF balance<10000 then
set VALUE="Cannot apply for loan";
ELSE
set VALUE ="Can apply for loan";
end if;
return value;
END;
$$
DELIMITER ;

select user_id,loan(balance) as Validate
from wallet;

```

RESULTS:

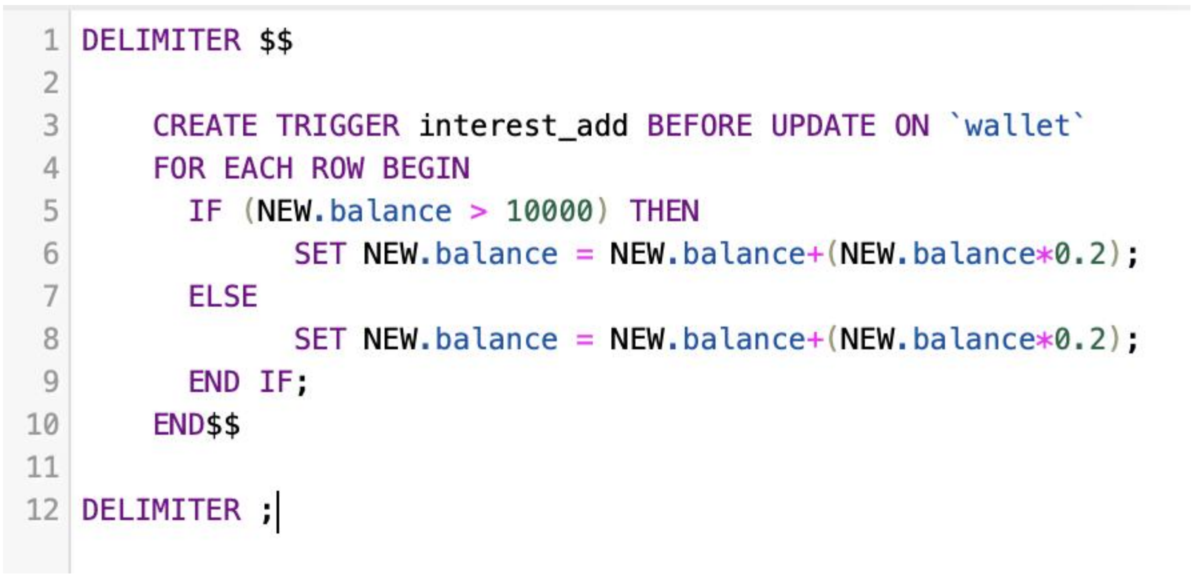


user_id	Validate
VB201702	Can apply for loan
VB201703	Cannot apply for loan
VB201704	Can apply for loan
VB201705	Cannot apply for loan

TRIGGERS AND CURSORS:

TRIGGERS:

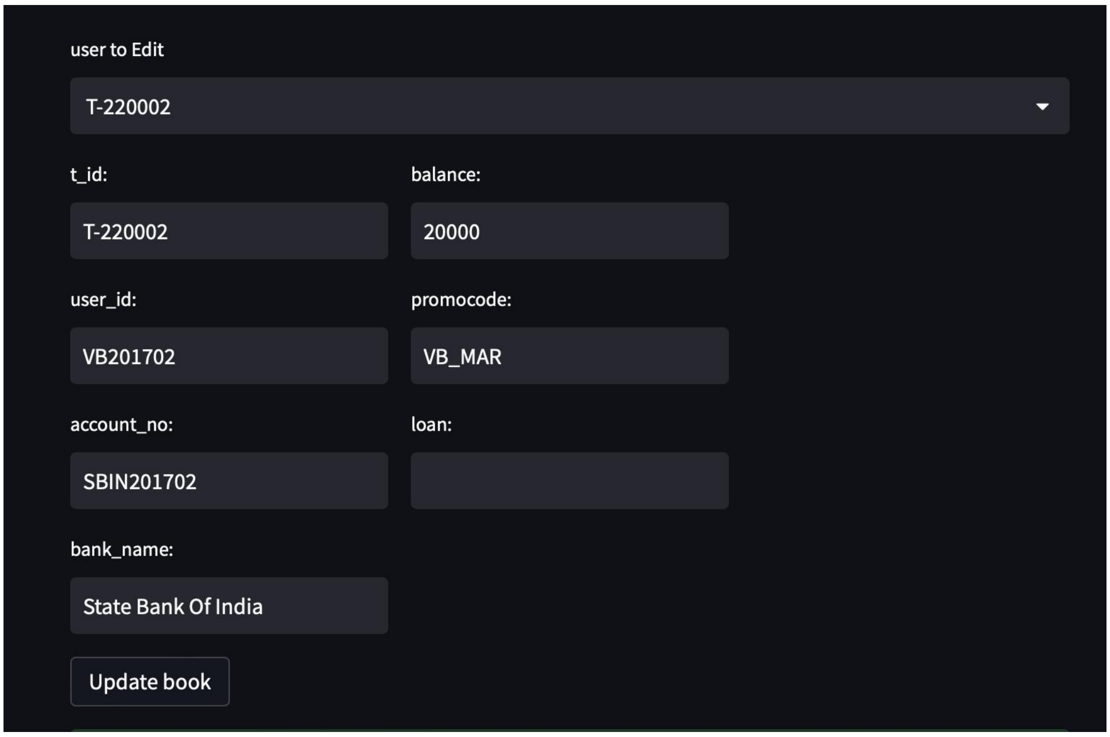
QUERY:



