

Moneta Bank

Project Developed by

C. B. Ananya – XII ‘A’

**BONAFIDE CERTIFICATE**

*Certified to be the Bonafide record of work done by*

*Miss / Master* **C. B. ANANYA** *of Class* **XII ‘A’**

in **SRI SANKARA SENIOR SECONDARY SCHOOL**, Chennai

During the Year 2020-21

Dated \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Subject Teacher \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Chennai

Submitted for Practical Examination

held in COMPUTER SCIENCE­­­­­­­­­

at SRI SANKARA SENIOR SECONDARY SCHOOL, Chennai

Dated \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Examiner

Seal

**Acknowledgement**

I would like to express my heartfelt thanks and gratitude to my Computer Science teachers Mrs. R. Revathy, Mrs. S. Vidyalakshmi, Lab Support Mr. Jose Cherian and our Principal Ma’am Mrs. Mita Venkatesh who gave me the golden opportunity to do this informative project on the topic **Moneta Bank**, an Online Banking Platform, which helped me in knowing in-depth about Python and MySQL and also kindled my interest in doing a lot of research through which I came to know about banking functions and many new computer applications.

Secondly, I would also like to thank my parents, family and friends who have helped me a lot in finalizing this project within the limited time frame.

Index

[**Scope of Moneta Bank project** 5](#_Toc65278558)

[**Savings Bank Account** 5](#_Toc65278559)

[**Fixed Deposit Account** 5](#_Toc65278560)

[**Student Loan** 6](#_Toc65278561)

[**Software used - Backend and Front end tools** 7](#_Toc65278562)

[**Details of the text / binary / csv files used** 12](#_Toc65278563)

[**Details of MySQL Tables** 14](#_Toc65278564)

[**Code** 20](#_Toc65278565)

[**Screenshots in Python Platform** 55](#_Toc65278566)

[**Future Improvements** 88](#_Toc65278567)

[**Bibliography** 89](#_Toc65278568)

**Synopsis**

# **Scope of Moneta Bank project**

The project aims to develop an online banking platform where following banking functions can be serviced through this computer application in Python and MySQL:

1. Savings Bank Account (Deposits)
2. Fixed Deposit Account (Deposits)
3. Student Loan (Lending)

## **Savings Bank Account**

Customers can deposit and withdraw money by having this account. An interest amount of 3.3% p.a. will be paid on the balance – balance taken at close of each business day. A minimum balance of Rs.500/- is to be maintained at all times.

This is a basic account facility which is mandatory to avail other banking products like FD and Student Loans, which features are optional.

## **Fixed Deposit Account**

Customers can deposit amount at a higher interest rate than Savings Bank account for Fixed Tenures, given in FD Interest Table of the Bank.

Interest rate varies based on the tenure of deposit.

Minimum Amount of Rs.1,000/- is required to start an FD Account.

Premature Withdrawal is permitted with a penalty amount charged at the time of withdrawal if FD is broken before Date of Maturity.

Terms and Conditions for Premature Withdrawal are as follows:

1. Breaking of fixed deposit any time before maturity date would cost the customer a penalty in the form of **1% reduction in interest rate offered** from date of opening to date of premature withdrawal.

## **Student Loan**

Students can avail Education Loan from College Level through Moneta Bank at Concessional Rate as per Govt. Rules.

Bank would sanction Education Loan for a Principal Loan Amount from **Rs.50,000/-** to **Rs. 1,00,00,000/-** subject to background checks.

Students can choose tenure of Repayment from 1 Year to maximum of 7 Years.

Repayment starts from 1 Year (max.) from the time of completion of degree term.

A Status Flag would be displayed indicating whether the Loan Repayment has Started or Not, if started - Loan Repayment In Progress or Completed.

Once Repayment schedule commences, payment from the student is accepted by the Bank in the form of EMI. Amount pending to close the loan would be displayed after each payment along with the Number of Installments Expected and Number of Installments Paid. Number of Expected Installments would increase by 1 at the end of each month of repayment Tenure. (In an ideal scenario, these two numbers would be equal, implying that there is no default as of current date).

If a payment is defaulted after the start of the Repayment, number of such defaults would be shown till payment is cleared.

More than three consecutive defaults would lead to revised (increased) EMI and Interest Rate.

# **Software used - Backend and Front end tools**

The software used are IDLE (Python 3.7 32-bit) and MySQL 5.6-Command Line Client-Unicode.

**Front-End Tools: Python**

The following modules offered by Python have been utilized in this project.

1. Tkinter – The Graphical User Interface (GUI) with multiple screens linked together have been created using this module.
2. Pillow (PIL) – Various Images and logos have been inserted into the screens using pillow module.
3. datetime – This module is of immense utility for conversion of date into various formats for compatibility between Python and MySQL as well as for numerous financial calculations.
4. os – This module is used for handling files in different directories.
5. mysql.connector – This is the primary module that establishes a link between Python Platform and MySQL (where the data is stored). Stored data is retrieved from MySQL tables and entered data is updated in the tables using the Python program aided by this module.
6. Random – This module is specifically used for the following two purposes:
7. Generation of CustomerID and Account Numbers:

Each time a new (12-digit) CustomerID or account number is generated, it checks whether the number already exists for any account and then it is fixed for the customer.

1. Generation of Random Number associated with encrypted password:

This project involves encryption of password using a type of **Caesar Cipher Algorithm**

**Working of this algorithm:**

**Encryption logic:**

A random number between 1 and 127 is selected and stored. This number is saved and stored for decryption during login and verification of password.

A function which takes the random number and a single character as parameters is defined.

Every character in the password along with the random number chosen (random number is the same for each iteration) are passed as parameters into the function using a “for” loop.

Inside the function, the ASCII value of the input character is stored and the random number is subtracted from this value.

If the resultant number is less than 0, then 255 is added to it, otherwise it is retained as it is.

**Significance of ASCII 255: ASCII VALUES RANGE FROM 0 TO 255 ONLY.**

The function returns the final number.

The final number lies between 0 and 255 with a valid character associated with it. The range of numbers (1 to 127) from which random number is chosen is also designed based on this requirement.

The character having this final number as ASCII value is then obtained.

This process happens for every character of the customer’s password and the output characters are added one after the other, to an initially empty string.

Output String / Encrypted Password = Concatenation of all Output Characters.

**Decryption Logic:**

This logic is simply the reverse of encryption.

The random number generated and saved during encryption along with every character of the encrypted password are passed as parameters into a function.

Inside the function, ASCII value of the input character from encrypted password is got.

The random number is added to it. (The resultant is greater than 0).

If the resultant is greater than 255 (there would be no corresponding character, as ASCII values range from 0 to 255 only), then 255 is subtracted from it to have a valid character associated.

If it is less than 255, then it is retained as it is.

The function returns the resultant number for each input encrypted character that enters the loop.

The character having this resultant number as ASCII value is then obtained.

This process happens for every character of the encrypted password and the output characters are added one after the other, to an initially empty string to generate the original password.

Output Decrypted password (Entered by Customer) = Concatenation of each Decrypted Character.

Only the encrypted version of a customer’s password is stored in the MySQL databases. The actual password as given by the customer during SignUp process is never visible in any of the tables at any point of time, thus adding to the security layer of the banking system.

The password given during SignUp is encrypted, stored and saved in MySQL tables. Each time Customer tries to Login, he/she will type their password (given during SignUp) and verification will be done as to whether the decrypted version of the password in the tables for the given CustomerID matches with the password that the Customer entered.

If yes, User successfully logs in, else the Customer would have entered the wrong password and an error message is displayed.

**Back-End Tools: MySQL, Text files**

MySQL is used to store the data of customers and the status of various features utilized by them in the form of tables in one database.

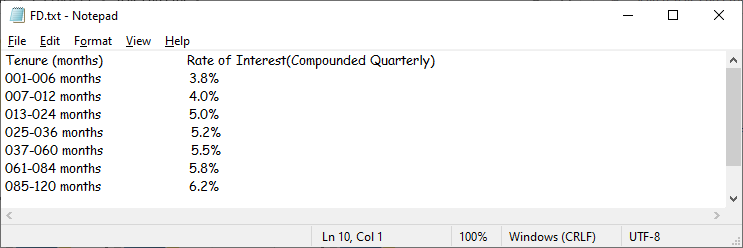
Text files are used to display the Terms and Conditions of various features at different points of time in the user interface of the application.

# **Details of the text / binary / csv files used**

Following are the text files used in the project. The Terms and Conditions of various products of the bank are stored in these text files. The details in these text files are retrieved by the Python Program and displayed in the User Interface at various points of time.

**FD.txt**

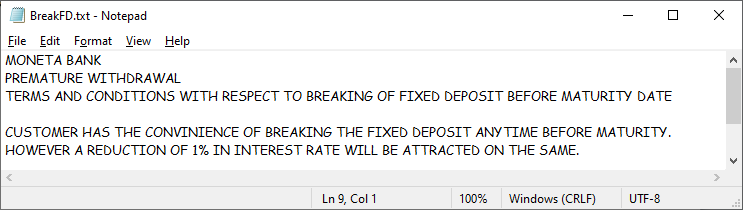
This text file has the Terms and Conditions of Moneta Bank regarding Fixed Deposit.



This same tabulation is shown in the User Interface when a Customer chooses to make a Fixed Deposit.

**BreakFD.txt**

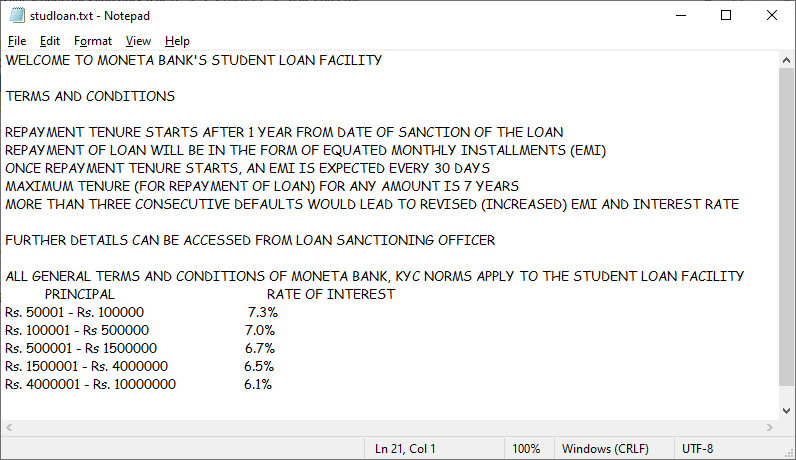
This file contains the Terms and Conditions of Moneta Bank regarding Premature Withdrawal/ Breaking of Fixed Deposit.



Details in this file (BreakFD.txt) will be visible in the application during both deposit as well as Premature Withdrawal of the Fixed Deposit.

**studloan.txt**

This text file has details of Terms and Conditions regarding the Student Loan. These details will be displayed in the application when a customer wishes to avail the same. It contains details regarding the Means of Repayment and the Interest Rates for a wide range of Principal Amounts starting from Rs. 50,001/- to Rs. 1,00,00,000/-



# **Details of MySQL Tables**

The database used in MySQL is named “BANKING”.

It has 5 tables:

1. login
2. type\_acc
3. sb
4. fd
5. studloan

* “login” table: Contains basic details regarding all the Customers who have signed up.

It contains the following Fields

* + Customer ID (CustID—Primary Key),
  + Customer Name (Name),
  + Gender,
  + Date Of Birth (dob),
  + Encrypted password (password)
  + Random Number associated with encryption (RandomNumber)

In MySQL: Description of the table

mysql> desc login;

+--------------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+--------------+-------------+------+-----+---------+-------+

| CustID | bigint(12) | NO | PRI | 0 | |

| Name | varchar(25) | NO | | NULL | |

| Gender | char(1) | YES | | NULL | |

| dob | date | YES | | NULL | |

| password | varchar(50) | YES | | NULL | |

| RandomNumber | int(3) | YES | | NULL | |

+--------------+-------------+------+-----+---------+-------+

* “type\_acc” table: This table has details regarding the facilities utilized by each customer.

It has the following fields:

* CustomerID (CustID—Primary Key),
* Customer Name (Name),
* Fixed Deposit (FD),
* Savings Bank Account (SB) and
* Student Loan (StudLoan).

In FD, SB and StudLoan columns, a record has ‘N’/’Y’ as values.

‘Y’ in a column implies that the customer has availed the feature and ‘N’ in a column implies that the customer is currently not availing the feature. (But customer may have used the feature in the past and might have completed full usage of the feature.

Ex: Customer may have done a Fixed Deposit and obtained the amount on Maturity and the account may have been closed later. Thus, currently the type\_acc table would show ‘N’ under FD for that customer).

In MySQL: Description of the table

mysql> desc type\_acc;

+----------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------+-------------+------+-----+---------+-------+

| CustID | bigint(12) | NO | PRI | 0 | |

| Name | varchar(25) | YES | | NULL | |

| FD | char(1) | YES | | N | |

| SB | char(1) | YES | | N | |

| StudLoan | char(1) | YES | | N | |

+----------+-------------+------+-----+---------+-------+

* “sb” table: contains details regarding customers having Savings Bank Account in Moneta Bank. Only after creation of Savings Bank Account, can a customer utilize other features. When this account is created, the ‘type\_acc’ table will have value ‘Y’ under ‘sb’ column.

It has the following details:

* Customer ID (CustID),
* Name of the customer (Name),
* Account Number (Acc\_No—Primary Key),
* Date Of Birth (dob),
* Date Of Opening of the Account (DateOfOpening),
* Current Principal (Principal),
* Interest Amount (Interest),
* Date Of Latest Quarterly Addition of Interest to Principal (DateOfLatestQR).

