import random;

import cx\_Oracle;

import datetime;

import time;

import sys;

con = cx\_Oracle.connect("HR-WIN10/GOOD\*8god@127.0.0.1/xe")

cur = con.cursor()

class Account:

def \_\_init\_\_(self,Account\_Id,Customer\_Id):

self.\_Account\_Id = Account\_Id;

self.\_Customer\_Id = Customer\_Id;

class Savings\_Account(Account):

def \_\_init\_\_(self,Account\_Id,Customer\_Id):

super().\_\_init\_\_(Account\_Id,Customer\_Id)

flag = 1

while flag:

print("Please enter the password . Special characters are allowed.Mininmum 8 characters Maximimum 15 characters")

passwd = input()

if len(passwd) < 8 or len(passwd) > 15:

print("Incorrect Password pattern. Please try again")

else:

print("Password set successfully")

flag = 0

self.\_\_Customer\_Password = passwd

print("Your balance in the account is 0")

self.\_\_Balance = 0;

acc = self.\_Account\_Id

cus = self.\_Customer\_Id

statement = "insert into SAVINGS\_ACCOUNT(ACCOUNT\_ID,CUSTOMER\_ID,BALANCE,PASSWORD) values (:0,:1,:2,:3)"

cur.execute(statement,{"0":acc,"1":cus,"2":self.\_\_Balance,"3":passwd})

con.commit()

print("Savings account created successfully")

class Current\_Account(Account):

def \_\_init\_\_(self,Account\_Id,Customer\_Id):

super().\_\_init\_\_(Account\_Id,Customer\_Id);

acc = self.\_Account\_Id

cus = self.\_Customer\_Id

flag = 1

while flag:

print("Please enter the password . Special characters are allowed.Mininmum 8 characters Maximimum 15 characters")

passwd = input()

if len(passwd) < 8 or len(passwd) > 15:

print("Incorrect Password pattern. Please try again")

else:

print("Password set successfully")

flag = 0

self.\_\_Customer\_Password = passwd

print("You should have a minimum balance of Rs.5000")

print("Please deposit an amount greater than or equal to Rs.5000")

amount = float(input())

if amount < 5000:

print("Sorry...You cannot open a current account unless you maintain a minimum balance of Rs.5000")

else:

self.\_\_Balance = amount;

statement = "insert into CURRENT\_ACCOUNT(ACCOUNT\_ID,CUSTOMER\_ID,BALANCE,PASSWORD) values (:0,:1,:2,:3)"

cur.execute(statement,{"0":acc,"1":cus,"2":self.\_\_Balance,"3":passwd})

con.commit()

print("Current account created successfully")

class Fixed\_Deposit:

def \_\_init\_\_(self,CID):

Customer\_Id = CID;

Initial\_Number = "51250"

End\_Number = str(random.randint(1,100000))

self.FixedDeposit\_Id = Initial\_Number + End\_Number

print("Your Fixed Deopsit ID is")

print(self.FixedDeposit\_Id)

print("Enter the amount you wish to deposit. Enter it int multiples of 1000")

print("Minimum FD Balance is 1000")

amount = float(input())

if (amount % 1000) == 0 and (amount > 0):

print("Enter the Term for Deposit in terms of month.Minimum term is 12 months")

term = int(input())

if term >= 12:

statement = "insert into FIXED\_DEPOSIT(ACCOUNT\_ID,CUSTOMER\_ID,MONEY\_DEPOSITED,TERM) values (:0,:1,:2,:3)"

cur.execute(statement,{"0":self.FixedDeposit\_Id,"1":Customer\_Id,"2":amount,"3":term})

con.commit()

print("Fixed deposit account opened successfully")

statement = "select ACCOUNT\_ID,MONEY\_DEPOSITED,TERM from FIXED\_DEPOSIT where CUSTOMER\_ID = :0"

result = cur.execute(statement,{"0":CID})

print("The Fixed Deposit accounts Owned by you are")

print("Account\_Id \t\t\t MONEY\_DEPOSITED \t\t\t TERM")

for each in result:

print(each[0],"\t\t\t",each[1],"\t\t\t\t\t",each[2])

else:

print("Enter a valid term")

else:

print("Enter a valid amount")

class Customer:

pass;

class New\_Customer(Customer):

def \_\_init\_\_(self):

Initial\_Number = "51250"