```
DELIMITER $$

    CREATE TRIGGER interest_add BEFORE UPDATE ON `wallet`
    FOR EACH ROW BEGIN
      IF (NEW.balance > 10000) THEN
            SET NEW.balance = NEW.balance+(NEW.balance*0.2);
      ELSE
            SET NEW.balance = NEW.balance+(NEW.balance*0.2);
      END IF;
    END$$

DELIMITER ;
```

RESULT:

UPDATED THE BALANCE TO 20000 WHICH IS > 10000:

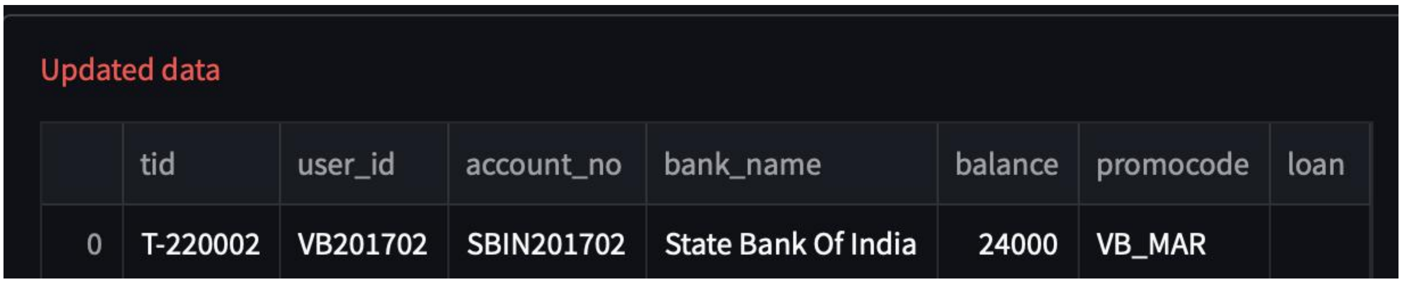


user to Edit

T-220002

t_id:	balance:
T-220002	20000
user_id:	promocode:
VB201702	VB_MAR
account_no:	loan:
SBIN201702	
bank_name:	
State Bank Of India	

AFTER UPDATION INTEREST OF 0.2 GOT ADDED ACC. TO CONDITION


Updated data

	tid	user_id	account_no	bank_name	balance	promocode	loan
0	T-220002	VB201702	SBIN201702	State Bank Of India	24000	VB_MAR	

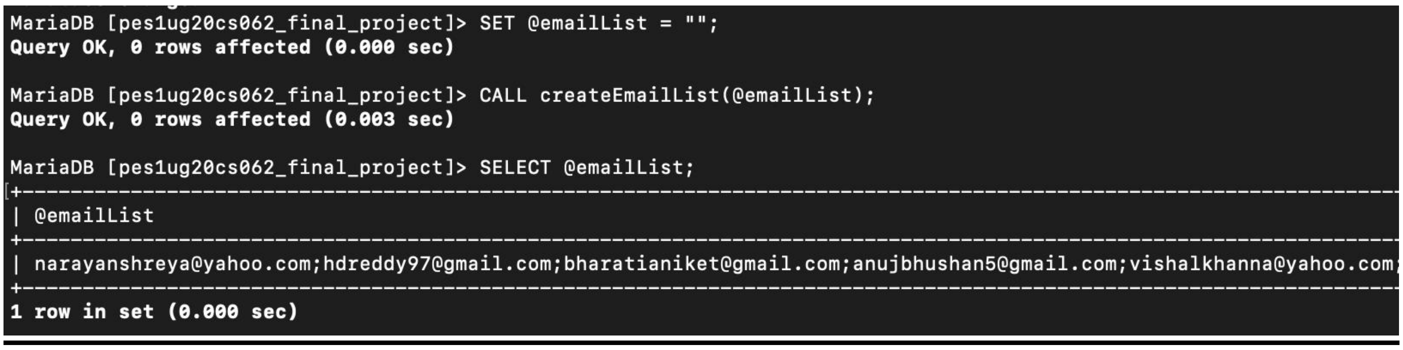
CURSORS:

QUERY:

```
DELIMITER $$
CREATE PROCEDURE createEmailList (
    INOUT emailList varchar(4000)
)
BEGIN
    DECLARE finished INTEGER DEFAULT 0;
    DECLARE emailAddress varchar(100) DEFAULT "";


    DEClARE curEmail
        CURSOR FOR
            SELECT email FROM user_account;


    DECLARE CONTINUE HANDLER
        FOR NOT FOUND SET finished = 1;

    OPEN curEmail;

    getEmail: LOOP
        FETCH curEmail INTO emailAddress;
        IF finished = 1 THEN
            LEAVE getEmail;
        END IF;

        SET emailList = CONCAT(emailAddress,";",emailList);
    END LOOP getEmail;
    CLOSE curEmail;