.

In MYSQL: Description of the table

mysql> desc sb;

+----------------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------------+-------------+------+-----+---------+-------+

| CustID | bigint(12) | YES | | NULL | |

| Name | varchar(25) | YES | | NULL | |

| dob | date | YES | | NULL | |

| DateOfOpening | date | YES | | NULL | |

| Principal | bigint(25) | YES | | NULL | |

| Acc\_No | bigint(12) | NO | PRI | NULL | |

| Interest | bigint(15) | YES | | 0 | |

| DateOfLatestQR | date | YES | | NULL | |

+----------------+-------------+------+-----+---------+-------+

Principal keeps changing quarterly.

Interest amount on Current Principal is calculated at the close of each business day and added to the Interest column every day. Once in 3 months from opening of account, the Interest component is added to the Principal and the Interest column is made zero. The date on which this is done every time is updated in the ‘Date Of Latest Quarterly Addition’ column. This process continues freshly for the next 3 months.

During creation of account, the Date of Latest Quarterly Addition Column has the same date as Date Of Opening of Account.

* “fd” table: contains details of Customers who are currently utilizing the Fixed Deposit facility of Moneta Bank.

It has the fields:

* Customer ID (CustID),
* Name of the Customer (Name),
* Date Of Birth (dob),
* Account Number (FDAcc\_No—Primary KEY),
* Status of the FIXED DEPOSIT -Whether running, broken before maturity or matured (FD\_Status)
* Date Of Opening of account (DateOfOpening),
* Principal amount deposited (Principal\_at\_open),
* Tenure,
* Rate of Interest (RoI),
* Current Principal (Principal),
* Interest Amount Accumulated (FDInterest),
* Maturity Value of the Fixed Deposit (MaturityValue),
* Date of Maturity of FD (MaturityDate),
* Date Of Latest Quarterly addition of Interest to Current Principal (DateOfLatestQR).
* Amount obtained on Breaking of FD if the Fixed Deposit was Broken (If\_broken\_amount)

Current Principal keeps changing quarterly. Interest on Current Principal is added on a daily basis for 3 months after which the Interest accumulated is added to Current Principal and Interest column is made zero. (Similar to Savings account but with higher Interest Rate).

At the end of the Tenure, the current Principal would be equal to Maturity Value.

The Current Principal at any point of time is not the same as the amount that will be obtained if the FD is broken. Amount obtained by customer on Breaking FD would be lesser than Current Principal due to penalty charges as mentioned in Terms and Conditions of Premature Withdrawal.

If the deposit is not broken, the field If\_broken\_amount has value -1.

If it is broken, it has the value that the customer obtains on breaking of FD.

In MySQL: Description of the table:

mysql> desc fd;

+-------------------+---------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------------------+---------------+------+-----+---------+-------+

| CustID | bigint(12) | YES | | NULL | |

| Name | varchar(25) | YES | | NULL | |

| dob | date | YES | | NULL | |

| DateOfOpening | date | YES | | NULL | |

| Principal\_at\_open | float | YES | | NULL | |

| Principal | decimal(12,2) | YES | | NULL | |

| FDAcc\_No | bigint(12) | NO | PRI | NULL | |

| Tenure | int(3) | YES | | NULL | |

| RoI | float | YES | | NULL | |

| FDInterest | decimal(12,2) | YES | | NULL | |

| MaturityValue | decimal(12,2) | YES | | NULL | |

| MaturityDate | date | YES | | NULL | |

| DateOfLatestQR | date | YES | | NULL | |

| FD\_Status | varchar(40) | YES | | NULL | |

| If\_broken\_amount | decimal(12,2) | YES | | NULL | |

+-------------------+---------------+------+-----+---------+-------+

* “studloan” table: This table contains details of the Customers who are presently availing the Student Loan Facility.

It has the columns:

* Customer ID (CustID),
* Name of the Customer (Name)
* Loan Account Number (Loan\_Acc\_No—Primary Key)
* Date Of Sanction of Loan (DateOfSanction)
* Principal amount borrowed (Loan\_Principal)
* Tenure of Repayment of Loan (Loan\_Tenure)
* Rate Of Interest applicable (Loan\_RoI)
* Each Equated Monthly Installment/ EMI (Loan\_EMI)
* Total Repayment amount (Loan\_totpay)
* Amount paid (Loan\_paid)
* Remaining Balanace (Loan\_balance)
* Number of Installments Paid/ done (Loan\_InsD)
* Number of Installments Expected Till Date (Loan\_InsEx)
* Number of Installments Defaulted (Loans\_defaults)
* Status of Repayment of loan (Loan\_status)

In MySQL: Description of the table:

mysql> desc studloan;

+----------------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------------+-------------+------+-----+---------+-------+

| CustID | bigint(12) | YES | | NULL | |

| Name | varchar(25) | YES | | NULL | |

| Loan\_Acc\_No | bigint(16) | NO | PRI | NULL | |

| DateOfSanction | date | YES | | NULL | |

| Loan\_Principal | float | YES | | NULL | |

| Loan\_Tenure | int(2) | YES | | NULL | |

| Loan\_RoI | float | YES | | NULL | |

| Loan\_EMI | int(9) | YES | | NULL | |

| Loan\_totpay | float | YES | | NULL | |

| Loan\_paid | float | YES | | NULL | |

| Loan\_balance | float | YES | | NULL | |

| Loan\_InsD | int(3) | YES | | NULL | |

| Loan\_InsEx | int(3) | YES | | NULL | |

| Loan\_defaults | int(3) | YES | | NULL | |

| Loan\_status | varchar(20) | YES | | NULL | |

+----------------+-------------+------+-----+---------+-------+

The Status column will have value “NOT STARTED” for a period of 1 year from the Date Of Sanction of Loan, after which it will show “STARTED AND RUNNING”. Once the Loan is repaid, the status will become “REPAID”.

During Repayment, each time the Customer pays an EMI, the paid amount in “Loan\_paid” column will increase by an amount of one installment of the customer’s loan and the remaining balance in “Loan\_balance” column will decrease by an amount of one installment. Number of Installments done (Loan\_InsD) will increase by 1 unit.

# **Code**

BankingProject-MonetaBank-UserInterface.py

# Importing Modules #

from tkinter import \*

from datetime import \*

from PIL import ImageTk, Image

import datetime as dt

import os

import random

import mysql.connector as sqltor

# mysql connection object #

mycon=sqltor.connect(host="localhost", user="root", password="mysql", database="banking")

# Cursor Objects #

cursor1= mycon.cursor()

cursor2= mycon.cursor()

cursor3= mycon.cursor()

cursor4= mycon.cursor()

cursor5= mycon.cursor()

cursor6= mycon.cursor()

cursor7= mycon.cursor()

cursor8= mycon.cursor()

cursor9= mycon.cursor()

cursor10= mycon.cursor()

cursor11= mycon.cursor()

cursor12= mycon.cursor()

cursor13= mycon.cursor()

cursor14= mycon.cursor()

cursor15= mycon.cursor()

cursor16= mycon.cursor()

cursor17= mycon.cursor()

cursor18= mycon.cursor()

cursor19= mycon.cursor()

cursor20= mycon.cursor()

cursor21= mycon.cursor()

cursor22= mycon.cursor()

cursor23= mycon.cursor()

cursor24= mycon.cursor()

cursor25= mycon.cursor()

cursor26= mycon.cursor()

cursor27= mycon.cursor()

cursor28= mycon.cursor()

cursor29= mycon.cursor()

cursor30= mycon.cursor()

master= Tk()

master.title('ONLINE BANK')

master.config(bg="#bbf5ae")

## leaving signup screen ##

def leave\_signup():

signup\_screen.destroy()

## Finishing SignUp process through validation ##

def finish\_signup():

name = temp\_name.get()

gender= temp\_gender.get()

dob= temp\_dob.get()

password = temp\_password.get()

if name =="" or gender=="" or dob=="" or password=="" :

notif.config(fg="red", text="ALL FIELDS REQUIRED \* ")

return

if gender not in 'FMfm':

notif.config(fg="red", text="INVALID INPUT IN GENDER FIELD")

return

try:

dob\_as\_date\_in\_py = dt.datetime.strptime(dob, "%Y-%m-%d")

dob\_as\_date\_in\_py = dob\_as\_date\_in\_py.date()

except:

notif.config(fg="red", text="INVALID INPUT IN DATE FIELD")

return

if len(password)<8:

notif.config(fg="red", text="MINIMUM 8 CHARACTERS REQUIRED FOR PASSWORD")

return

if " " in password:

notif.config(fg="red", text="PASSWORD CANNOT HAVE EMPTY SPACES")

return

def getcyphertext(rand\_encrypt, inchar):

# encrypt string using Caesar Cipher algorithm

asciival = ord(inchar)

tmpasciival = asciival - rand\_encrypt

if tmpasciival < 0:

outval = (255 + tmpasciival)

else:

outval = tmpasciival

return outval

rand\_encrypt = random.randint(1,127)

# inputstring is password

inputstring = password

outputstring = ""

for i in inputstring:

cypherval = getcyphertext(rand\_encrypt, i)

outputstring = outputstring + chr(cypherval)

password=outputstring

######## GENERATION OF CUSTOMER ID #######

def gencustid():

global customer\_id

customer\_id= random.randint(100000000000,999999999999)

gencustid()

def check\_customer\_id\_repeat():

cursor14.execute("select CustID from login;")

data14= cursor14.fetchall()

for i in data14:

for j in i:

if customer\_id==j:

gencustid()

check\_customer\_id\_repeat()

else:

pass

check\_customer\_id\_repeat()

Label(signup\_screen, text="YOUR CUSTOMER ID IS: "+str(customer\_id), font=('Calibri', 20)).grid(row=9, sticky=W)

Label(signup\_screen, text="\*CUSTOMER ID IS THE KEY ATTRIBUTE YOU USE TO LOGIN AND IS USED TO LINK ALL YOUR ACCOUNTS IN MONETA BANK", font=('Calibri', 16)).grid(row=10, sticky=W)

query1="insert into login values(%s, %s, %s, %s, %s, %s);"

cursor1.execute(query1, (customer\_id, name, gender, dob, password, rand\_encrypt))

mycon.commit()

query2="insert into Type\_acc values(%s, %s, %s, %s, %s);"

cursor30.execute(query2, (customer\_id, name,'N', 'N', 'N'))

mycon.commit()

notif.config(fg="red", text="")

Label(signup\_screen, text="ACCOUNT HAS BEEN CREATED", fg="green", width = 25, font=('Calibri', 26)).grid(row=8, sticky=N, pady=12)

Button(signup\_screen, text="CLOSE", command= leave\_signup, font=('Calibri', 14), bg="#b2a6ed").grid(row=14, sticky=N, pady=12)

## signup process - receiving input ##

def signup():

login\_signup.destroy()

global temp\_name

global temp\_gender

global temp\_dob

global temp\_password

global notif

global signup\_screen

temp\_name= StringVar()

temp\_gender= StringVar()

temp\_dob= StringVar()

temp\_password= StringVar()

signup\_screen= Toplevel(master)

signup\_screen.title('Sign Up')

signup\_screen.config(bg="#bbf5ae")

#Labels

Label(signup\_screen, text="Please enter your details below to sign up", font=('Calibri', 14)).grid(row=0, column=0, sticky=N, pady=12)

Label(signup\_screen, text="Account holder Name", font=('Calibri', 14)).grid(row=1, column=0, sticky=W, pady=12)

Label(signup\_screen, text="Gender(F/M)", font=('Calibri', 14)).grid(row=2, column=0, sticky=W, pady=12)

Label(signup\_screen, text="Date Of Birth as YYYY-MM-DD", font=('Calibri', 14)).grid(row=3, column=0, sticky=W, pady=10)

Label(signup\_screen, text="Please Enter a strong password: \*Minimum 8 characters required ", font=('Calibri', 14)).grid(row=4, column=0, sticky=W, pady=12)

notif = Label(signup\_screen, font=('Calibri', 14), bg="#bbf5ae")

notif.grid(row=9, sticky=N, pady=12)

#Entries

Entry(signup\_screen, textvariable=temp\_name).grid(row=1, column=1)

Entry(signup\_screen, textvariable=temp\_gender).grid(row=2, column=1)

Entry(signup\_screen, textvariable=temp\_dob).grid(row=3, column=1)

Entry(signup\_screen, textvariable=temp\_password, show="\*").grid(row=4, column=1)

#Buttons

Button(signup\_screen, text="Sign Up", command= finish\_signup, font=('Calibri', 14), bg="#b2a6ed").grid(row=8, sticky=N, pady=12)

## Creation of FD Account ##

def createfd():

global fd\_notif

global temp\_principal\_fd

global temp\_tenure\_fd

temp\_principal\_fd=StringVar()

temp\_tenure\_fd = StringVar()

Create\_fd= Toplevel(master)

Create\_fd.title('Creating Fixed Deposit Account')

Create\_fd.config(bg="#bbf5ae")

fd\_notif=Label(Create\_fd, font=('Calibri', 14))

fd\_notif.grid(row=16, sticky=N, pady=12)

#Getting rates from text files

f1=open(r'FD.txt', 'r')

#getting heading line

firstline = f1.readline()

g= f1.read()

#Labels

Label(Create\_fd, text= "Welcome "+ login\_name+" to create your own FIXED DEPOSIT ACCOUNT!", font=('Calibri', 14)).grid(row=0, sticky=W, pady=5)

Label(Create\_fd, text= "Here are the interest rates we offer for a wide range of tenures you choose as of "+str(date.today()), font=('Calibri', 14)).grid(row=1, sticky=W, pady=5)