End\_Number = str(random.randint(1,100000))

self.\_\_Customer\_Id = Initial\_Number + End\_Number

print("Your customer ID is")

print(self.\_\_Customer\_Id)

print("Enter your First\_Name")

self.\_\_Customer\_Fname = input()

print("Enter your Last\_Name")

self.\_\_Customer\_Lname = input()

print("Enter Address Line1")

self.\_\_Customer\_ALine1 = input()

print("Enter Address Line2")

self.\_\_Customer\_ALine2 = input()

print("Enter your City")

self.\_\_City = input()

print("Enter your State")

self.\_\_State = input()

print("Enter Pincode")

code = input()

flag = 1

while flag:

if len(code) == 6:

self.\_\_Pincode = int(code)

flag = 0

else:

print("Enter a Valid 6-digit Pincode")

code = input()

flag = 1

while flag:

print("Please enter the password . Special characters are allowed.Mininmum 8 characters Maximimum 15 characters")

passwd = input()

if len(passwd) < 8 or len(passwd) > 15:

print("Incorrect Password pattern. Please try again")

else:

print("Password set successfully")

flag = 0

self.\_\_Customer\_Password = passwd

statement = "Insert into CUSTOMERS(CUSTOMER\_ID,FIRST\_NAME,LAST\_NAME,LINE1,LINE2,CITY,STATE,PINCODE,PASSWORD) values(:0,:1,:2,:3,:4,:5,:6,:7,:8)"

cur.execute(statement,{"0":self.\_\_Customer\_Id,"1":self.\_\_Customer\_Fname,"2":self.\_\_Customer\_Lname,"3":self.\_\_Customer\_ALine1,"4":self.\_\_Customer\_ALine2,"5":self.\_\_City,"6":self.\_\_State,"7":self.\_\_Pincode,"8":self.\_\_Customer\_Password})

con.commit();

print("Registration was successful")

class Existing\_Customer(Customer):

def \_\_init\_\_(self):

print("Enter the Customer\_Id")

CID = input()

statement = "select PASSWORD from CUSTOMERS where CUSTOMER\_ID = :0"

result = cur.execute(statement,{"0":CID})

count = cur.fetchall()

counter = len(count)

if counter > 0:

passwd = count[0][0]

trial = 1

flag = 1

while flag:

password = input("Enter your password")

if password == passwd and trial <= 3:

print("Successful Log in")

flag = 0

self.show\_menu(CID);

elif trial== 3:

print("You have entered wrong password for three times.Please try after sometime")

flag = 0

else:

print("You have entered a wrong password. Try again")

trial = trial + 1;

else:

print("Check your Customer\_Id")

return

def show\_menu(self,CID):

flag = 1

while flag:

print("\t\t\t\tSUB MENU")

print("\t\t\t1.Address Change")

print("\t\t\t2.Money Deposit")

print("\t\t\t\t2.1Open S.A")

print("\t\t\t\t2.2Open C.A")

print("\t\t\t\t2.3Open F.D")

print("\t\t\t3.Money Deposit")

print("\t\t\t4.Money Withdrawal")

print("\t\t\t5.Print Statement")

print("\t\t\t6.Account Closure")

print("\t\t\t7.Avail Loan")

print("\t\t\t8.LogOut")

print("\t\t\tEnter your choice")

choice = input()

if choice == "1":

t = Transaction()

t.Address\_Change(CID)

flag = 1

if choice == "2.1":

savings = Savings\_Account(CID,CID)

flag = 1

if choice == "2.2":

current = Current\_Account(CID,CID)

flag = 1

if choice == "2.3":

fixed = Fixed\_Deposit(CID)

flag = 1

if choice == "3":

print("Enter the account type you wish to make a deposit.Eg-Savings or Current")

account\_type = input()

if account\_type.upper() != ("SAVINGS") and (account\_type.upper() != ("CURRENT")):

print("Enter a valid account type")

if account\_type.upper() == "SAVINGS":

t = Transaction()

t.Savings\_Account\_Deposit(CID);

if account\_type.upper() == "CURRENT":

t = Transaction()

t.Current\_Account\_Deposit(CID);

flag = 1

if choice == "4":

print("Enter the account type from which you wish to withdraw.Eg-Savings or Current")

account\_type = input()

if (account\_type.upper() != "SAVINGS") and (account\_type.upper() != "CURRENT"):

print("Enter a valid account type")

if account\_type.upper() == "SAVINGS":

t = Transaction()

t.Savings\_Account\_Withdraw(CID);

if account\_type.upper() == "CURRENT":

t = Transaction()

t.Current\_Account\_Withdraw(CID)

flag = 1

if choice == "5":

t = Transaction()

t.Print\_Statement(CID)

flag = 1

if choice == "6":

print("Which account you wish to close")

account\_type = input()

if (account\_type.upper() != 'SAVINGS') and (account\_type.upper()!= 'CURRENT'):

print("Enter a valid account type")

if account\_type.upper() == 'SAVINGS':

atype = 'SAVINGS'

t = Transaction()

t.Account\_Closure(CID,atype)

if account\_type.upper() == 'CURRENT':

atype = 'CURRENT'

t = Transaction()

t.Account\_Closure(CID,atype)

flag = 1

if choice == "7":

t = Transaction()

t.Avail\_Loan(CID)

flag = 1

if choice == "8":

print("Logged out successfully")

flag = 0

class Transaction:

def Address\_Change(self,Customer\_Id):

print("Enter Address Line1")