END$$
DELIMITER ;
```

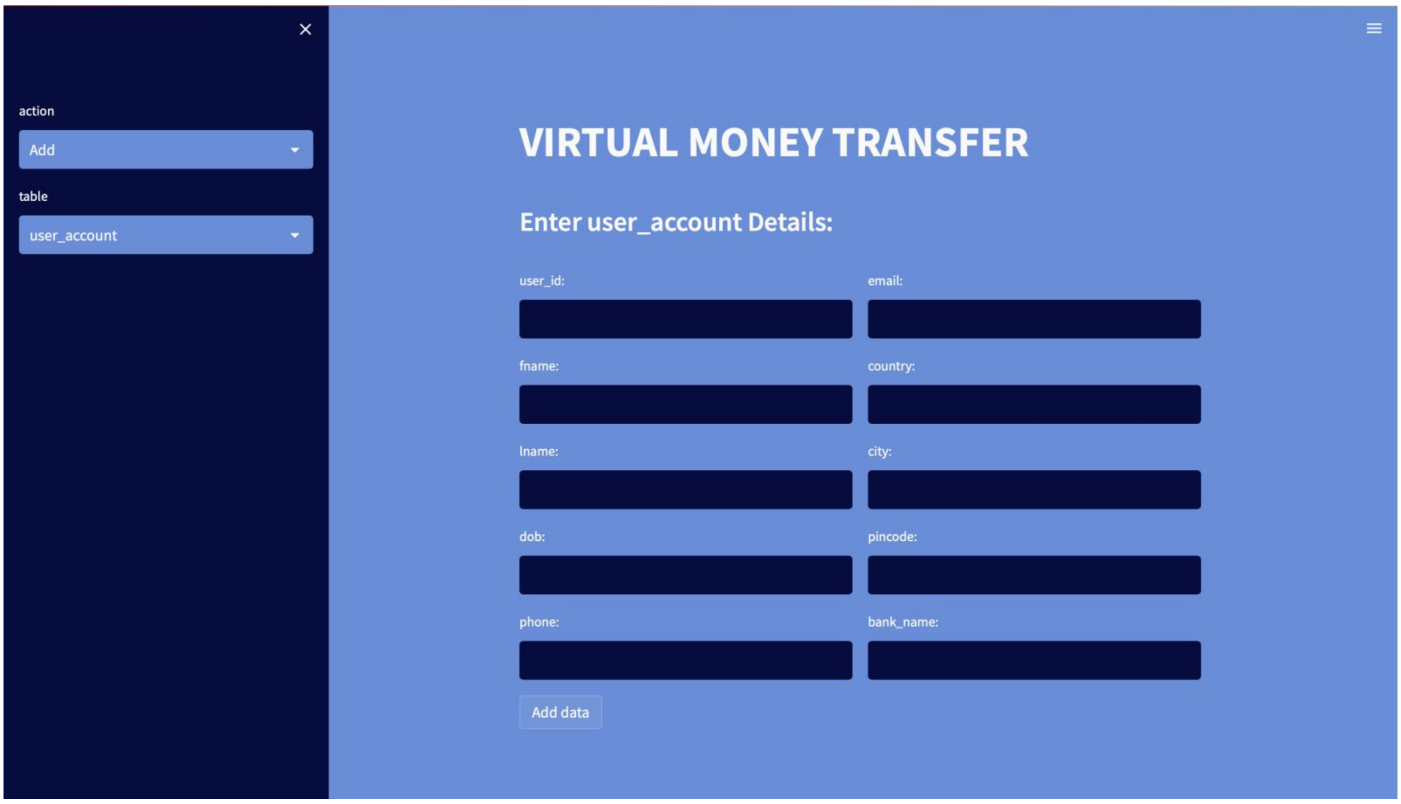
RESULT:

LIST OF ALL EMAILS:



```
MariaDB [pes1ug20cs062_final_project]> SET @emailList = "";
Query OK, 0 rows affected (0.000 sec)

MariaDB [pes1ug20cs062_final_project]> CALL createEmailList(@emailList);
Query OK, 0 rows affected (0.003 sec)