Label(Create\_fd, text= firstline, font=('Calibri', 14)).grid(row=2, sticky=W, pady=5)

Label(Create\_fd, text= "--------------------------------------------------------------------------------------------------", font=('Calibri', 14)).grid(row=3, sticky=W, pady=5)

Label(Create\_fd, text=g, font=('Calibri', 14)).grid(row=4, sticky=W, pady=5)

global display\_breakfd\_terms\_initial

def display\_breakfd\_text\_atcreation():

global display\_breakfd\_terms\_initial

display\_breakfd\_terms\_initial = Toplevel(master)

display\_breakfd\_terms\_initial.title('PREMATURE WITHDRAWAL/BREAKING FD - TERMS AND CONDITIONS')

display\_breakfd\_terms\_initial.config(bg="#bbf5ae")

f4 = open(r'BreakFD.txt')

g4 = f4.read()

Label(display\_breakfd\_terms\_initial, text = g4, font=('Calibri', 14)).grid(row=0, sticky=W, pady=5)

def close\_display\_breakfd\_terms\_initial():

global display\_breakfd\_terms\_initial

display\_breakfd\_terms\_initial.destroy()

Button(display\_breakfd\_terms\_initial, text="CLOSE", command = close\_display\_breakfd\_terms\_initial, font=('Calibri', 14), bg="#b2a6ed").grid(row=10, sticky=N, pady=5)

Label(Create\_fd, text = "CUSTOMERS HAVE THE CONVINIENCE OF PREAMTURE WITHDRAWAL AS WELL!!", font=('Calibri', 14)).grid(row=5, column=0, sticky=W, pady=5)

Label(Create\_fd, text = "Click here to know Terms and Conditions for Premature Withdrawal", font=('Calibri', 14)).grid(row=6, column=0, sticky=W, pady=5)

Button(Create\_fd, text="PREMATURE WITHDRAWAL/ BREAKING FD - TERMS AND CONDITIONS", command= display\_breakfd\_text\_atcreation, font=('Calibri', 14), bg="#b2a6ed").grid(row=6, column=1)

Label(Create\_fd, text="Enter the amount you would like to deposit (minimum deposit Rs.1000)", font=('Calibri', 14)).grid(row=7, column=0, sticky=W, pady=5)

Entry(Create\_fd, textvariable= temp\_principal\_fd).grid(row=7, column=1)

Label(Create\_fd, text="Enter the tenure you choose (in months)", font=('Calibri', 14)).grid(row=8, column=0, sticky=W, pady=5)

Entry(Create\_fd, textvariable= temp\_tenure\_fd).grid(row=8, column=1)

fddeptime = date.today()

DateOfOpenFD= fddeptime.strftime("%Y-%m-%d")

def genfdacc():

global FD\_Acc\_No

FD\_Acc\_No= random.randint(100000000000,999999999999)

def check\_fd\_accno\_repeat():

cursor14.execute("select FDAcc\_No from fd")

data14= cursor14.fetchall()

for i in data14:

for j in i:

if FD\_Acc\_No==j:

genfdacc()

check\_fd\_accno\_repeat()

else:

pass

genfdacc()

check\_fd\_accno\_repeat()

def openfd():

try:

FDTenure= int(temp\_tenure\_fd.get())

except:

fd\_notif.config(bg="#bbf5ae", fg="red", text="INVALID TENURE INPUT")

return

try:

FDPrincipal = int(temp\_principal\_fd.get())

except:

fd\_notif.config(bg="#bbf5ae", fg="red", text="INVALID PRINCIPAL INPUT")

return

DateOfMaturityOfFD= fddeptime + timedelta(days=int((FDTenure\*365/12)))

if FDPrincipal<1000:

fd\_notif.config(bg="#bbf5ae", fg="red", text="MINIMUM DEPOSIT Rs.1000! REQUIRED")

return

cursor15.execute("select dob from login where CustID="+str(login\_custid)+";")

data15=cursor15.fetchone()

for i in data15:

dob=i

if FDTenure<=6:

FDIntRate=2.5

elif FDTenure>6 and FDTenure<=12:

FDIntRate=4.0

elif FDTenure>12 and FDTenure<=24:

FDIntRate=5.0

elif FDTenure>24 and FDTenure<=36:

FDIntRate=5.2

elif FDTenure>36 and FDTenure<=60:

FDIntRate=5.5

elif FDTenure>60 and FDTenure<=84:

FDIntRate=5.8

elif FDTenure>84 and FDTenure<=120:

FDIntRate=6.2

else:

fd\_notif.config(bg="#bbf5ae", fg="red", text="INVALID TENURE INPUT")

return

Label(Create\_fd, text="Rate of Interest applicable: "+ str(FDIntRate) +"%", font=('Calibri', 14)).grid(row=10, column=0, sticky=W, pady=5)

query15="insert into FD values(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"

cursor15.execute(query15, (login\_custid, login\_name, dob, DateOfOpenFD, FDPrincipal, FDPrincipal, FD\_Acc\_No, FDTenure, FDIntRate, 0, FDPrincipal\*((1 + (FDIntRate/400)))\*\*(FDTenure/3), DateOfMaturityOfFD, DateOfOpenFD, 'RUNNING', -1))

mycon.commit()

fd\_notif.config(fg="green", text="ACCOUNT CREATED AND AMOUNT SUCCESSFULLY DEPOSITED")

Label(Create\_fd, text="THE ACCOUNT NO. FOR THIS ACCOUNT IS: "+str(FD\_Acc\_No), font=('Calibri', 20), bg="#C0B2FF", fg="black").grid(row=13, sticky=W, pady=5)

cursor15.execute("update type\_acc set FD='Y' where CustID="+str(login\_custid)+";")

mycon.commit()

def closeCreate\_fd():

Go\_To\_DB\_after\_fd\_creation.destroy()

account\_dashboard.destroy()

Create\_fd.destroy()

login\_session()

Go\_To\_DB\_after\_fd\_creation = Toplevel(master)

Go\_To\_DB\_after\_fd\_creation.title('Go To Dashboard')

Go\_To\_DB\_after\_fd\_creation.config(bg="#bbf5ae")

Button(Go\_To\_DB\_after\_fd\_creation, text="CLOSE AND GO TO DASHBOARD", command=closeCreate\_fd, font=('Calibri', 14), bg="#b2a6ed").grid(row=2, sticky=N)

Button(Create\_fd, text="Confirm and Open", command= openfd, font=('Calibri', 14), bg="#b2a6ed").grid(row=9, sticky=N, pady=12)

## Display of existing FD Account ##

def displayfd():

cursor16.execute("select \* from fd where CustID="+str(login\_custid)+";")

data16=cursor16.fetchone()

fd\_display\_custid = data16[0]

fd\_display\_Name = data16[1]

fd\_display\_DOB= data16[2]

fd\_display\_DOO= data16[3]

global fd\_display\_Princi

fd\_display\_Princi\_atopen=data16[4]

fd\_display\_Princi\_now= data16[5]

fd\_display\_accountNo=data16[6]

fd\_display\_tenure= data16[7]

fd\_display\_roi= data16[8]

fd\_interest\_tilldate= data16[9]

fd\_display\_maturity\_value= round(data16[10], 2)

fd\_display\_maturity\_date= data16[11]

fd\_display\_date\_of\_latest\_qr = data16[12]

fd\_display\_status = data16[13]

fd\_display\_if\_broken\_amount = data16[14]

tod= date.today()

fd\_tenure\_tilldate\_indays = (tod-fd\_display\_DOO).days

fd\_tenure\_tilldate\_inmonths = fd\_tenure\_tilldate\_indays//30

global displayfd\_screen

try:

displayfd\_screen.destroy()

except:

pass

displayfd\_screen=Toplevel(master)

displayfd\_screen.title("Your Fixed Deposit--For an assured growth")

displayfd\_screen.config(bg="#bbf5ae")

Label(displayfd\_screen, text="WELCOME "+login\_name, font=('Calibri', 14)).grid(row=0, column=0, sticky=W, pady=5)

Label(displayfd\_screen, text="ACCOUNT NUMBER: ", font=('Calibri', 14)).grid(row=10, column=0, sticky=W, pady=5)

Label(displayfd\_screen, text=str(fd\_display\_accountNo), font=('Calibri', 14)).grid(row=10, column=2, sticky=W, pady=5)

Label(displayfd\_screen, text="Your Principal deposit: ", font=('Calibri', 14)).grid(row=11, column=0, sticky=W, pady=5)

Label(displayfd\_screen, text=str(fd\_display\_Princi\_atopen), font=('Calibri', 14)).grid(row=11, column=2, sticky=W, pady=5)

Label(displayfd\_screen, text="Rate of Interest: ", font=('Calibri', 14)).grid(row=12, column=0, sticky=W, pady=5)

Label(displayfd\_screen, text=str(fd\_display\_roi)+"%", font=('Calibri', 14)).grid(row=12, column=2, sticky=W, pady=5)

Label(displayfd\_screen, text="You opened FD on: ", font=('Calibri', 14)).grid(row=13, column=0, sticky=W, pady=5)

Label(displayfd\_screen, text=str(fd\_display\_DOO), font=('Calibri', 14)).grid(row=13, column=2, sticky=W, pady=5)

Label(displayfd\_screen, text="Your FD Maturity Date: ", font=('Calibri', 14)).grid(row=14, column=0, sticky=W, pady=5)

Label(displayfd\_screen, text=str(fd\_display\_maturity\_date), font=('Calibri', 14)).grid(row=14, column=2, sticky=W, pady=5)

Label(displayfd\_screen, text="Your FD Maturity Value: ", font=('Calibri', 14)).grid(row=15, column=0, sticky=W, pady=5)

Label(displayfd\_screen, text=str(fd\_display\_maturity\_value), font=('Calibri', 14)).grid(row=15, column=2, sticky=W, pady=5)

global fd\_break\_screen

def break\_fd():

f3 = open(r'BreakFD.txt')

g2= f3.read()

fd\_break\_screen= Toplevel(master)

fd\_break\_screen.title("Breaking Fixed Deposit Account before maturity")

fd\_break\_screen.config(bg="#bbf5ae")

Label(fd\_break\_screen, text="Your Principal Deposit: "+str(fd\_display\_Princi\_atopen), font=('Calibri', 14)).grid(row=0, sticky=W, pady=5)

Label(fd\_break\_screen, text="Your Amount at the end of your original tenure would be: "+str(fd\_display\_maturity\_value), font=('Calibri', 14)).grid(row=1, sticky=W, pady=5)

Label(fd\_break\_screen, text=g2, font=('Calibri', 14)).grid(row=2, sticky=W, pady=5)

def calc\_break\_fd():

aft\_fd\_break\_screen= Toplevel(master)

aft\_fd\_break\_screen.title('Closure of Fixed Deposit before Maturity Date')

aft\_fd\_break\_screen.config(bg="#bbf5ae")

Label(aft\_fd\_break\_screen, text="Upon premature withdrawal of your Fixed Deposit Account, you can recieve an amount of: " + str(round(fd\_display\_Princi\_atopen\*((1 + ((fd\_display\_roi - 1.0)/400))\*\*((4\*fd\_tenure\_tilldate\_inmonths)/12)),2)), font=('Calibri', 14)).grid(row=2)

def del\_from\_fd\_tables():

cursor17.execute("update fd set FD\_Status='BROKEN BEFORE MATURITY', If\_broken\_amount="+str(round(fd\_display\_Princi\_atopen\*((1 + ((fd\_display\_roi - 1.0)/400))\*\*((4\*fd\_tenure\_tilldate\_inmonths)/12)),2))+" where CustID="+str(fd\_display\_custid)+";")

mycon.commit()

aft\_fd\_break\_screen.destroy()

fd\_break\_screen.destroy()

displayfd\_screen.destroy()

account\_dashboard.destroy()

login\_session()

Button(aft\_fd\_break\_screen, text="CLOSE", command= del\_from\_fd\_tables, font=('Calibri', 14), bg="#b2a6ed").grid(row=4, sticky=W, pady=5)

Button(fd\_break\_screen, text="I have read all the Terms and Conditions, agree to the same, and would like to Break the Fixed Deposit Account", command= calc\_break\_fd, font=('Calibri', 14), bg="#b2a6ed").grid(row=5, sticky=W, pady=5)

if date.today()< fd\_display\_maturity\_date:

if fd\_display\_status=='BROKEN BEFORE MATURITY':

Label(displayfd\_screen, text= "YOU HAVE BROKEN YOUR FIXED DEPOSIT AND ARE ELIGIBLE TO RECEIVE AN AMOUNT OF: "+str(fd\_display\_if\_broken\_amount), font=('Calibri', 14)).grid(row=16, column=0, sticky=W, pady=5)

else:

Button(displayfd\_screen, text="Break FD", command = break\_fd, font=('Calibri', 14), bg="#b2a6ed").grid(row=17, sticky=W, pady=5)

if date.today()>=fd\_display\_maturity\_date:

Label(displayfd\_screen, text="FIXED DEPOSIT HAS MATURED!", font=('Calibri', 14), fg='green').grid(row=17, sticky=W, pady=5)

Label(displayfd\_screen, text="KINDLY VISIT ANY LOCAL BRANCH OFFICE FOR RECEIPT OF THE AMOUNT UPON ACCOUNT CLOSURE.", font=('Calibri', 14), fg='green').grid(row=18, sticky=W, pady=5)

def go\_out\_of\_fd\_to\_dashboard():

displayfd\_screen.destroy()

account\_dashboard.destroy()

login\_session()