Line1 = input()

print("Enter Address Line2")

Line2 = input()

print("Enter City")

City = input()

print("Enter State")

State = input()

print("Enter 6 digit Pincode")

code = input()

flag = 1

while flag:

if len(code) == 6:

Pincode = int(code)

flag = 0

else:

print("Enter a Valid 6-digit Pincode")

code = input()

statement = "Update CUSTOMERS set LINE1 = :2, LINE2 = :3, CITY = :4, STATE = :5 ,PINCODE = :6 where CUSTOMER\_ID = :7"

cur.execute(statement,{"2":Line1,"3":Line2,"4":City,"5":State,"6":Pincode,"7":Customer\_Id})

con.commit()

def Savings\_Account\_Deposit(self,CID):

print("Enter the Account number")

Anumber = input()

if Anumber == CID:

statement = "select PASSWORD,STATUS,BALANCE from SAVINGS\_ACCOUNT where CUSTOMER\_ID = :0"

result = cur.execute(statement,{"0":CID})

counter = cur.fetchall()

count = len(counter)

if count > 0:

passwd = counter[0][0]

status = counter[0][1]

Bal = counter[0][2]

trial = 1

flag = 1

if status == "Unlocked":

while flag:

password = input("Enter your password")

if password == passwd and trial <= 3:

print("Successful Log in")

flag = 0

elif trial == 3:

print("You have entered an eroneous password 3 times consecutively.Your account is locked.Please contact bank for more details")

flag = 0

cur.execute("update SAVINGS\_ACCOUNT set STATUS = 'Locked' where CUSTOMER\_ID = :0",{"0":CID})

con.commit()

else:

print("You have entered a wrong password. Try again")

trial = trial + 1;

else:

print("Your Account is Locked")

return

print("Enter the amount you wish to deposit")

amount = float(input())

if amount <= 0.0:

print("Invalid Amount")

else:

total = Bal + (amount + ((7.5/100)/12)\*amount)

statement = "update SAVINGS\_ACCOUNT set BALANCE = :0 where CUSTOMER\_ID = :1"

cur.execute(statement,{"0":total,"1":CID})

con.commit()

statement = "insert into TRANSACTION\_SAVINGS\_ACCOUNT(ACCOUNT\_ID,TRANSACTION\_TYPE,DATE\_OF\_TRANSACTION,AMOUNT\_UNDER\_TRANSACTION,BALANCE)values(:0,:1,:2,:3,:4)"

cur.execute(statement,{"0":CID,"1":"Credit","2":datetime.datetime.now(),"3":amount,"4":total})

con.commit()

print("Your new Balance is ",total)

else:

print("You may not have savings account")

else:

print("Invalid Account number")

def Current\_Account\_Deposit(self,CID):

print("Enter the Account number")

Anumber = input()

if Anumber == CID:

statement = "select PASSWORD,STATUS,BALANCE from CURRENT\_ACCOUNT where CUSTOMER\_ID = :0"

result = cur.execute(statement,{"0":CID})

counter = cur.fetchall()

count = len(counter)

if count > 0:

passwd = counter[0][0]

status = counter[0][1]

Bal = counter[0][2]

trial = 1

flag = 1

if status == "Unlocked":

while flag:

password = input("Enter your password")

if password == passwd and trial <= 3:

print("Successful Log in")

flag = 0

elif trial == 3:

print("You have entered an eroneous password 3 times consecutively.Your account is locked.Please contact bank for more details")

flag = 0

cur.execute("update CURRENT\_ACCOUNT set STATUS = 'Locked' where CUSTOMER\_ID = :0",{"0":CID})

con.commit()

else:

print("You have entered a wrong password. Try again")

trial = trial + 1;

else:

print("Your Account is Locked")

return

print("Enter the amount you wish to deposit")

amount = float(input())

if amount <= 0.0:

print("Invalid Amount")

else:

total = Bal + amount

statement = "update CURRENT\_ACCOUNT set BALANCE = :0 where CUSTOMER\_ID = :1"