MariaDB [pes1ug20cs062_final_project]> SELECT @emailList;
+------------------------------------------------------------------------------------------------------------------
| @emailList
+------------------------------------------------------------------------------------------------------------------
| narayanshreya@yahoo.com;hdreddy97@gmail.com;bharatianiket@gmail.com;anujbhushan5@gmail.com;vishalkhanna@yahoo.com;
+------------------------------------------------------------------------------------------------------------------
1 row in set (0.000 sec)
```

Developing a Frontend:
MAIN PAGE:


action

Add

table

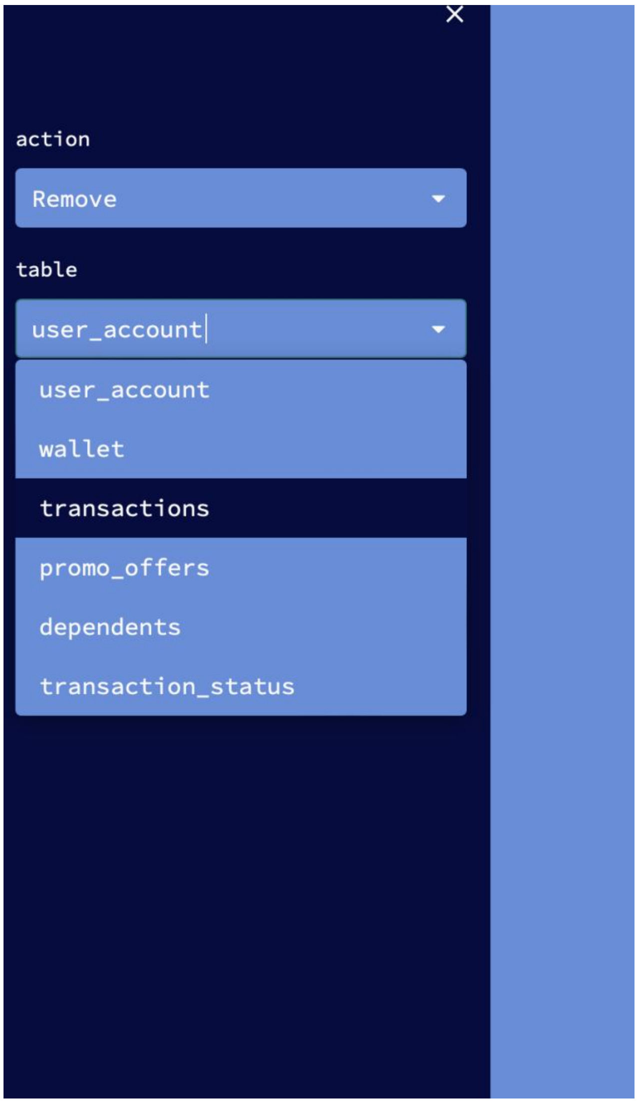
user_account

VIRTUAL MONEY TRANSFER

Enter user_account Details:

user_id: | email:
fname: | country:
lname: | city:
dob: | pincode:
phone: | bank_name:

NAVIGATE THROUGH TABLES:

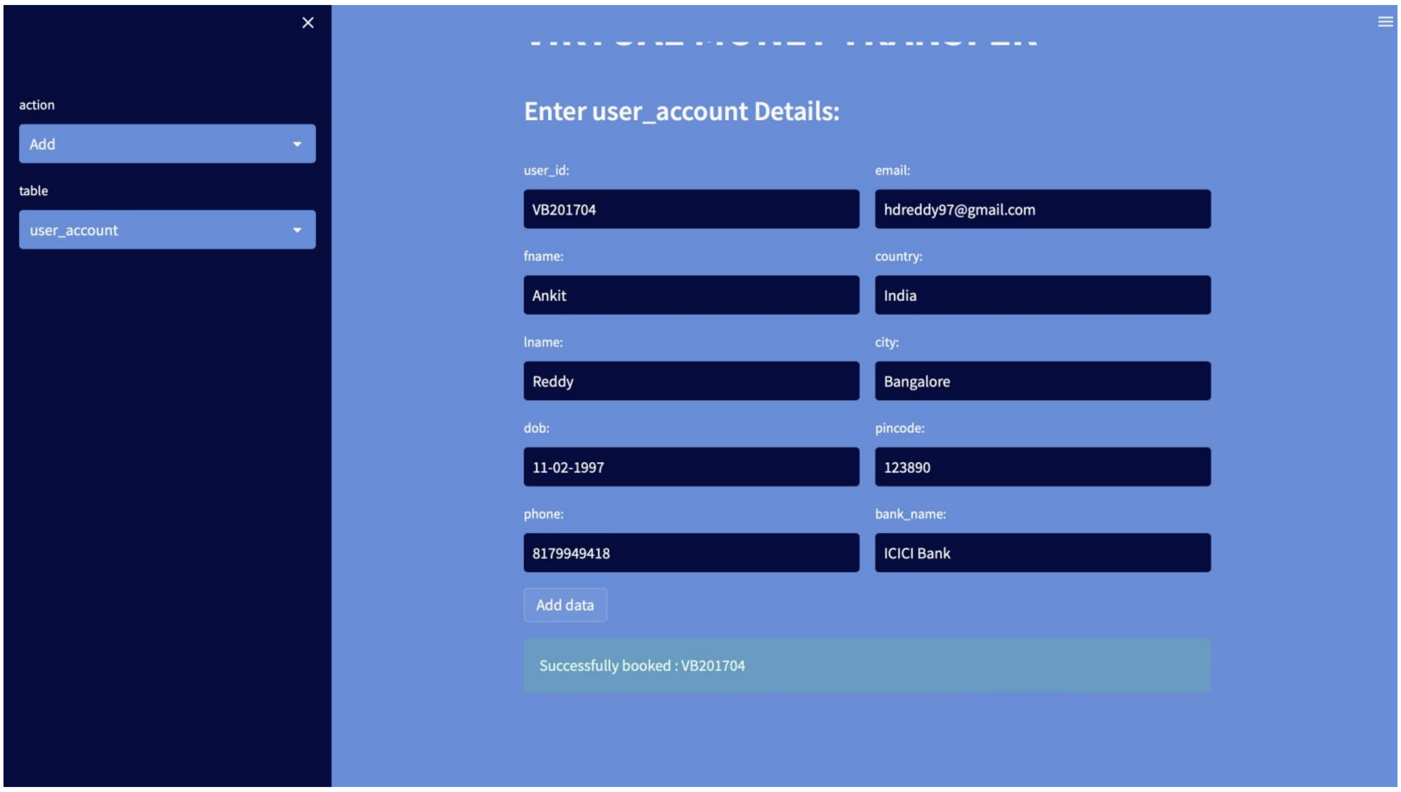

action

Remove

table

user_account

ADDING VALUES:

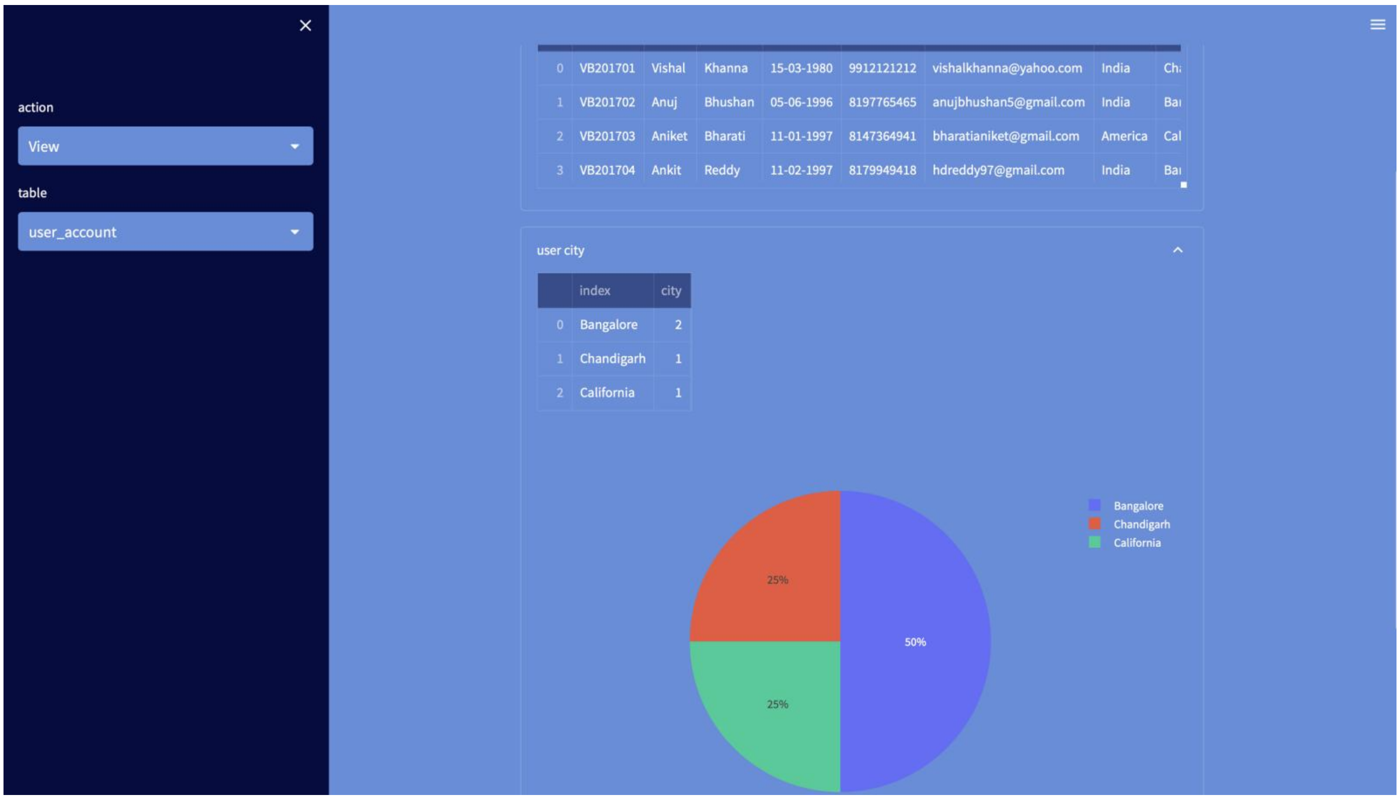


Enter user_account Details:

user_id: VB201704	email: hdreddy97@gmail.com
fname: Ankit	country: India
lname: Reddy	city: Bangalore
dob: 11-02-1997	pincode: 123890
phone: 8179949418	bank_name: ICICI Bank

Successfully booked : VB201704

VIEWING VALUES:

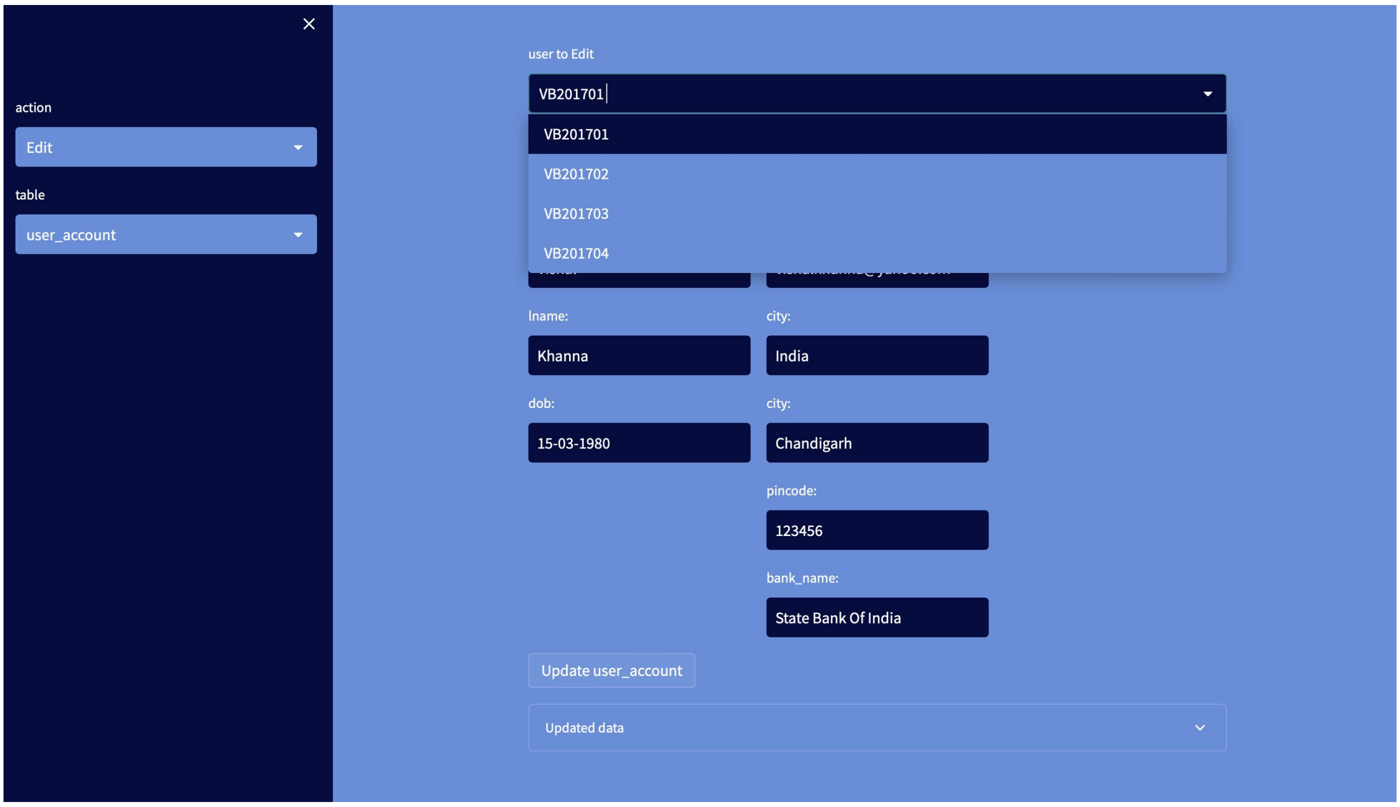


0	VB201701	Vishal	Khanna	15-03-1980	9912121212	vishalkhanna@yahoo.com	India	Ch
1	VB201702	Anuj	Bhushan	05-06-1996	8197765465	anujbhushan5@gmail.com	India	Ba
2	VB201703	Aniket	Bharati	11-01-1997	8147364941	bharatianiket@gmail.com	America	Cal
3	VB201704	Ankit	Reddy	11-02-1997	8179949418	hdreddy97@gmail.com	India	Ba

user city

	index	city
0	Bangalore	2
1	Chandigarh	1
2	California	1

DROP DOWN MENU TO CHOOSE ON WHICH VALUE TO EDIT:



action

Edit

table

user_account

user to Edit

VB201701

VB201701
VB201702
VB201703
VB201704

lname: Khanna

city: India

dob: 15-03-1980

city: Chandigarh

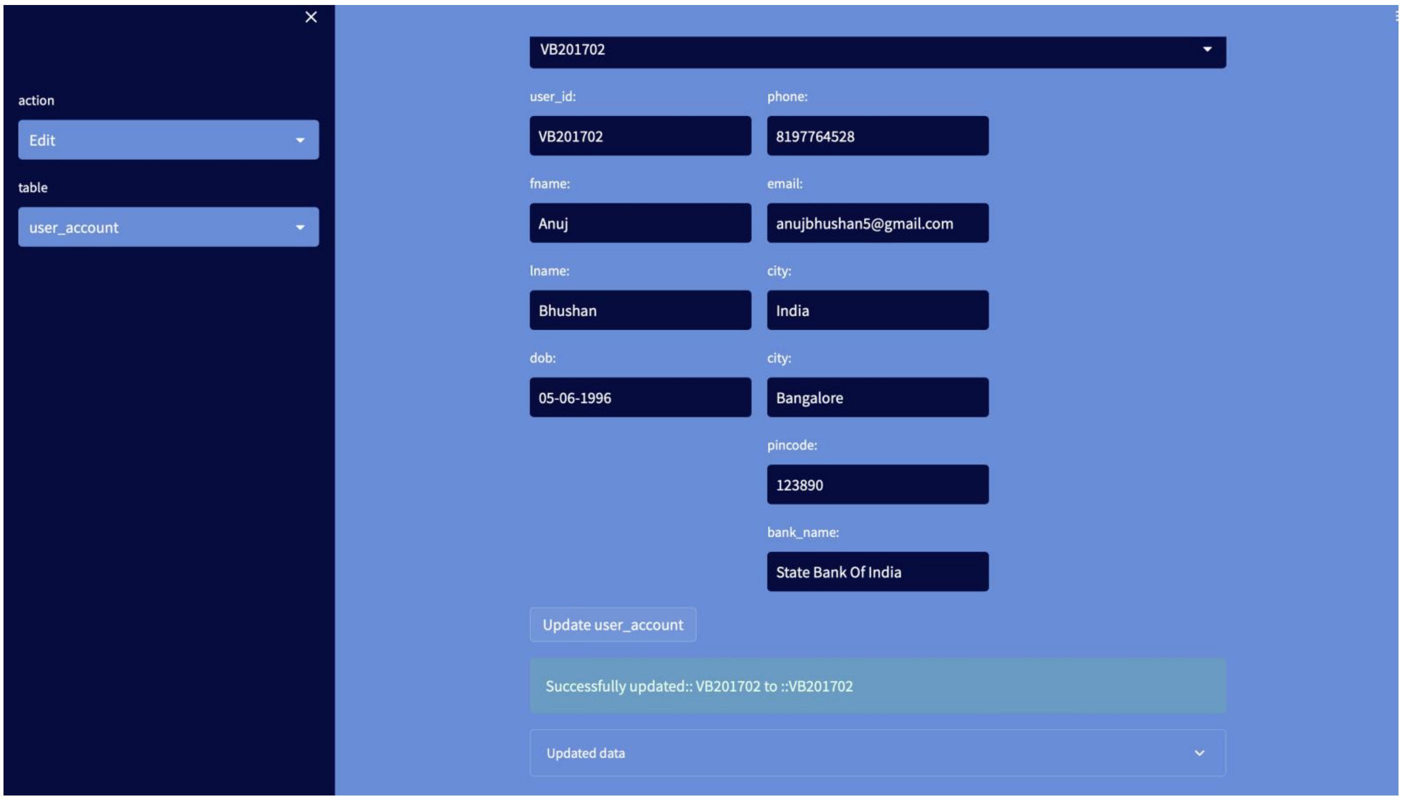
pincode: 123456

bank_name: State Bank Of India

Update user_account

Updated data