Button(displayfd\_screen, text="DASHBOARD", command= go\_out\_of\_fd\_to\_dashboard, font=('Calibri', 14), bg="#b2a6ed", fg="black").grid(row=0, column= 100, sticky=E)

## Creating SB Account ##

def createsb():

global temp\_principal\_sb

global sb\_notif

temp\_principal\_sb=StringVar()

Create\_sb= Toplevel(master)

Create\_sb.title('Creating Savings Bank account')

Create\_sb.config(bg="#bbf5ae")

sb\_notif=Label(Create\_sb, font=('Calibri', 14))

sb\_notif.grid(row=9, sticky=N, pady=12)

#Labels

Label(Create\_sb, text="Welcome "+login\_name+" to create your own SAVINGS BANK ACCOUNT!", font=('Calibri',14)).grid(row=0, sticky=N, pady=5)

Label(Create\_sb, text="Interest Rate offered: \n 3.3%", font=('Calibri', 14)).grid(row=1, sticky=W, pady=5)

Label(Create\_sb, text="Enter the amount you would like to deposit: (Minimum INR Rs.500)", font=('Calibri', 14)).grid(row=2, sticky=W, pady=5)

#Entries

Entry(Create\_sb, textvariable=temp\_principal\_sb).grid(row=2, column=20)

sbdeptime= date.today()

DateOfOpenSB= sbdeptime.strftime("%y-%m-%d")

#Generating Account Number

def gensbacc():

global SB\_Acc\_No

SB\_Acc\_No= random.randint(100000000000,999999999999)

def check\_sb\_accno\_repeat():

cursor8.execute("select Acc\_No from sb")

data8= cursor8.fetchall()

for i in data8:

for j in i:

if SB\_Acc\_No==j:

gensbacc()

check\_sb\_accno\_repeat()

else:

pass

gensbacc()

check\_sb\_accno\_repeat()

def opensb():

SbPrincipal=int(temp\_principal\_sb.get())

if SbPrincipal<500:

sb\_notif.config(fg="red", text="MINIMUM BALANCE Rs.500 REQUIRED!!")

return

#Amount= Principal

cursor5.execute("select dob from login where CustID="+str(login\_custid)+";")

data5=cursor5.fetchone()

for i in data5:

dob=i

query6="insert into SB values(%s, %s, %s, %s, %s, %s, %s, %s)"

cursor6.execute(query6, (login\_custid, login\_name, dob, DateOfOpenSB, SbPrincipal, SB\_Acc\_No,0, DateOfOpenSB))

mycon.commit()

sb\_notif.config(fg="green", text="ACCOUNT CREATED AND AMOUNT SUCCESSFULLY DEPOSITED")

Label(Create\_sb, text="THE ACCOUNT NO. FOR THIS ACCOUNT IS: "+str(SB\_Acc\_No), font=('Calibri', 14)).grid(row=6, sticky=W, pady=5)

cursor7.execute("update type\_acc set SB='Y' where CustID="+str(login\_custid)+";")

mycon.commit()

def closeCreate\_sb():

Go\_To\_DB\_after\_sb\_creation.destroy()

account\_dashboard.destroy()

Create\_sb.destroy()

login\_session()

Go\_To\_DB\_after\_sb\_creation = Toplevel(master)

Go\_To\_DB\_after\_sb\_creation.title('Go To Dashboard')

Go\_To\_DB\_after\_sb\_creation.config(bg="#bbf5ae")

Button(Go\_To\_DB\_after\_sb\_creation, text="CLOSE AND GO TO DASHBOARD", command=closeCreate\_sb, font=('Calibri', 14), bg="#b2a6ed").grid(row=2, sticky=N)

#Buttons

Button(Create\_sb, text="Confirm and Open", command= opensb, font=('Calibri', 14), bg="#b2a6ed").grid(row=4, sticky=W, pady=5)

## Displaying details of existing SB Account ##

def displaysb():

cursor9.execute("select \* from sb where CustID="+str(login\_custid)+";")

data9=cursor9.fetchone()

Name = data9[1]

DOB= data9[2]

DOO= data9[3]

global sb\_display\_Princi

sb\_display\_Princi=data9[4]

SBAccountNo=data9[5]

sb\_interest\_tilldate= data9[6]

tod= date.today()

def show\_prin\_aft\_wd():

wdmoney\_sb = int(sbwdmoney.get())

global sb\_display\_Princi

if sb\_display\_Princi-wdmoney\_sb<500.0:

Label(sb\_withdrawal,text= "You have to retain a minimum balance of INR Rs.500 in your account!", font=('Calibri', 14), fg="red").grid(row=3, sticky=W, pady=5)

else:

sb\_withdrawal.destroy()

successfulwithdrawal=Toplevel(master)

successfulwithdrawal.title("Withdrawal Successful")

successfulwithdrawal.config(bg="#bbf5ae")

Label(successfulwithdrawal, text="Amount Rs."+str(wdmoney\_sb)+" successfully withdrawn", font=('Calibri', 14)).grid(row=2)

Label(successfulwithdrawal, text="New Amount is: Rs."+str(sb\_display\_Princi-wdmoney\_sb), font=('Calibri', 14)).grid(row=3, sticky=W, pady=5)

sb\_display\_Princi = sb\_display\_Princi-wdmoney\_sb

cursor12.execute("update sb set Principal="+str(sb\_display\_Princi)+" where CustID="+str(login\_custid)+";")

mycon.commit()

def close\_show\_prin\_after\_wd():

successfulwithdrawal.destroy()

displaysb\_screen.destroy()

account\_dashboard.destroy()

global login\_session

login\_session()

Button(successfulwithdrawal, text="CLOSE AND GO TO DASHBOARD", command= close\_show\_prin\_after\_wd, font=('Calibri', 14), bg="#b2a6ed").grid(row=14, sticky=W, pady=5)

def sbwithdrawal():

global sb\_withdrawal

global sbwdmoney

sbwdmoney= StringVar()

sb\_withdrawal= Toplevel(master)

sb\_withdrawal.title('Withdrawal from Savings Bank Account')

sb\_withdrawal.config(bg="#bbf5ae")

Label(sb\_withdrawal, text="Enter the amount you would like to withdraw", font=('Calibri', 14)).grid(row=0, column=0, sticky=N, pady=5)

Entry(sb\_withdrawal, textvariable= sbwdmoney).grid(row=0, column=1)

Button(sb\_withdrawal, text= "Confirm and Withdraw", command= show\_prin\_aft\_wd, font=('Calibri', 14), bg="#b2a6ed").grid(row=1, sticky=W)

def show\_prin\_aft\_deposit():

depmoney\_sb = int(sbdepmoney.get())

global sb\_display\_Princi

sb\_deposit.destroy()

successfuldeposit=Toplevel(master)

successfuldeposit.title("Deposit Successful")

successfuldeposit.config(bg="#bbf5ae")

Label(successfuldeposit, text="Amount Rs."+str(depmoney\_sb)+" successfully deposited", font=('Calibri', 14)).grid(row=2)

Label(successfuldeposit, text="New Amount is: " + str(sb\_display\_Princi+depmoney\_sb), font=('Calibri', 14)).grid(row=3, sticky=W, pady=5)

sb\_display\_Princi=sb\_display\_Princi+ depmoney\_sb

cursor13.execute("update sb set Principal="+str(sb\_display\_Princi)+" where CustID="+str(login\_custid)+";")

mycon.commit()

def close\_show\_prin\_aft\_deposit():

successfuldeposit.destroy()

displaysb\_screen.destroy()

account\_dashboard.destroy()

login\_session()

Button(successfuldeposit, text="CLOSE AND GO TO DASHBOARD", command= close\_show\_prin\_aft\_deposit, font=('Calibri', 14), bg="#b2a6ed").grid(row=14, sticky=W, pady=5)

def sbdeposit():

global sb\_deposit

global sbdepmoney

sbdepmoney= StringVar()

sb\_deposit= Toplevel(master)

sb\_deposit.title('Deposit into Savings Bank Account')

sb\_deposit.config(bg="#bbf5ae")

Label(sb\_deposit, text="Enter the amount you would like to deposit", font=('Calibri', 14)).grid(row=0, column=0, sticky=W, pady=5)

Entry(sb\_deposit, textvariable= sbdepmoney).grid(row=0, column=1)

Button(sb\_deposit, text="Confirm and Deposit", command= show\_prin\_aft\_deposit, font=('Calibri', 14), bg="#b2a6ed").grid(row=1, sticky=W)

def go\_out\_of\_sb\_to\_dashboard():

displaysb\_screen.destroy()

account\_dashboard.destroy()

login\_session()

global displaysb\_screen

try:

displaysb\_screen.destroy()

except:

pass

displaysb\_screen=Toplevel(master)

displaysb\_screen.title("Your Valuable Savings")

displaysb\_screen.config(bg="#bbf5ae")

global sb\_withdrawal #sb\_withdrawal is screen name # sbwithdrawal is function name

global sb\_deposit

Label(displaysb\_screen, text="WELCOME "+login\_name, font=('Calibri',14)).grid(row=0, column=0, sticky=W, pady=5)

Label(displaysb\_screen, text="Account Number: "+str(SBAccountNo), font=('Calibri',14)).grid(row=9, sticky=W, pady=5)

Label(displaysb\_screen, text="Your Amount is: "+str(sb\_display\_Princi), font=('Calibri', 14)).grid(row=10, sticky=W, pady=5)

Button(displaysb\_screen, text="Withdrawal", command = sbwithdrawal, font=('Calibri', 14), bg="#b2a6ed").grid(row=11, sticky=W, pady=5)

Button(displaysb\_screen, text="Deposit", command=sbdeposit, font=('Calibri', 14), bg="#b2a6ed").grid(row=12, sticky=W, pady=5)

Button(displaysb\_screen, text="DASHBOARD", command = go\_out\_of\_sb\_to\_dashboard, font=('Calibri', 14), bg="#b2a6ed", fg="black").grid(row=0, column=100, sticky=E)

## Applying for new student Loan ##

def createstudloan():

Create\_studloan=Toplevel(master)

Create\_studloan.title("Apply for Student Loan")

Create\_studloan.config(bg="#bbf5ae")

Create\_studloan\_notif= Label(Create\_studloan, font=('Calibri', 14), bg="#bbf5ae")

Create\_studloan\_notif.grid(row=20, sticky=N)

f4= open(r'studloan.txt') ###### Rates of Interest for student loan are displayed from text file ######

studloanasked = StringVar()

studloantenureasked = StringVar()

g4= f4.read()

Label(Create\_studloan, text=g4, font=('Calibri', 14)).grid(row=0, column=0, sticky=W, pady=5)

Label(Create\_studloan, text= "Enter the amount you would like to apply a loan for from the given ranges", font=('Calibri', 14)).grid(row=16, column=0, sticky=W, pady=5)

Label(Create\_studloan, text= "Enter the Tenure (in years)", font=('Calibri', 14)).grid(row=17, column=0, sticky=W, pady=5)

Entry(Create\_studloan, textvariable= studloanasked).grid(row=16, column=1)

Entry(Create\_studloan, textvariable= studloantenureasked).grid(row=17, column=1)

global studloanasked\_innum

global studloantenureasked\_innum

def get\_createstudloan\_var():

global studloanasked\_innum

global studloantenureasked\_innum

try:

studloanasked\_innum=0

studloanasked\_innum = int(studloanasked.get())

if (studloanasked\_innum>10000000) or (studloanasked\_innum<50001):

Create\_studloan\_notif.config(fg="red", text="INVALID INPUT")

return

except:

Create\_studloan\_notif.config(fg="red", text="INVALID INPUT")

else:

Create\_studloan\_notif.config(fg="red", text="")

try:

studloantenureasked\_innum=0

studloantenureasked\_innum = int(studloantenureasked.get())

if (studloantenureasked\_innum) > 7 or (studloantenureasked\_innum<1):

Create\_studloan\_notif.config(fg="red", text="INVALID INPUT")

return

except:

Create\_studloan\_notif.config(fg="red", text="INVALID INPUT")

else:

Create\_studloan\_notif.config(fg="red", text="")

global studloan\_roi

def sanction\_studloan():

if studloanasked\_innum>= 50001 and studloanasked\_innum<=100000:

global studloan\_roi

studloan\_roi= 7.3

elif studloanasked\_innum>= 100001 and studloanasked\_innum<=500000:

studloan\_roi= 7.0

elif studloanasked\_innum>= 500001 and studloanasked\_innum<=1500000:

studloan\_roi= 6.7

elif studloanasked\_innum>= 1500001 and studloanasked\_innum<=4000000:

studloan\_roi= 6.5

elif studloanasked\_innum>= 4000001 and studloanasked\_innum<=10000000:

studloan\_roi= 6.1

sanction\_studloan()

def genstudloanacc():

global StudLoan\_Acc\_No

StudLoan\_Acc\_No= random.randint(100000000000,999999999999)

def check\_studloan\_accno\_repeat():

cursor19.execute("select Loan\_Acc\_No from studloan")

data19= cursor18.fetchall()

for i in data19:

for j in i:

if StudLoan\_Acc\_No==j:

genstudloanacc()

check\_studloan\_accno\_repeat()

else:

pass

genstudloanacc()

check\_studloan\_accno\_repeat()

date\_of\_loan\_sanction = date.today()

loan\_princi= studloanasked\_innum

loan\_tenure = studloantenureasked\_innum

loan\_roi = studloan\_roi

########## EMI Calculation Formula ##########

loan\_emi = (loan\_princi\*(loan\_roi\*0.01/12)\*((1 + (loan\_roi\*0.01/12))\*\*(studloantenureasked\_innum\*12)))/((((1 + (loan\_roi\*0.01/12))\*\*(studloantenureasked\_innum\*12)) - 1))