cur.execute(statement,{"0":total,"1":CID})

con.commit()

statement = "insert into TRANSACTION\_CURRENT\_ACCOUNT(ACCOUNT\_ID,TRANSACTION\_TYPE,DATE\_OF\_TRANSACTION,AMOUNT\_UNDER\_TRANSACTION,BALANCE)values(:0,:1,:2,:3,:4)"

cur.execute(statement,{"0":CID,"1":"Credit","2":datetime.datetime.now(),"3":amount,"4":total})

con.commit()

print("Your new Balance is ",total)

else:

print("You may not have current account")

else:

print("Invalid Account number")

def Savings\_Account\_Withdraw(self,CID):

print("Enter your Account Number")

Anumber = input()

if Anumber == CID:

statement = "select PASSWORD,STATUS,BALANCE from SAVINGS\_ACCOUNT where CUSTOMER\_ID = :0"

result = cur.execute(statement,{"0":CID})

count = cur.fetchall()

counter = len(count)

if counter > 0:

passwd = count[0][0]

status = count[0][1]

Bal = count[0][2]

trial = 1

flag = 1

if status == "Unlocked":

while flag:

password = input("Enter your password")

if password == passwd and trial <= 3:

print("Successful Log in")

flag = 0

print("Enter the amount you wish to withdraw")

amount = float(input())

if amount > 0:

today\_date = datetime.datetime.now()

today\_day = datetime.datetime.today().day

back\_date = today\_date + datetime.timedelta(-(today\_day-1))

maxlimit = 30 - today\_day

front\_date = today\_date + datetime.timedelta((maxlimit))

statement = "select count(TRANSACTION\_TYPE) from TRANSACTION\_SAVINGS\_ACCOUNT where DATE\_OF\_TRANSACTION between :0 and :1 and TRANSACTION\_TYPE = 'Debit'"

result = cur.execute(statement,{"0":back\_date,"1":front\_date})

for each in result:

county = each

if county[0] <= 10:

if Bal-amount >=0:

print(amount," withdrawn Successfully")

print("Remaining Balance ",Bal-amount)

statement = "update SAVINGS\_ACCOUNT set BALANCE = :0 where CUSTOMER\_ID = :1"

cur.execute(statement,{"0":Bal-amount,"1":CID})

con.commit()

statement = "insert into TRANSACTION\_SAVINGS\_ACCOUNT(Account\_Id,Transaction\_Type,Date\_of\_Transaction,Amount\_Under\_Transaction,Balance)values(:0,:1,:2,:3,:4)"

cur.execute(statement,{"0":CID,"1":"Debit","2":datetime.datetime.now(),"3":amount,"4":Bal-amount})

con.commit()

else:

print("Insufficient Balance")

else:

print("Maximum Withdrawal Chances are 10.You have reached the limit")

else:

print("Invalid amount value")

elif trial == 3:

print("You have entered an eroneous password 3 times consecutively.Your account is locked.Please contact bank for more details")

flag = 0

cur.execute("update SAVINGS\_ACCOUNT set STATUS = 'Locked' where CUSTOMER\_ID = :0",{"0":CID})

con.commit()

else:

print("You have entered a wrong password. Try again")

trial = trial + 1;

else:

print("Your Account is Locked")

return

else:

print("You may not have a savings account")

else:

print("Invalid Account number")

def Current\_Account\_Withdraw(self,CID):

print("Enter your Account Number")

Anumber = input()

if Anumber == CID:

statement = "select PASSWORD,STATUS,BALANCE from CURRENT\_ACCOUNT where CUSTOMER\_ID = :0"

result = cur.execute(statement,{"0":CID})

count = cur.fetchall()

counter = len(count)

if counter > 0:

passwd = count[0][0]

status = count[0][1]

Bal = count[0][2]

trial = 1

flag = 1

if status == "Unlocked":

while flag:

password = input("Enter your password")

if password == passwd and trial <= 3:

print("Successful Log in")

flag = 0

print("Enter the amount you wish to withdraw")

amount = float(input())

if (amount > 0) and ((Bal-amount) >= 5000):

print("Amount Withdrawn successfully")

print("Balance after transaction is ",Bal-amount)

statement = "update CURRENT\_ACCOUNT set Balance = :0 where Customer\_Id = :1"

cur.execute(statement,{"0":Bal-amount,"1":CID})

con.commit()

statement = "insert into TRANSACTION\_CURRENT\_ACCOUNT(Account\_Id,Transaction\_Type,Date\_of\_Transaction,Amount\_Under\_Transaction,Balance)values(:0,:1,:2,:3,:4)"