UPDATING THE VALUES:



action

Edit

table

user_account

VB201702

user_id: VB201702

phone: 8197764528

fname: Anuj

email: anujbhushan5@gmail.com

lname: Bhushan

city: India

dob: 05-06-1996

city: Bangalore

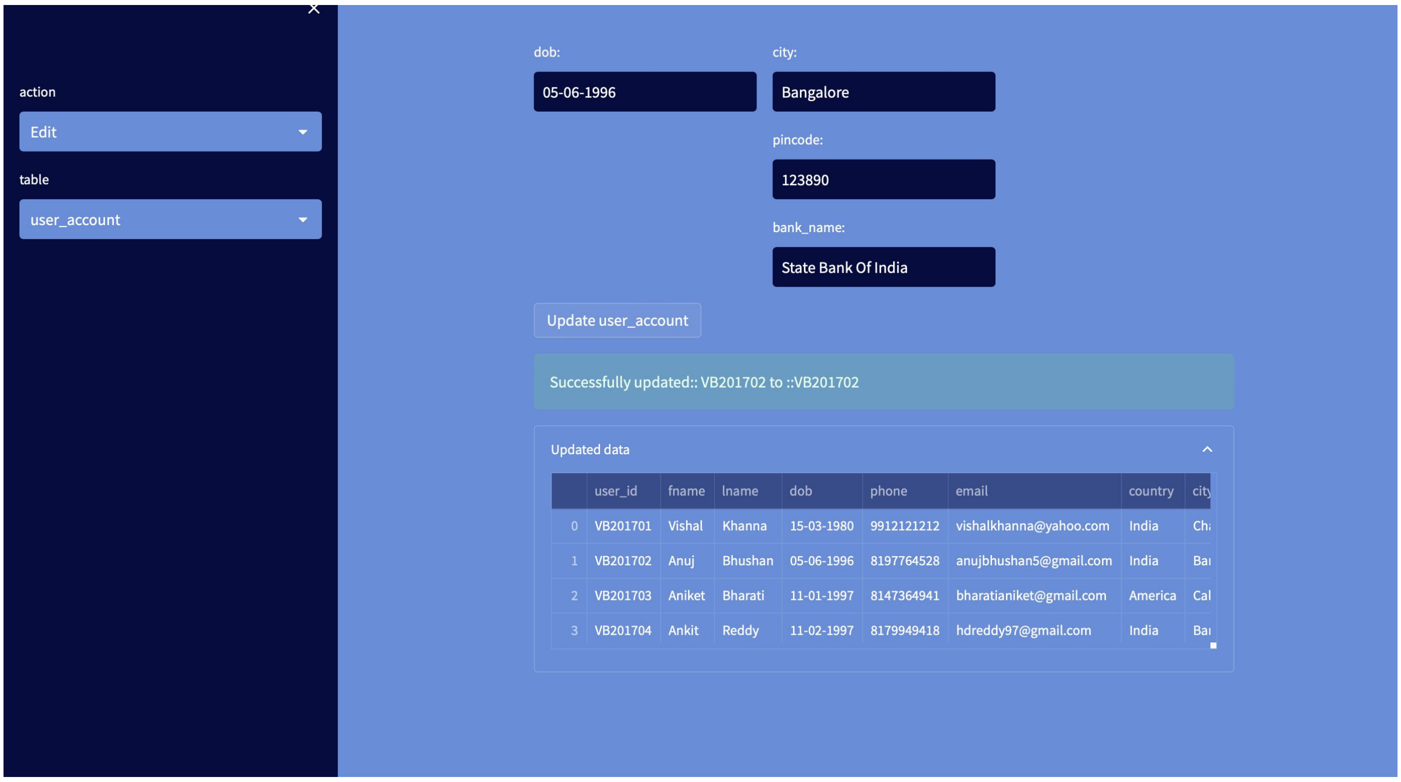
pincode: 123890

bank_name: State Bank Of India

Update user_account

Successfully updated:: VB201702 to ::VB201702

Updated data



action

Edit

table

user_account

dob: 05-06-1996

city: Bangalore

pincode: 123890

bank_name: State Bank Of India

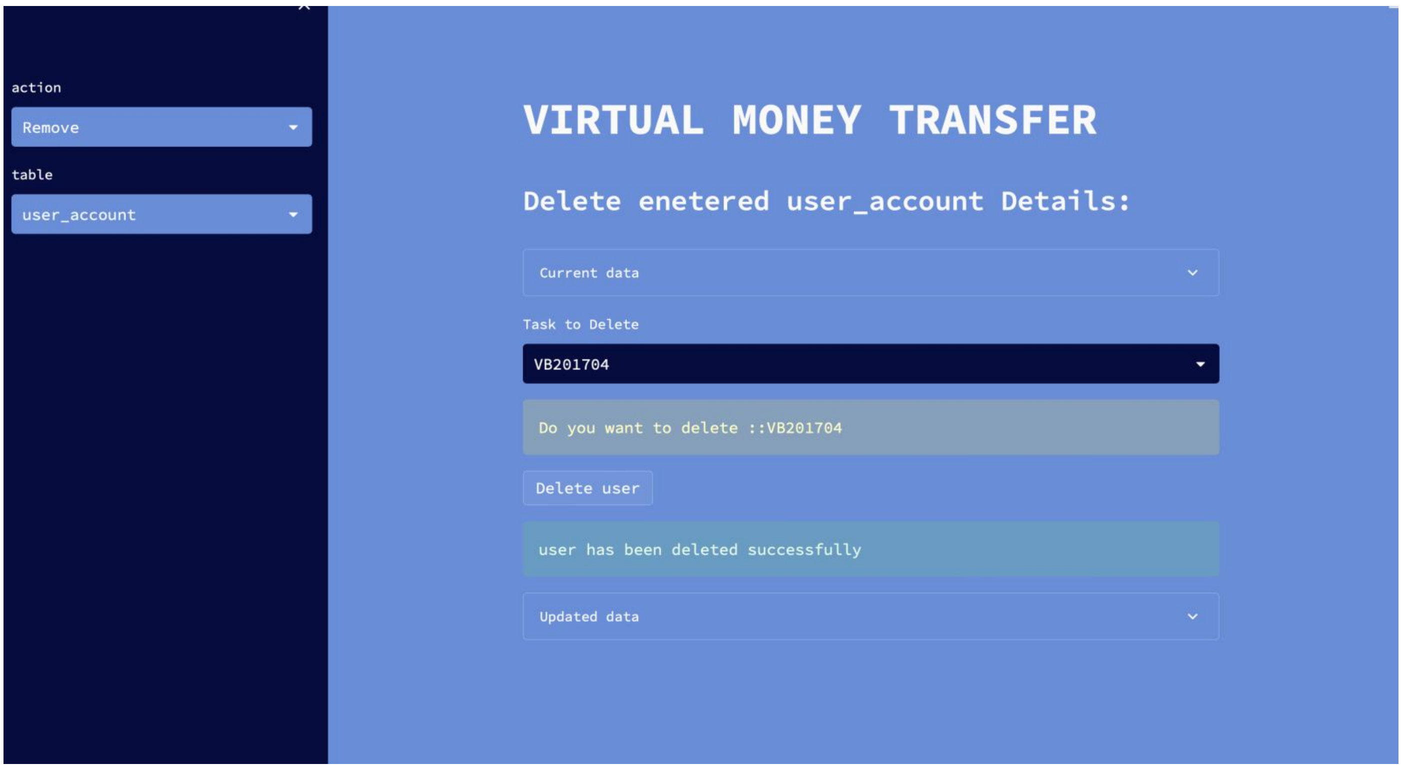
Update user_account

Successfully updated:: VB201702 to ::VB201702

Updated data

	user_id	fname	lname	dob	phone	email	country	city
0	VB201701	Vishal	Khanna	15-03-1980	9912121212	vishalkhanna@yahoo.com	India	Chi
1	VB201702	Anuj	Bhushan	05-06-1996	8197764528	anujbhushan5@gmail.com	India	Bai
2	VB201703	Aniket	Bharati	11-01-1997	8147364941	bharatianiket@gmail.com	America	Cal
3	VB201704	Ankit	Reddy	11-02-1997	8179949418	hdreddy97@gmail.com	India	Bai