########## EMI Calculation Formula ##########

loan\_tot\_int= (loan\_emi\*12\*studloantenureasked\_innum)- studloanasked\_innum

loan\_tot\_topay = loan\_emi\*12\*studloantenureasked\_innum

loan\_status="NOT STARTED"

query20= "insert into studloan values(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s);"

cursor20.execute(query20, (login\_custid, login\_name, StudLoan\_Acc\_No, date\_of\_loan\_sanction, loan\_princi, loan\_tenure, loan\_roi, loan\_emi, loan\_tot\_topay, 0, loan\_tot\_topay, 0, 0, 0, loan\_status))

mycon.commit()

cursor21.execute("update type\_acc set studloan='Y' where CustID="+str(login\_custid)+";")

mycon.commit()

Label(Create\_studloan, text= "YOUR LOAN HAS BEEN SUNCTIONED! BEST WISHES!", font=('Calibri', 14), fg="green").grid(row=19, column=0, sticky=W, pady=5)

Label(Create\_studloan, text= "LOAN ACCOUNT NUMBER: "+ str(StudLoan\_Acc\_No), font=('Calibri', 20), bg="#C0B2FF", fg="black").grid(row=21, column=0, sticky=W, pady=5)

def close\_get\_loan():

Create\_studloan.destroy()

account\_dashboard.destroy()

login\_session()

Button(Create\_studloan, text="CLOSE AND GO TO DASHBOARD", command = close\_get\_loan, font=('Calibri', 14), bg="#b2a6ed").grid(row=23)

Button(Create\_studloan, text="SUBMIT", command = get\_createstudloan\_var, font=('Calibri', 14), bg="#b2a6ed").grid(row=21)

## Displaying details of existing Loan ##

def display\_studloan():

cursor21.execute("select Loan\_status from studloan where CustID="+str(login\_custid)+";")

data21= cursor21.fetchone()

for statii in data21:

curr\_loan\_status= statii

cursor22.execute("select \* from studloan where CustID="+str(login\_custid)+";")

data22= cursor22.fetchone()

loan\_display\_custid= data22[0]

loan\_display\_name = data22[1]

loan\_display\_accno = data22[2]

loan\_display\_dos = data22[3].strftime("%Y-%m-%d")

loan\_display\_princi = data22[4]

loan\_display\_tenure = data22[5]

loan\_display\_roi = data22[6]

loan\_display\_emi = data22[7]

#below 5 values are known if status not started

loan\_display\_totamt = data22[8]

loan\_display\_sofarpaid = data22[9]

loan\_display\_rem\_balance = data22[10]

loan\_display\_Installments\_done = data22[11]

loan\_display\_Installments\_expected = data22[12]

loan\_display\_defaults = data22[13]

loan\_display\_status = data22[14]

try:

display\_studloan\_details.destroy()

except:

pass

display\_studloan\_details= Toplevel(master)

display\_studloan\_details.title("Student Loan Details: ")

display\_studloan\_details.config(bg="#bbf5ae")

Label(display\_studloan\_details, text="WELCOME "+login\_name, font=('Calibri', 14)).grid(row=0, sticky=W, pady=5)

Label(display\_studloan\_details, text="YOUR STUDENT LOAN STATUS (REPAYMENT): ", font=('Calibri', 14)).grid(row=1, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text=loan\_display\_status, font=('Calibri', 14)).grid(row=1, column=2, sticky=W, pady=5)

Label(display\_studloan\_details, text="ACCOUNT NUMBER: ", font=('Calibri', 14)).grid(row=2, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text=str(loan\_display\_accno), font=('Calibri', 14)).grid(row=2, column=2, sticky=W, pady=5)

Label(display\_studloan\_details, text="DATE OF SANCTION OF LOAN: ", font=('Calibri', 14)).grid(row=3, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text=loan\_display\_dos, font=('Calibri', 14)).grid(row=3, column=2, sticky=W, pady=5)

Label(display\_studloan\_details, text="LOAN AMOUNT BORROWED: ", font=('Calibri', 14)).grid(row=4, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text="Rs."+str(loan\_display\_princi), font=('Calibri', 14)).grid(row=4, column=2, sticky=W, pady=5)

Label(display\_studloan\_details, text="TENURE OF REPAYMENT: ", font=('Calibri', 14)).grid(row=5, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text=str(loan\_display\_tenure)+" years ("+str(loan\_display\_tenure\*12)+" monthly installments)", font=('Calibri', 14)).grid(row=5, column=2, sticky=W, pady=5)

Label(display\_studloan\_details, text="RATE OF INTEREST: ", font=('Calibri', 14)).grid(row=6, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text=str(loan\_display\_roi)+"%", font=('Calibri', 14)).grid(row=6, column=2, sticky=W, pady=5)

Label(display\_studloan\_details, text="EMI (PER MONTH): ", font=('Calibri', 14)).grid(row=7, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text="Rs."+str(loan\_display\_emi), font=('Calibri', 14)).grid(row=7, column=2, sticky=W, pady=5)

Label(display\_studloan\_details, text="TOTAL REPAYMENT AMOUNT: ", font=('Calibri', 14)).grid(row=8, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text="Rs."+str(loan\_display\_totamt), font=('Calibri', 14)).grid(row=8, column=2, sticky=W, pady=5)

if statii=="STARTED AND RUNNING":

Label(display\_studloan\_details, text="AMOUNT PAID: ", font=('Calibri', 14)).grid(row=9, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text="Rs."+str(loan\_display\_sofarpaid), font=('Calibri', 14)).grid(row=9, column=2, sticky=W, pady=5)

Label(display\_studloan\_details, text="REMAINING BALANCE: ", font=('Calibri', 14)).grid(row=10, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text="Rs."+str(loan\_display\_rem\_balance), font=('Calibri', 14)).grid(row=10, column=2, sticky=W, pady=5)

Label(display\_studloan\_details, text="NO. OF INSTALLMENTS EXPECTED TILL DATE: ", font=('Calibri', 14)).grid(row=11, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text=str(loan\_display\_Installments\_expected), font=('Calibri', 14)).grid(row=11, column=2, sticky=W, pady=5)

Label(display\_studloan\_details, text="NO. OF INSTALLMENTS PAID TILL DATE: ", font=('Calibri', 14)).grid(row=12, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text=str(loan\_display\_Installments\_done), font=('Calibri', 14)).grid(row=12, column=2, sticky=W, pady=5)

Label(display\_studloan\_details, text="NO. OF INSTALLMENTS DEFAULTED: ", font=('Calibri', 14)).grid(row=13, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text=str(loan\_display\_defaults), font=('Calibri', 14)).grid(row=13, column=2, sticky=W, pady=5)

def payemi():

emipayment = Toplevel(master)

emipayment.title("EMI PAYMENT")

emipayment.config(bg="#bbf5ae")

Label(emipayment, text="WELCOME: "+login\_name, font=('Calibri', 14)).grid(row=0, sticky=W, pady=5)

Label(emipayment, text="PAY EMI Rs."+str(loan\_display\_emi), font=('Calibri', 14)).grid(row=1, column=0,sticky=W, pady=5)

def confirm\_emi\_payment():

cursor22.execute("update studloan set Loan\_paid= Loan\_paid+"+str(loan\_display\_emi)+" where CustID="+ str(login\_custid)+";")

mycon.commit()

cursor23.execute("update studloan set Loan\_Balance= Loan\_Balance - "+str(loan\_display\_emi)+" where CustID="+ str(login\_custid)+";")

mycon.commit()

cursor24.execute("update studloan set Loan\_InsD= Loan\_InsD + 1 where CustID="+ str(login\_custid)+";")

mycon.commit()

cursor25.execute("update studloan set Loan\_defaults = Loan\_defaults - 1 where (CustID="+ str(login\_custid)+" and Loan\_defaults>=1);")

mycon.commit()

if loan\_display\_tenure\*12==(loan\_display\_Installments\_done+1):

cursor26.execute("update studloan set Loan\_status='REPAID' where CustID="+str(login\_custid)+";")

mycon.commit()

Label(emipayment, text="AMOUNT SUCCESSFULLY PAID", font=('Calibri', 20), fg="green").grid(row=2, sticky=W, pady=5)

def close\_emi\_payment():

emipayment.destroy()

display\_studloan\_details.destroy()

display\_studloan()

Button(emipayment, text="CLOSE AND GO TO STUDENT LOAN DETAILS", command=close\_emi\_payment, font=('Calibri', 14), bg="#b2a6ed").grid(row=3, sticky=N, pady=10)

Button(emipayment, text="CONFIRM", command= confirm\_emi\_payment, font=('Calibri', 12), bg="#b2a6ed").grid(row=2, sticky=N, pady=5)

if loan\_display\_Installments\_done < (12\*loan\_display\_tenure):

Button(display\_studloan\_details, text="PAY EMI", command=payemi, font=('Calibiri', 14), bg="#b2a6ed").grid(row=15, sticky=N, pady=5)

elif statii=="REPAID":

Label(display\_studloan\_details, text="AMOUNT PAID: ", font=('Calibri', 14)).grid(row=9, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text="Rs."+str(loan\_display\_sofarpaid), font=('Calibri', 14)).grid(row=9, column=2, sticky=W, pady=5)

Label(display\_studloan\_details, text="REMAINING BALANCE: ", font=('Calibri', 14)).grid(row=10, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text="Rs."+str(loan\_display\_rem\_balance), font=('Calibri', 14)).grid(row=10, column=2, sticky=W, pady=5)

Label(display\_studloan\_details, text="LOAN HAS BEEN FULLY REPAI5D!", font=('Calibri bold', 18), fg="green").grid(row=11, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text="KINDLY VISIT ANY LOCAL BRANCH OFFICE OF MONETA BANK", font=('Calibri', 14)).grid(row=12, column=0, sticky=W, pady=5)

Label(display\_studloan\_details, text="FOR RECEIPT OF DOCUMENTS REGARDING COMPLETION OF LOAN REPAYMENT", font=('Calibri', 14)).grid(row=13, column=0, sticky=W, pady=5)

def go\_out\_of\_studloan\_to\_dashboard():

display\_studloan\_details.destroy()

account\_dashboard.destroy()

login\_session()

Button(display\_studloan\_details, text="DASHBOARD", command=go\_out\_of\_studloan\_to\_dashboard, font=('Calibri', 14), bg="#b2a6ed", fg="black").grid(row=0, column=100, sticky=E)

## Logging in after validation ##

def login\_session():

global account\_dashboard

cursor2.execute("select CustID from login;")

data= cursor2.fetchall()

List\_of\_all\_custids=[]

for i in data:

for j in i:

List\_of\_all\_custids.append(j)

global login\_custid

global login\_name

login\_custid = (temp\_login\_custid.get())

login\_password= temp\_login\_password.get()

if login\_custid=="" or login\_password=="":

login\_notif.config(fg="red", text="ALL FIELDS REQUIRED")

return

try:

login\_custid= int(temp\_login\_custid.get())

except:

login\_notif.config(bg="#bbf5ae", fg="red", text="INVALID CUSTOMERID FORMAT")

for each\_custid in List\_of\_all\_custids:

if each\_custid==login\_custid:

acc=True

break

else:

login\_notif.config(fg="red", text="ACCOUNT DOES NOT EXIST")

return

if acc==True:

cursor3.execute("select Name, Password, RandomNumber from login where CustID="+str(login\_custid)+";")

data3= cursor3.fetchone()

pw= False

for i in range(1):

########## DECRYPTION USING CAESAR CIPHER ALGORITHM ##########

def getdecyphertext(rand\_decypher, inchar):

asciival = ord(inchar)

tmpasciival = asciival + rand\_decypher

if tmpasciival > 255:

outval = (tmpasciival - 255)

else:

outval = tmpasciival

return outval

global login\_name

login\_name= data3[0]

encrypted\_password = data3[1]

rand\_decypher = data3[2]

decyphertext = ""

for j in encrypted\_password:

decypherval = getdecyphertext(rand\_decypher, j)

decyphertext = decyphertext + chr(decypherval)

if decyphertext== login\_password:

pw=True

break

else:

login\_notif.config(fg="red", text="PASSWORD INCORRECT")

return

if pw==True:

try:

login\_screen.destroy()

except:

pass

account\_dashboard = Toplevel(master)

account\_dashboard.title('Dashboard')

account\_dashboard.config(bg="#bbf5ae")

#Labels

Label(account\_dashboard, text= "CUSTOMER ID: "+str(login\_custid), font=('Calibri', 14)).grid(row=0, column=0, sticky=W)

Label(account\_dashboard, text= "Account Dashboard", font=('Calibri', 14)).grid(row=0, sticky=N, pady=10)

Label(account\_dashboard, text= "WELCOME "+login\_name.capitalize(), font=('Calibri', 14)).grid(row=1, sticky=N, pady=5)

def logout():

account\_dashboard.destroy()

loginsignup()

Button(account\_dashboard, text= "LOGOUT", command=logout, font=('Calibri', 14), bg="#b2a6ed", fg="black").grid(row=0, column=10, pady=5)

#Buttons

cursor4.execute("select FD, SB, StudLoan from Type\_acc where CustID="+str(login\_custid)+";")

data4= cursor4.fetchone()

#first checking status of sb acoount

if data4[1]=='N':

Button(account\_dashboard, text="CREATE A SAVINGS BANK ACCOUNT", command=createsb, width=100, font=('Calibri', 14), bg="#b2a6ed").grid(row=3, sticky=N, pady=5)

else:

if data4[1]=='Y':

Button(account\_dashboard, text="DISPLAY/ADD TO/WITHDRAW FROM SAVINGS", command=displaysb, width=100, font=('Calibri', 14), bg="#b2a6ed").grid(row=3, sticky=N, pady=5)

if data4[0]=='N':