cur.execute(statement,{"0":CID,"1":"Debit","2":datetime.datetime.now(),"3":amount,"4":Bal-amount})

con.commit()

elif amount < 0:

print("Invalid amount value. Enter a valid amount")

return

else:

print("Minimum balance of 5000 has to be maintained.Not happening for mentioned amount")

return

elif trial == 3:

print("You have entered an eroneous password 3 times consecutively.Your account is locked.Please contact bank for more details")

flag = 0

cur.execute("update CURRENT\_ACCOUNT set STATUS = 'Locked' where CUSTOMER\_ID = :0",{"0":CID})

con.commit()

else:

print("You have entered a wrong password. Try again")

trial = trial + 1;

else:

print("Your Account is Locked")

return

else:

print("You may not have a current account")

else:

print("Invalid Account number")

def Print\_Statement(self,CID):

print("Enter your Account ID")

AID = input()

if CID == AID:

print("Enter the account whose statement you want to access.Eg.Savings or Current")

account\_type = input()

if (account\_type.upper() != 'SAVINGS') and (account\_type.upper() != 'CURRENT'):

print("Enter a valid account type")

if (account\_type.upper() == 'SAVINGS'):

print("Enter the Dates duration between which you wish to view account statement")

print("Date From(YYYY-MM-DD)")

date\_f = input()

date\_from = datetime.datetime.strptime(date\_f,"%Y-%m-%d")

print("Date to(YYYY-MM-DD)")

date\_t = input()

date\_to = datetime.datetime.strptime(date\_t,"%Y-%m-%d")

if (date\_to > date\_from):

statement = "select Transaction\_Type,Date\_of\_Transaction,Amount\_Under\_Transaction,Balance from TRANSACTION\_SAVINGS\_ACCOUNT where Account\_Id = :0 and Date\_of\_Transaction Between :1 and :2"

res = cur.execute(statement,{"0":AID,"1":date\_from,"2":date\_to})

count = cur.fetchall()

counter = len(count)

if counter <= 0:

print("No Transaction")

else:

print("Your Savings account transaction")

print("Type Transaction\_Date Amount Balance")

i = 0

while counter != 0:

print(count[i][0]," ",count[i][1].strftime("%Y-%m-%d")," ",count[i][2]," ",count[i][3])

i = i + 1

counter = counter - 1

else:

print("Enter valid date range")

if (account\_type.upper() == 'CURRENT'):

print("Enter the Dates duration between which you wish to view account statement")

print("Date From(YYYY-MM-DD)")

date\_f = input()

date\_from = datetime.datetime.strptime(date\_f,"%Y-%m-%d")

print("Date to(YYYY-MM-DD")

date\_t = input()

date\_to = datetime.datetime.strptime(date\_t,"%Y-%m-%d")

if (date\_to > date\_from):

statement = "select Transaction\_Type,Date\_of\_Transaction,Amount\_Under\_Transaction,Balance from TRANSACTION\_CURRENT\_ACCOUNT where Account\_Id = :0 and Date\_of\_Transaction Between :1 and :2"

res = cur.execute(statement,{"0":AID,"1":date\_from,"2":date\_to})

counter = cur.fetchall()

count = len(counter)

if count <= 0:

print("No Transaction")

else:

print("Your Current account transaction")

print("Type Transaction\_Date Amount Balance")

i = 0

while count != 0:

print(counter[i][0]," ",counter[i][1].strftime("%Y-%m-%d")," ",counter[i][2]," ",counter[i][3])

i = i + 1

count = count - 1

else:

print("Enter valid date range")

else:

print("Enter a valid account number")

def Account\_Closure(self,CID,Account\_type):

if Account\_type == 'SAVINGS':

statement = "select Line1,Line2,City,State,Pincode,Balance from CUSTOMERS,SAVINGS\_ACCOUNT where CUSTOMERS.Customer\_Id = :0 and CUSTOMERS.Customer\_Id = SAVINGS\_ACCOUNT.Account\_Id";

result = cur.execute(statement,{"0":CID})

for single in result:

eachresult = single

print("Balance ",eachresult[5]," will be sent to the below Addresss")

print(eachresult[0]," ",eachresult[1]," ",eachresult[2]," ",eachresult[3]," ",eachresult[4])

statement = "insert into CLOSED\_ACCOUNT\_NEW(Account\_ID,Time\_Closed,Account\_Type) values (:0,:1,:2)"

cur.execute(statement,{"0":CID,"1":datetime.datetime.now(),"2":Account\_type})

con.commit()

statement = "delete from SAVINGS\_ACCOUNT where Customer\_Id = :0"