DELETING THE VALUES:



action

Remove

table

user_account

VIRTUAL MONEY TRANSFER

Delete enetered user_account Details:

Current data

Task to Delete

VB201704

Do you want to delete ::VB201704

Delete user

user has been deleted successfully

Updated data

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","depe
ndents","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 ,c
ity ,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,stat
us, 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 ,c
ity ,pincode,bank_name):
c.execute('INSERT INTO
user_account(user_id,fname,lname ,dob ,phone ,email ,country ,city ,pinc
ode,bank_name) VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)',
(user_id,fname,lname ,dob ,phone ,email ,country ,city ,pincode,bank_nam
e))
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,stat
us, 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 ,d
ob,relation) VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s)',
(dependent_id,trans_id,user_ref_id,fname,lname,phone ,email ,dob,relatio
n))
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_pho
ne ,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,ln
ame ,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_d
ate,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_s
tatus,
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_f
name,new_lname,new_phone,new_email,new_dob,new_relation,dependent_id,tra
ns_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_p
hone,new_email,new_dob,new_relation,dependent_id,trans_id,user_ref_id,fn
ame,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_pho
ne ,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_d
ate,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_f
name,new_lname,new_phone,new_email,new_dob,new_relation,dependent_id,tra
ns_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)
```

```python
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

```python
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)
```