Button(account\_dashboard, text="CREATE A FIXED DEPOSIT ACCOUNT", command=createfd, width=100, font=('Calibri', 14), bg="#b2a6ed").grid(row=4, sticky=N, pady=5)

if data4[0]=='Y':

Button(account\_dashboard, text="DISPLAY FIXED DEPOSIT ACCOUNT DETAILS", command=displayfd, width=100, font=('Calibri', 14), bg="#b2a6ed").grid(row=4, sticky=N, pady=5)

if data4[2]=='N':

Button(account\_dashboard, text="APPLY FOR A STUDENT LOAN", command= createstudloan, width=100, font=('Calibri', 14), bg="#b2a6ed").grid(row=5, sticky=N, pady=5)

if data4[2]=='Y':

Button(account\_dashboard, text="VIEW STUDENT LOAN DETAILS/ PAY EMI", command= display\_studloan, width=100, font=('Calibri', 14), bg="#b2a6ed").grid(row=5, sticky=N, pady=5)

## Login Screen ##

def login():

login\_signup.destroy()

#Variables

global temp\_login\_custid

global temp\_login\_password

global login\_notif

global login\_screen

global login\_custid

temp\_login\_custid = StringVar()

temp\_login\_password = StringVar()

#Login Screen

login\_screen = Toplevel(master)

login\_screen.title('Login')

login\_screen.config(bg="#bbf5ae")

#Labels

Label(login\_screen, text="Login to your account", font=('Calibri', 14)).grid(row=0, sticky=W)

Label(login\_screen, text="CustomerID", font=('Calibri', 14)).grid(row=1, sticky=W)

Label(login\_screen, text="Password", font=('Calibri', 14)).grid(row=2, sticky=W)

login\_notif = Label(login\_screen, font=('Calibri', 14), bg="#bbf5ae")

login\_notif.grid(row=4, sticky=N)

#Entry

Entry(login\_screen, textvariable= temp\_login\_custid).grid(row=1, column=1, padx=5)

Entry(login\_screen, textvariable= temp\_login\_password, show="\*").grid(row=2, column=1, padx=5)

#Buttons

global login\_session

Button(login\_screen, text = "Login", command= login\_session, width=17, font=('Calibri', 14), bg="#b2a6ed").grid(row=3, sticky=W, pady=5)

img= Image.open('onlb.png')

img= img.resize((500,350))

img= ImageTk.PhotoImage(img)

img2= Image.open('moneta\_logo.png')

img2= img2.resize((100,67))

img2= ImageTk.PhotoImage(img2)

Label(master, image=img2).grid(row=0, column=0, sticky=N)

Label(master, text = "MONETA BANK", font=('Calibri',30), bg="#bbf5ae").grid(row=3, sticky=N)

Label(master, text= "Where your money is safe and growing", font=('Calibri',16, 'italic'), bg="#bbf5ae").grid(row=4, sticky=N)

Label(master, image=img).grid(row=5, sticky=N, pady=15)

## Main Screen for Login/SignUP ##

def loginsignup():

global login\_signup

login\_signup = Toplevel(master)

login\_signup.title('SignUp-Login Page')

login\_signup.config(bg="#bbf5ae")

Label(login\_signup, text = "MONETA BANK", font=('Calibri',30), bg="#bbf5ae").grid(row=0, sticky=N, pady=10)

Label(login\_signup, text= "Where your money is safe and growing", font=('Calibri',16, 'italic'), bg="#bbf5ae").grid(row=1, sticky=N)

Label(login\_signup, image=img).grid(row=2, sticky=N, pady=15)

Button(login\_signup, text= "Sign Up", font=('Calibri',14), width=20, command=signup, bg="#b2a6ed").grid(row=3, sticky=N)

Button(login\_signup, text="Login", font=('Calibri', 14), width=20, command=login, bg="#b2a6ed").grid(row=4, sticky=N, pady=12)

Button(master, text="Click to Login/SignUp Page", font=('Calibri', 14), bg="#b2a6ed", width=20, command=loginsignup).grid(row=6, sticky=N)

master.mainloop()

Other Python Programs containing code for calculations to be done on daily and quarterly basis:

*Savings Bank updations (For daily Interest Calculation)*

**sb-daily-run.py**

import mysql.connector as sqltor

mycon = sqltor.connect(host='localhost', user='root', password='mysql', database='banking')

cursor1= mycon.cursor()

cursor1.execute("update sb set Interest= Interest + (Principal\*3.3)/(365\*100);")

mycon.commit()

mycon.close()

*Savings bank updations (To be run daily, but changes made are quarterly with respect to opening of every customer’s account)*

**sb-quarterly-run.py**

import mysql.connector as sqltor

mycon= sqltor.connect(host='localhost', user='root', password='mysql', database='banking')

cursor1= mycon.cursor()

cursor2= mycon.cursor()

cursor3= mycon.cursor()

cursor1.execute("update sb set Principal = Principal + Interest where datediff(curdate(),DateOfLatestQR)=91;")

mycon.commit()

cursor2.execute("update sb set Interest=0 where datediff(curdate(),DateOfLatestQR)=91;")

mycon.commit()

cursor3.execute("update sb set DateOfLatestQR = curdate() where datediff(curdate(), DateOfLatestQR)=91;")

mycon.commit()

mycon.close()

*Fixed Deposit Updations (For Daily Interest Calculation)*

**fd-daily-run.py**

import mysql.connector as sqltor

mycon= sqltor.connect(host='localhost', user='root', password='mysql', database='banking')

cursor1= mycon.cursor()

cursor2=mycon.cursor()

cursor1.execute("select \* from fd where curdate()<=MaturityDate and FD\_Status='RUNNING';")

data1= cursor1.fetchall()

for i in data1:

if i[7]<=6:

cursor2.execute("update fd set FDInterest= FDInterest + ((Principal\*3.8)/(100\*365)) where CustID="+ str(i[0])+";")

elif i[7]>6 and i[7]<=12:

cursor2.execute("update fd set FDInterest= FDInterest + ((Principal\*4.0)/(100\*365)) where CustID="+ str(i[0])+";")

elif i[7]>12 and i[7]<=24:

cursor2.execute("update fd set FDInterest= FDInterest + ((Principal\*5.0)/(100\*365)) where CustID="+ str(i[0])+";")

elif i[7]>24 and i[7]<=36:

cursor2.execute("update fd set FDInterest= FDInterest + ((Principal\*5.2)/(100\*365)) where CustID="+ str(i[0])+";")

elif i[7]>36 and i[7]<=60:

cursor2.execute("update fd set FDInterest= FDInterest + ((Principal\*5.5)/(100\*365)) where CustID="+ str(i[0])+";")

elif i[7]>60 and i[7]<=84:

cursor2.execute("update fd set FDInterest= FDInterest + ((Principal\*5.8)/(100\*365)) where CustID="+ str(i[0])+";")

elif i[7]>84 and i[7]<=120:

cursor2.execute("update fd set FDInterest= FDInterest + ((Principal\*6.2)/(100\*365)) where CustID="+ str(i[0])+";")

else:

pass

cursor3.execute("update fd set FD\_Status= 'COMPLETED' where curdate()<=MaturityDate and FD\_Status!='BROKEN BEFORE MATURITY';"

mycon.commit()

mycon.close()

*Fixed Deposit Updations (For Quarterly Addition of Interest to Principal)*

**fd-quarterly-run.py**

import mysql.connector as sqltor

from datetime import \*

mycon= sqltor.connect(host='localhost', user='root', password='mysql', database='banking')

cursor1= mycon.cursor()

cursor2= mycon.cursor()

cursor3= mycon.cursor()

curdate = date.today()

curdate = curdate.strftime("%Y-%m-%d")

query1= "update fd set Principal=Principal+FDInterest where (datediff(%s,DateOfLatestQR)=91 and %s<=MaturityDate and FD\_Status='RUNNING');"

cursor1.execute(query1, (curdate, curdate))

mycon.commit()

query2= "update fd set FDInterest=0 where (datediff(%s, DateOfLatestQR)=91 and %s<=MaturityDate and FD\_Status='RUNNING');"

cursor2.execute(query2, (curdate, curdate))

mycon.commit()

query3= "update fd set DateOfLatestQR=curdate() where (datediff(%s, DateOfLatestQR)=91 and %s<=MaturityDate and FD\_Status='RUNNING');"

cursor3.execute(query3, (curdate, curdate))

mycon.commit()

mycon.close()

*Student Loan updations (Run on a daily basis to update number of installments expected till current date as well as checking completion of Loans where all installments have been paid)*

**studloan-daily-run.py**

from datetime import \*

import mysql.connector as sqltor

mycon= sqltor.connect(host='localhost', user='root', password='mysql', database='banking')

cursor1= mycon.cursor()

cursor2=mycon.cursor()

cursor3=mycon.cursor()

cursor4=mycon.cursor()

cursor5=mycon.cursor()

cursor6=mycon.cursor()

cursor7=mycon.cursor()

cursor8=mycon.cursor()

cursor9=mycon.cursor()

cursor1.execute("select \* from studloan;")

data1= cursor1.fetchall()

#print(data1)

for i in data1:

CustID= i[0]

Name=i[1]

acc\_no = i[2]

dos= i[3]

princi = i[4]

tenure = i[5]

roi=i[6]

each\_emi = i[7]

tot\_topay = i[8]

paid= i[9]

balance\_topay = i[10]

ins\_done = i[11]

ins\_expected = i[12]

defaults = i[13]

status = i[14]

tod= date.today()

ndays = (tod-dos).days

dor\_starts = dos + timedelta(days=365)

if ndays>=365 and status=="NOT STARTED":

cursor2.execute("update studloan set Loan\_status='STARTED AND RUNNING' where Name='"+Name+"';")

mycon.commit()

if (tod-dor\_starts).days%30==0 and (tod-dor\_starts).days!=0:

cursor3.execute("update studloan set Loan\_InsEx = Loan\_InsEx + 1 where Name='"+Name+"';")

mycon.commit()

cursor4.execute("update studloan set Loan\_defaults = Loan\_defaults + 1 where Name='"+Name+"' and Loan\_InsD<=Loan\_InsEx;")

mycon.commit()

if status=="STARTED AND RUNNING":

if (tod-dor\_starts).days%30==0 and (tod-dor\_starts).days!=0:

cursor5.execute("update studloan set Loan\_InsEx = Loan\_InsEx + 1 where Name='"+Name+"';")

mycon.commit()

cursor6.execute("update studloan set Loan\_defaults = Loan\_defaults + 1 where Name='"+Name+"' and Loan\_InsD<=Loan\_InsEx;")

mycon.commit()

if ins\_done==tenure\*12 and paid==tot\_topay:

cursor7.execute("update studloan set Loan\_status = 'REPAYED' where Name='"+Name+"';")

mycon.commit()

mycon.close()

# 

# **Screenshots in Python Platform**

On running the python program, this is the first screen that appears:

This is the master screen. All other windows are secondary screens which are in a loop with the master screen.



Clicking “Click to Login/SignUp”:

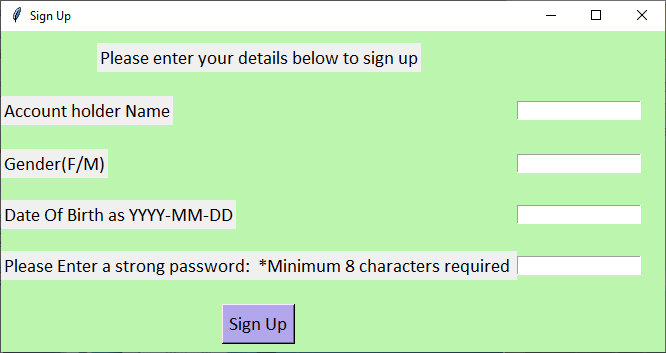
The window below opens:



Clicking “Sign Up” now, to create an online account with Moneta Bank.

The above page closes.

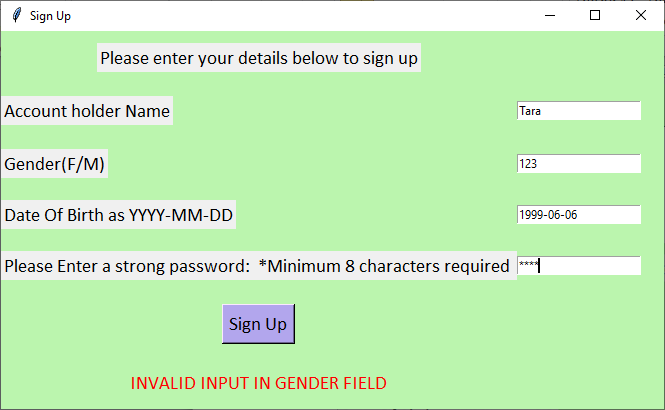
A new window will open as follows.