cur.execute(statement,{"0":CID})

con.commit()

statement = "delete from TRANSACTION\_SAVINGS\_ACCOUNT where Account\_Id = :0"

cur.execute(statement,{"0":CID})

con.commit()

print("Account Closed Successfully")

if Account\_type == 'CURRENT':

statement = "select Line1,line2,City,State,Pincode,Balance from CUSTOMERS,CURRENT\_ACCOUNT where CUSTOMERS.Customer\_Id = :0 and CUSTOMERS.Customer\_Id = CURRENT\_ACCOUNT.Account\_Id";

res = cur.execute(statement,{"0":CID})

for each in res:

eachresult = each

print("Balance ",eachresult[5]," will be sent to the below Addresss")

print(eachresult[0]," ",eachresult[1]," ",eachresult[2]," ",eachresult[3]," ",eachresult[4])

statement = "insert into CLOSED\_ACCOUNT\_NEW(Account\_ID,Time\_Closed,Account\_Type) values (:0,:1,:2)"

cur.execute(statement,{"0":CID,"1":datetime.datetime.now(),"2":Account\_type})

con.commit()

statement = "delete from CURRENT\_ACCOUNT where Customer\_Id = :0"

cur.execute(statement,{"0":CID})

con.commit()

statement = "delete from TRANSACTION\_CURRENT\_ACCOUNT where Account\_Id = :0"

cur.execute(statement,{"0":CID})

con.commit()

print("Account Closed Successfully")

def Avail\_Loan(self,CID):

statement = "select Balance from SAVINGS\_ACCOUNT where Customer\_Id = :0"

result = cur.execute(statement,{"0":CID})

for each in result:

Balance = each[0]

print("Enter the Loan Amount You wish to avail.Enter in multiples of 1000")

required\_amount = (float(input()))

print("Enter the term")

term = (int(input()))

if ((required\_amount % 1000 == 0) and (term > 0)):

eligibility = 2 \* Balance

if required\_amount <= eligibility:

print("A loan Amount of ",required\_amount," is granted")

first\_num = "34560"

last\_num = str(random.randint(1,100000))

loan\_num = first\_num + last\_num

statement = "insert into LOAN\_ACCOUNT(Account\_Id,Customer\_Id,Loan\_Amount,Term)values(:0,:1,:2,:3)"

cur.execute(statement,{"0":loan\_num,"1":CID,"2":required\_amount,"3":term})

con.commit()

else:

print("Sorry.You are not eligible for this amount.You can avail a maximum loan amount of " ,eligibility)

else:

print("Invalid input")

class Admin:

def \_\_init\_\_(self):

self.\_\_Admin\_Id = "5125011111"

self.\_\_Password = "sharp@123"

self.validate()

def validate(self):

flag = 1

while flag:

print("Enter Admin\_Id")

ID = input()

print("Enter Password")

passwd = input()

if ID == self.\_\_Admin\_Id and passwd == self.\_\_Password:

print("Admin Login Successfull")

flag = 0

self.sub\_menu()

else:

print("Invalid login credentials")

def sub\_menu(self):

flag = 1

while flag:

print("\t\t\t\t\tADMIN-MENU")

print("\t\t\t\t1.Print Closed Accounts")

print("\t\t\t\t2.FD Report of a Customer")

print("\t\t\t\t3.FD Report of a Customer vis-a-vis another customer")

print("\t\t\t\t4.FD Report with respect to a particular FD amount")

print("\t\t\t\t5.Loan Report of a Customer")

print("\t\t\t\t6.Loan Report of a Customer vis-a-vis another customer")

print("\t\t\t\t7.Loan Report with respect to particular loan amount")

print("\t\t\t\t8.Loan-FD Report of customers")

print("\t\t\t\t9.Report of Customer who are yet to avail loan")

print("\t\t\t\t10.Report of Customers who are yet to open FD account")

print("\t\t\t\t11.Report of Customers who neither have loan or FD account")

print("\t\t\t\t12.Admin Logout")

print("Enter your choice")

choice = input()

if choice == "1":

a = Admin\_Services()

a.View\_Closed\_Accounts()

flag = 1

if choice == "2":

a = Admin\_Services()

a.FD\_Report\_Customer()

flag = 1

if choice == "3":

a = Admin\_Services()

a.FD\_Customer\_vis\_Customer()

flag = 1

if choice == "4":

a = Admin\_Services()

a.FD\_Report\_wrt\_Amount()

flag = 1

if choice == "5":

a = Admin\_Services()

a.Loan\_Report\_Customer()

flag = 1

if choice == "6":