Entering required details:

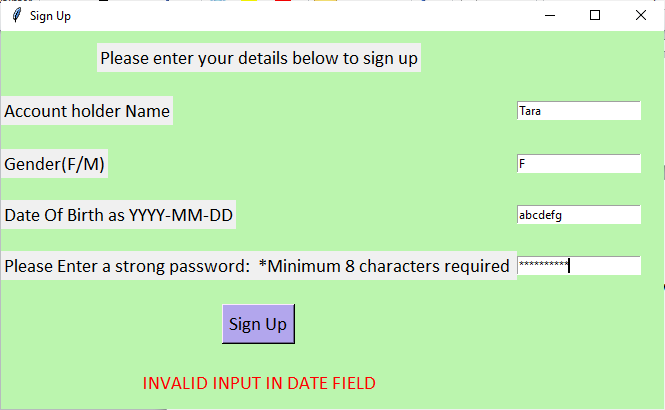
(#Trying invalid inputs for verifying validation process)

* Giving invalid datatype in gender field:



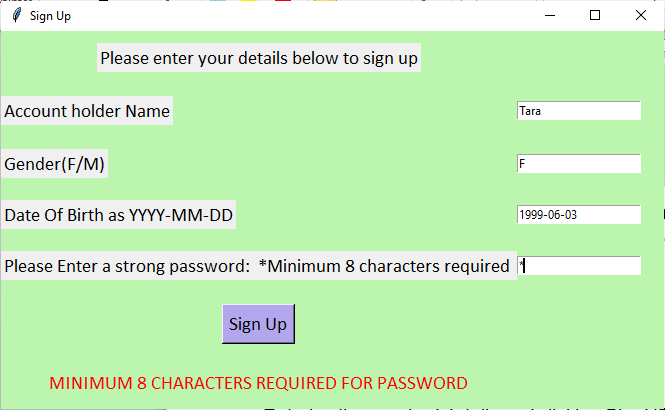
Error message is raised at the bottom of the window.

* Giving valid input for gender field, but invalid input for date field:



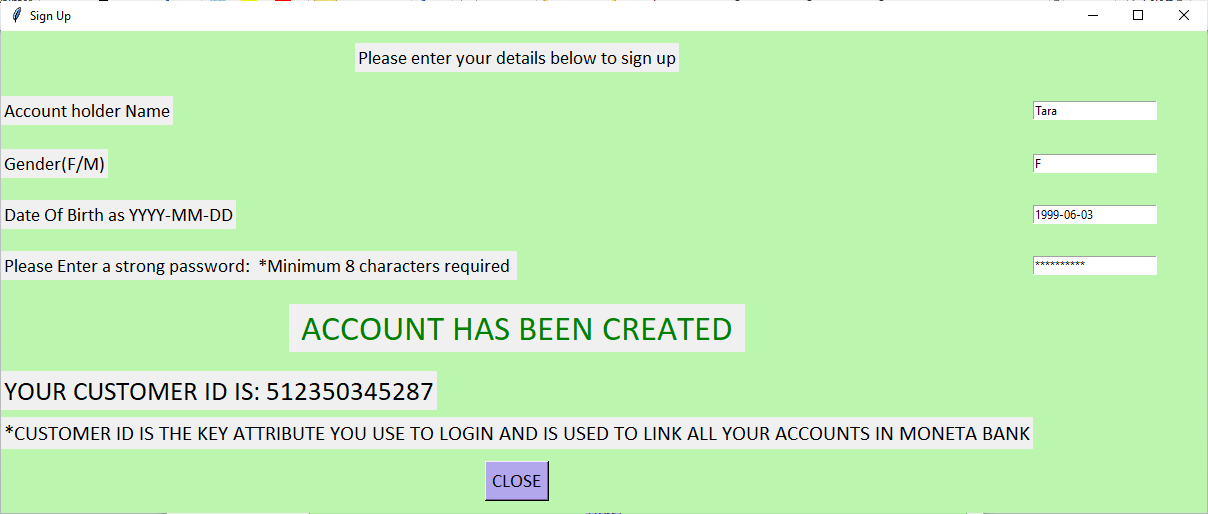
Error message is raised at the bottom of the window and entry has to be rectified for proceeding.

* Giving valid gender and date entries, but password with length less than 8:



Error message is raised and entry must be rectified.

Now, giving valid input in all fields and clicking “SignUp”:

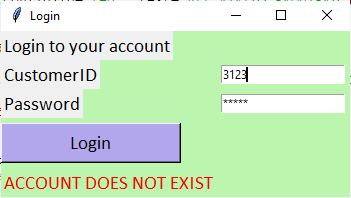


Clicking “CLOSE” goes back to 1st main Screen (1st Screenshot).

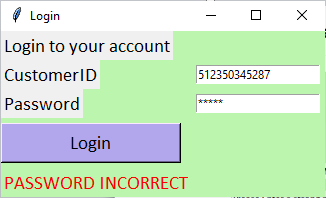
Clicking “Click to Login/SignUp page”🡪 goes to 2nd Screen.

In 2nd Screen- clicking “Login”

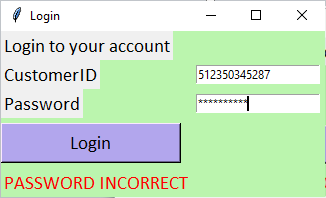
When an invalid Customer ID is given: Error message is raised



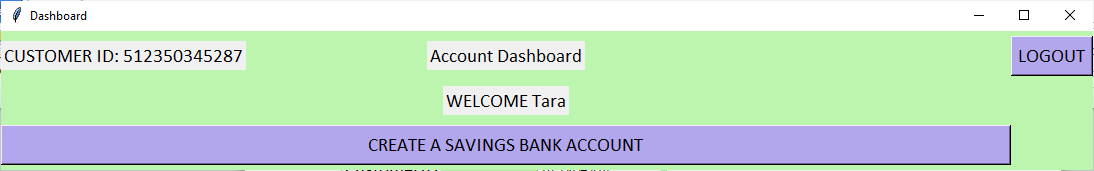
When incorrect password for existing Customer ID is given: Error is raised.



Giving correct details now : (\* “PASSWORD INCORRECT” message was for previous entry. Below Screenshot is before clicking Login)



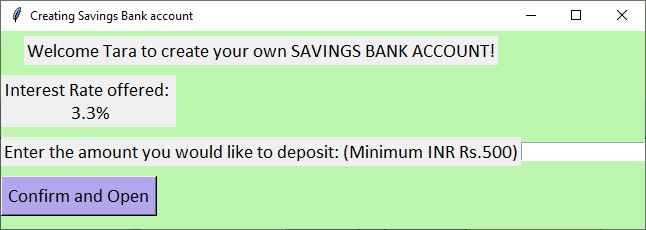
Clicking “Login”:



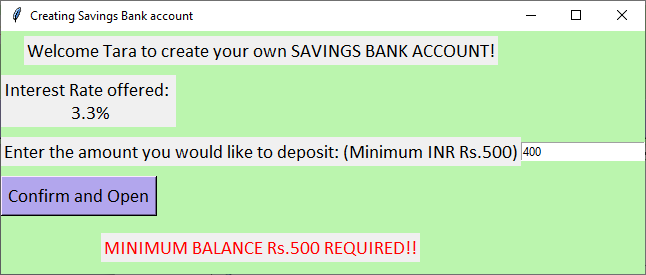
Initially, the only feature available would be Savings Bank Account. Once, this is created, options to avail other features would be displayed.

Click “CREATE A SAVINGS BANK ACCOUNT”

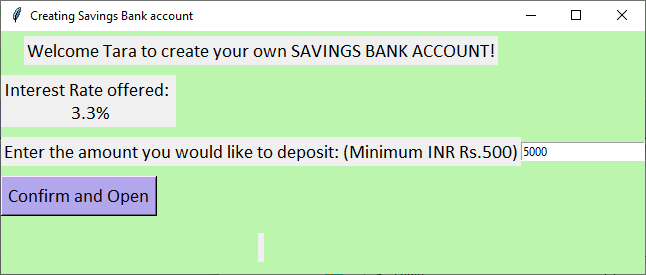
**CREATING SAVINGS BANK ACCOUNT**



On entering an amount less than Rs.500, a message is displayed in the bottom of the window saying that minimum balance is required.

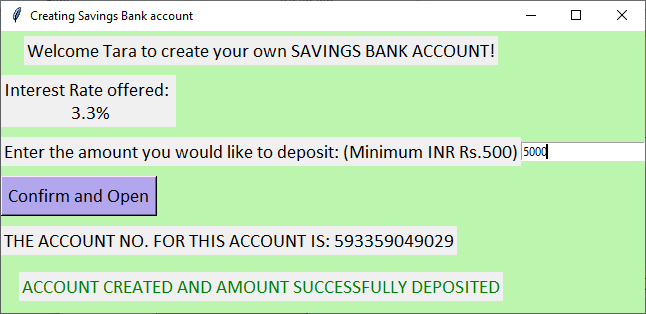


On typing a new valid amount (Rs.5000/-)

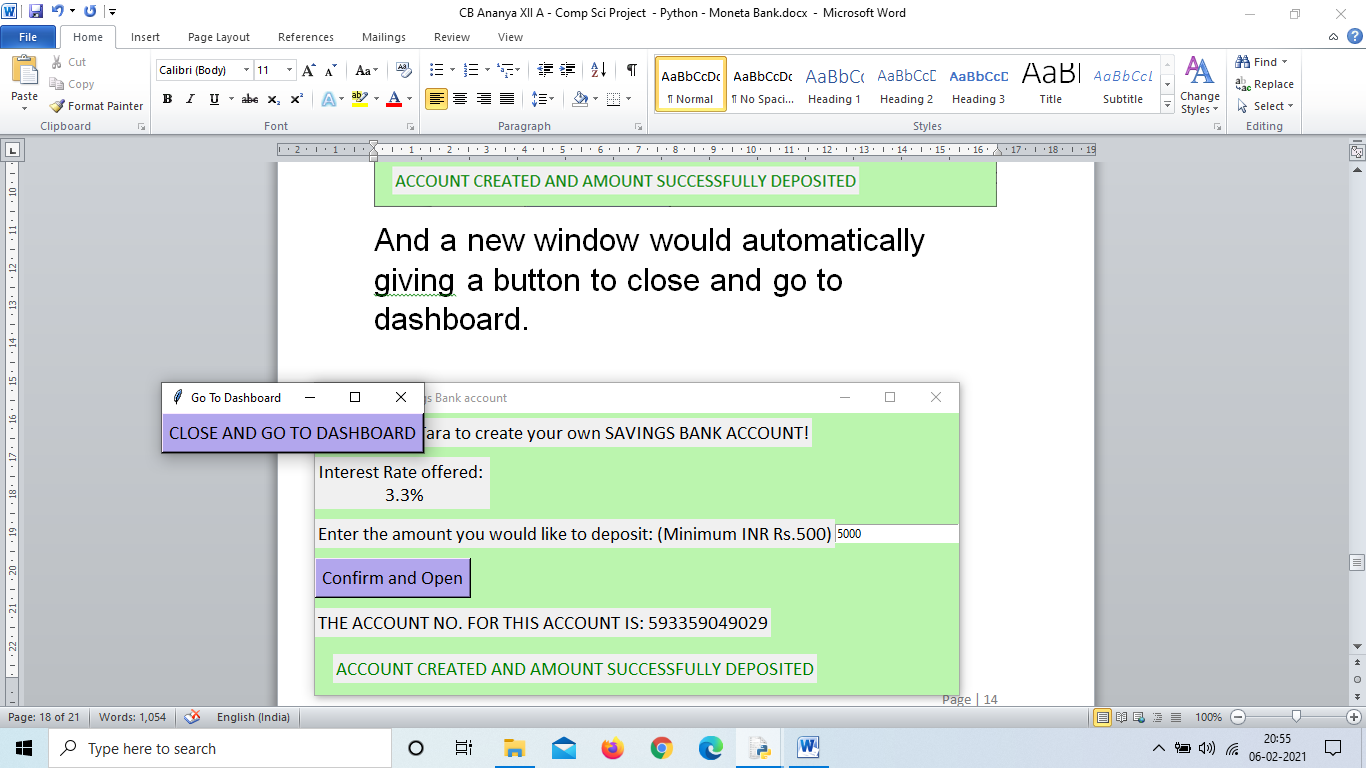


Clicking “Confirm and Open”:

“Create Savings Bank account” window would show the following details:

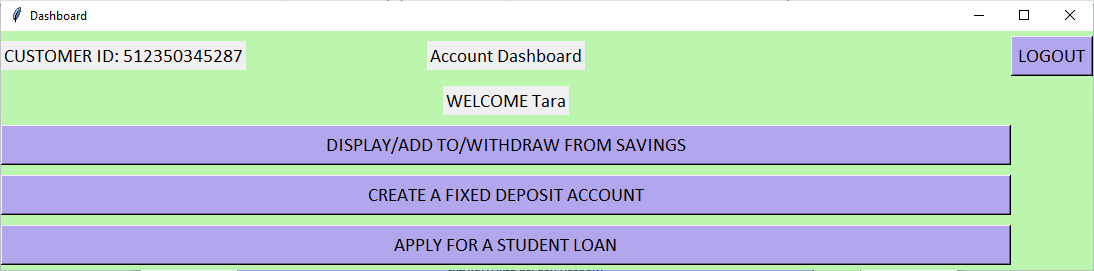


A new window would pop-up immediately giving a button to close and go to dashboard.



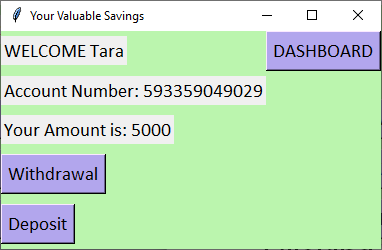
Clicking the button would close the “GO TO DASHBOARD” and “CREATING SAVINGS BANK ACCOUNT” windows and lead back to the dashboard.

The refreshed dashboard screen would appear with new features(Fixed deposit and Student Loan) enabled.