a = Admin\_Services()

a.Loan\_Report\_vis\_a\_vis\_Customer()

flag = 1

if choice == "7":

a = Admin\_Services()

a.Loan\_Report\_wrt\_Amount()

flag = 1

if choice == "8":

a = Admin\_Services()

a.Loan\_FD\_Report()

flag = 1

if choice == "9":

a = Admin\_Services()

a.Customers\_Yet\_to\_Avail\_Loan()

flag = 1

if choice == "10":

a = Admin\_Services()

a.Customers\_Yet\_to\_Deposit()

flag = 1

if choice == "11":

a = Admin\_Services()

a.Customers\_Neither\_FD\_nor\_Loan()

flag = 1

if choice == "12":

print("Admin Logout was sucessfull")

flag = 0

class Admin\_Services:

def View\_Closed\_Accounts(self):

statement = "select \* from Closed\_Account\_New"

res = cur.execute(statement)

print("Account\_ID\tClosed\_Time\tAccount\_Type")

for each in res:

print(each[0]," ",each[1].strftime("%Y-%m-%d")," ",each[2])

print("\n")

def FD\_Report\_Customer(self):

print("Enter the Customer\_Id whose FD Report you wish to view")

Cust\_Id = input()

statement = "select Account\_Id, Money\_Deposited, Term from FIXED\_DEPOSIT where Customer\_Id = :0"

result = cur.execute(statement,{"0":Cust\_Id})

count = cur.fetchall()

counter = len(count)

i=0

if counter > 0:

print("FD Account \t Amount\_Deposited \t Term")

while (counter > 0):

print(count[i][0]," \t ",count[i][1]," \t ",count[i][2])

i = i + 1

counter = counter - 1

else:

print("Not applicable. No FD accounts available for this customer")

def FD\_Customer\_vis\_Customer(self):

print("Enter the Customer\_Id")

Cust\_Id = input()

statement = "select Account\_Id from FIXED\_DEPOSIT where Customer\_Id = :0"

result = cur.execute(statement,{"0":Cust\_Id})

count = cur.fetchall()

counter = len(count)

if counter > 0:

print("Customer\_Id\tFD\_Account\_Id\t Deposit\_Amount\tTerm")

statement = "select distinct Customer\_Id,Account\_Id,Money\_Deposited,Term from FIXED\_DEPOSIT where Money\_Deposited >=(select sum(Money\_Deposited) from FIXED\_DEPOSIT where Customer\_Id = :0)"

result = cur.execute(statement,{"0":Cust\_Id})

for number in result:

print(number[0],"\t",number[1],"\t",number[2],"\t",number[3])

else:

print("No rows were found")

def FD\_Report\_wrt\_Amount(self):

print("Enter a valid FD amount in multiples of 1000")

amount = float(input())

if ((amount % 1000 == 0) and amount > 0):

statement = "select distinct c.Customer\_Id, c.First\_Name,c.Last\_Name,d.Money\_Deposited from CUSTOMERS c , FIXED\_DEPOSIT d where c.Customer\_Id = d.Customer\_Id and d.Money\_Deposited > :0"

result = cur.execute(statement,{"0":amount})

count = cur.fetchall()

counter = len(count)

i = 0

if counter > 0:

print("FD\_ANUMBER \t First\_Name \t Last\_Name \t Money\_Deposited")

while(counter > 0):

print(count[i][0]," \t ",count[i][1]," \t ",count[i][2]," \t ",count[i][3])

i = i +1

counter = counter - 1

else:

print("No Entries Available")

else:

print("Enter valid amount.A non-negative amount in multiples of 1000")

def Loan\_Report\_Customer(self):

print("Enter the Customer\_Id")

Cust\_Id = input()

statement = "select Account\_Id , Loan\_Amount , Term from LOAN\_ACCOUNT where Customer\_Id = :0"

cur.execute(statement,{"0":Cust\_Id})

count = cur.fetchall()

counter = len(count)

i=0

if counter > 0:

print("Loan\_Account\_Id \t Loan\_Amount \t Repayment\_Term")

while(counter > 0):

print(count[i][0]," \t\t",count[i][1]," \t\t ",count[i][2])

i = i + 1

counter = counter - 1

else:

print("Loan is not availed by this customer")

def Loan\_Report\_vis\_a\_vis\_Customer(self):

print("Enter the Customer\_Id")

Cust\_Id = input()

statement = "select Account\_Id from LOAN\_ACCOUNT where Customer\_Id = :0"

result = cur.execute(statement,{"0":Cust\_Id})

count = cur.fetchall()

counter = len(count)

if counter > 0:

print("Customer\_Id\tLoan\_Account\_Id\t Loan\_Amount\tRepayment-Term")

statement = "select distinct Customer\_Id,Account\_Id,Loan\_Amount,Term from LOAN\_ACCOUNT where Loan\_Amount >=(select sum(Loan\_Amount) from LOAN\_ACCOUNT where Customer\_Id = :0)"

result = cur.execute(statement,{"0":Cust\_Id})

for number in result:

print(number[0],"\t",number[1],"\t ",number[2],"\t",number[3])

else:

print("No rows were matched")

def Loan\_Report\_wrt\_Amount(self):

print("Enter a valid amount in multiples of 1000")

amount = float(input())

if ((amount % 1000 == 0) and amount > 0):

statement = "select distinct c.Customer\_Id, c.First\_Name,c.Last\_Name,l.Loan\_Amount from CUSTOMERS c ,LOAN\_ACCOUNT l where c.Customer\_Id = l.Customer\_Id and l.Loan\_Amount > :0"

result = cur.execute(statement,{"0":amount})

count = cur.fetchall()

counter = len(count)

i = 0

if counter > 0:

print("Customer\_ID \t First\_Name \t Last\_Name \t Loan\_Availed")

while(counter > 0):

print(count[i][0]," \t ",count[i][1]," \t ",count[i][2]," \t ",count[i][3])

i = i +1

counter = counter - 1

else:

print("No Entries Available")

else:

print("Enter valid amount.A non-negative amount in multiples of 1000")

def Loan\_FD\_Report(self):

stmt\_1 = "create or replace view LOAN\_VIEW as select Customer\_Id, sum(Loan\_Amount) as Sum\_Loan from LOAN\_ACCOUNT group by Customer\_Id"

cur.execute(stmt\_1)

stmt\_2 = "create or replace view FIXED\_VIEW as select Customer\_Id, sum(Money\_Deposited) as Sum\_Deposit from FIXED\_DEPOSIT group by Customer\_Id"

cur.execute(stmt\_2)

statement = "select \* from LOAN\_VIEW inner join FIXED\_VIEW on LOAN\_VIEW.Customer\_Id = FIXED\_VIEW.Customer\_Id and SUM\_LOAN > SUM\_DEPOSIT"

result = cur.execute(statement)

print("Customer\_Id \t SUM\_LOAN\_AMOUNT \t SUM\_FIXED\_DEPOSIT")

for entry in result:

print(entry[0],"\t",entry[1],"\t\t\t ",entry[3])

def Customers\_Yet\_to\_Avail\_Loan(self):

statement = "select c.Customer\_Id, c.First\_Name, c.Last\_Name from CUSTOMERS c where c.Customer\_Id not in (select distinct Customer\_Id from LOAN\_ACCOUNT)"

table = cur.execute(statement)

print("Customer\_ID \t First\_Name \t Last\_Name")

for each\_entry in table:

print(each\_entry[0],"\t",each\_entry[1],"\t ",each\_entry[2])

def Customers\_Yet\_to\_Deposit(self):

statement = "select c.Customer\_Id, c.First\_Name, c.Last\_Name from CUSTOMERS c where c.Customer\_Id not in (select distinct Customer\_Id from FIXED\_DEPOSIT)"

table = cur.execute(statement)

print("Customer\_ID \t First\_Name \t Last\_Name")

for each\_entry in table:

print(each\_entry[0],"\t",each\_entry[1],"\t ",each\_entry[2])

def Customers\_Neither\_FD\_nor\_Loan(self):

statement = "select c.Customer\_ID,c.First\_Name,c.Last\_Name from CUSTOMERS c where c.Customer\_Id not in(select unique customer\_id from LOAN\_ACCOUNT) and c.Customer\_Id not in (select unique customer\_id from FIXED\_DEPOSIT)"

table = cur.execute(statement)

print("Customer\_ID \t First\_Name \t Last\_Name")

for each\_entry in table:

print(each\_entry[0],"\t",each\_entry[1],"\t ",each\_entry[2])

checker = True

while checker:

print("\t\t\t\tMAIN MENU")

print("\t\t\t1.Sign Up(New Customer)")

print("\t\t\t2.Sign In(Existing Customer)")

print("\t\t\t3.Admin Sign")

print("\t\t\t4.Quit")

print("Enter your choice")

choice = input()

if choice == "1":

cust = New\_Customer()

if choice == "2":

Ecust = Existing\_Customer()

if choice == "3":

admin = Admin()

if choice == "4":

print("ByeBye.Have a Good Day")

checker = False

cur.close()

con.close()