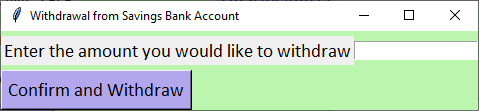
Clicking Display/Add to/Withdraw From Savings:

Name, Account Number and current balance are displayed



**Withdrawal:**

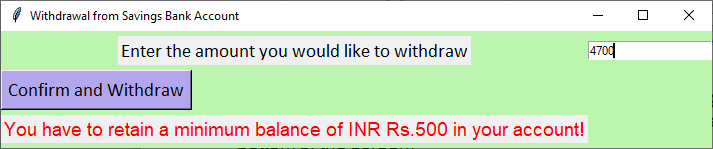
On clicking “Withdrawal”, the screen below appears. Enter the amount you would like to withdraw.



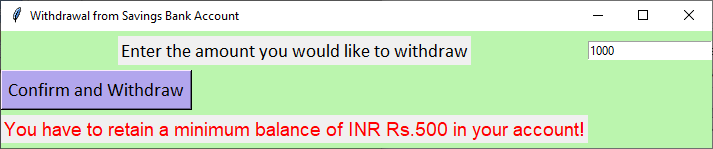
NOTE: Withdrawal amount should be such that remaining balance would be Rs.500/- or above. Otherwise, a message will be displayed at the bottom of the screen.

(In this case, since amount was Rs.5000/-, any withdrawal amount greater than Rs.4500/- would be an invalid input)

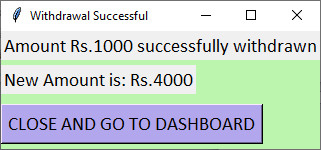
Trying withdrawal of Rs.4700 gives the following message:



Giving a valid input (<=Rs.4500/-):

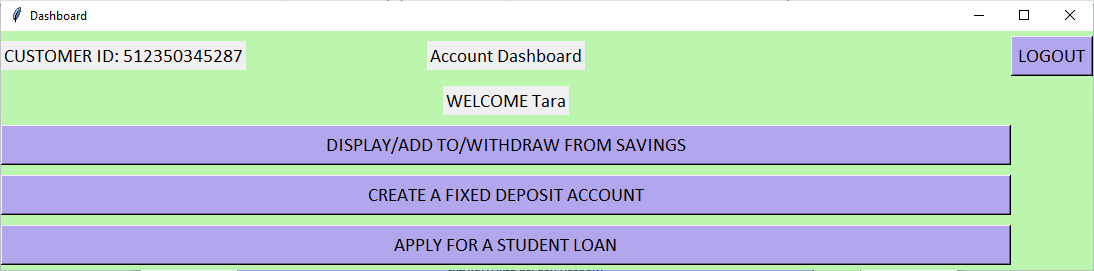


Clicking “Confirm and Withdraw”, the above window closes and a new window with withdrawal details will appear:

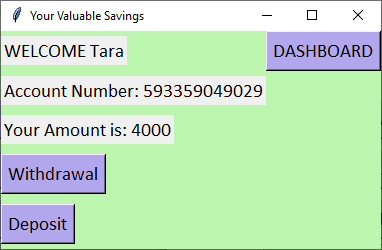


Click “CLOSE AND GO TO DASHBOARD”:

The Dashboard appears again:



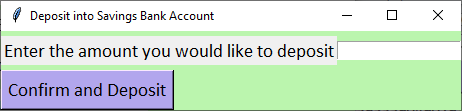
Again going for “DISPLAY/ADD TO/WITHDRAW FROM SAVINGS”:



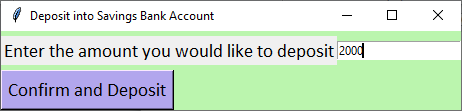
(The amount has been updated)

**Deposit:**

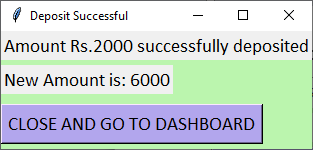
Clicking “Deposit” to add to your savings, the following new window opens:



Entering the deposit amount: (Addition to account does not have any minimum amount requirement)

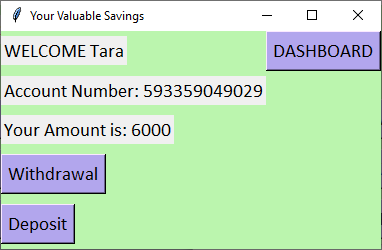


Clicking “Confirm and Deposit”, the above window closes and a new window (below) displaying details of the deposit will appear:



Clicking “CLOSE AND GO TO DASHBOARD” goes back to dashboard.

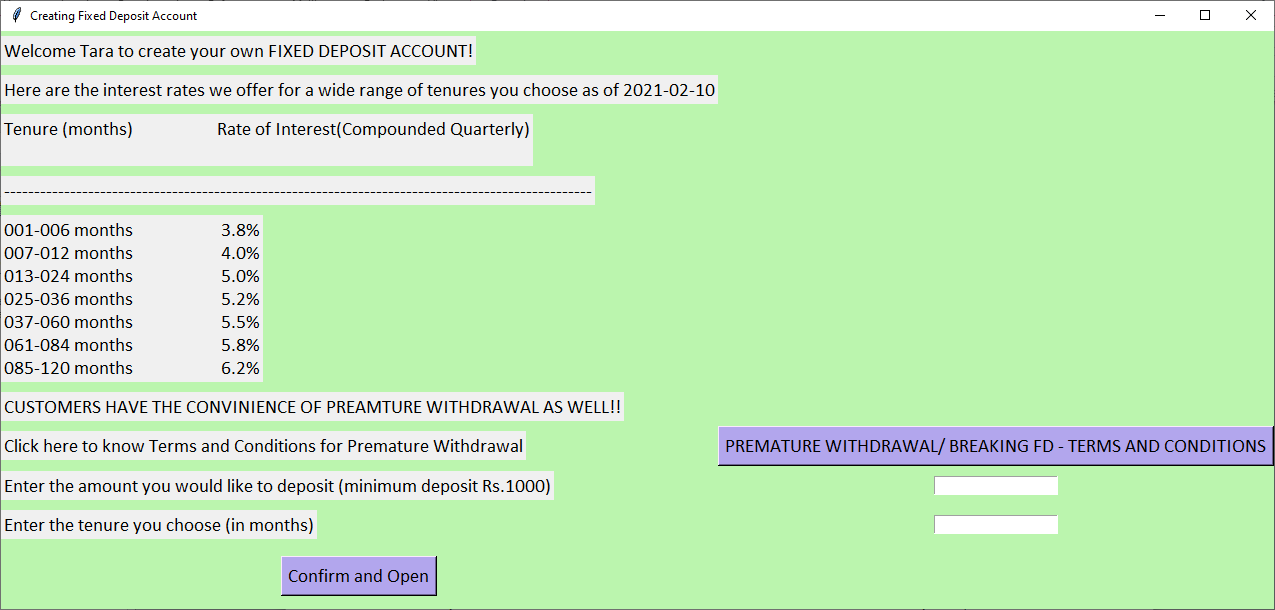
Again, going to “DISPLAY/ADD TO/WITHDRAW FROM SAVINGS”, the updated amount is displayed:



Clicking DASHBOARD and going back to DASHBOARD:

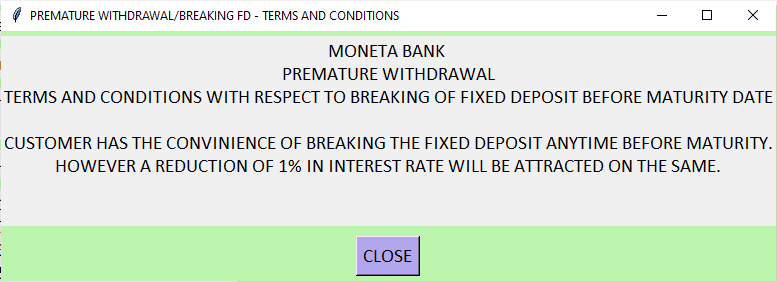
**FIXED DEPOSIT ACCOUNT:**

In the dashboard, clicking “CREATE A FIXED DEPOSIT ACCOUNT”:



All interest rates and tenures are displayed and a separate button for viewing “Premature Withdrawal -Terms and Conditions” are given, to keep the customer informed regarding their choices.

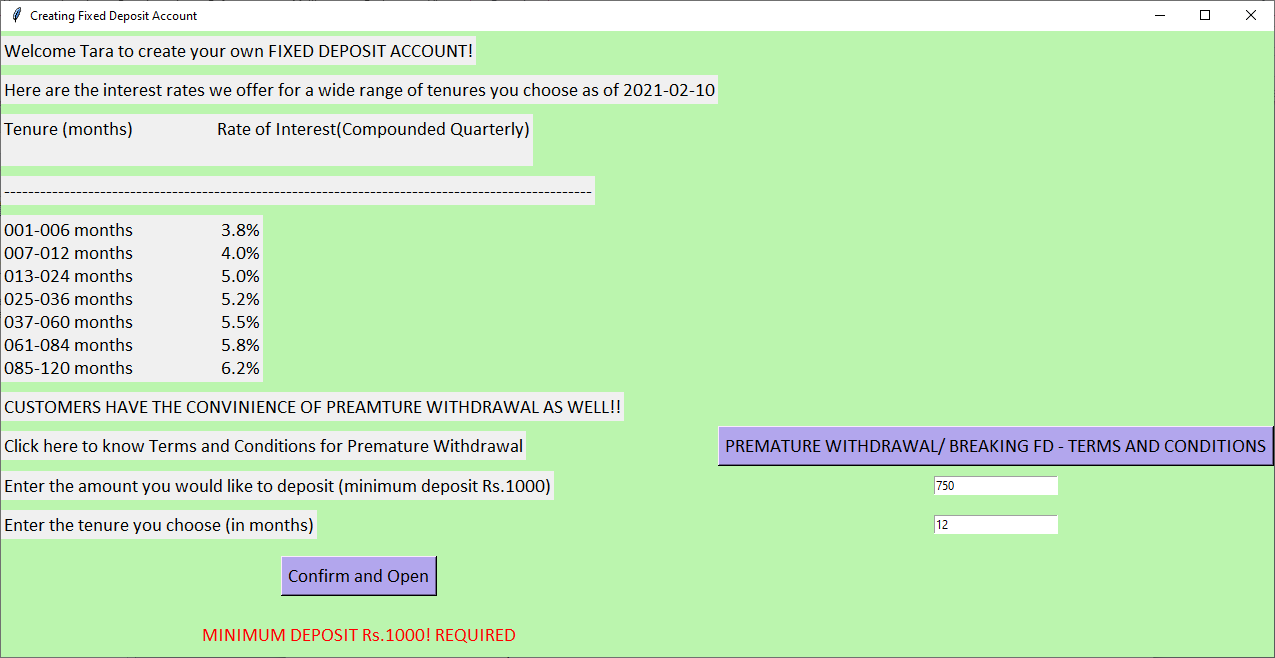
Clicking “PREMATURE WITHDRAWAL/ BREAKING FD- TERMS AND CONDITIONS”, the window below opens.



Click ‘CLOSE’. The window closes and goes back to FD creation window.

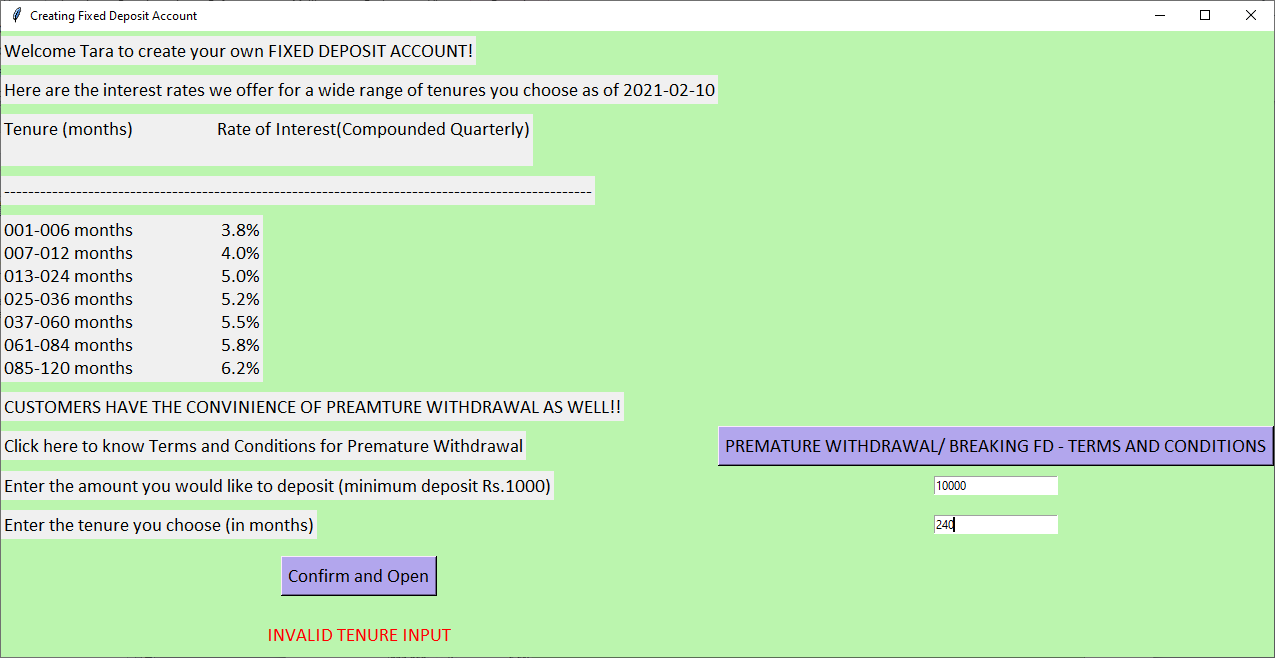
Entering Data:

* Giving an amount less than Rs.1000/-: (Entering Rs.750)

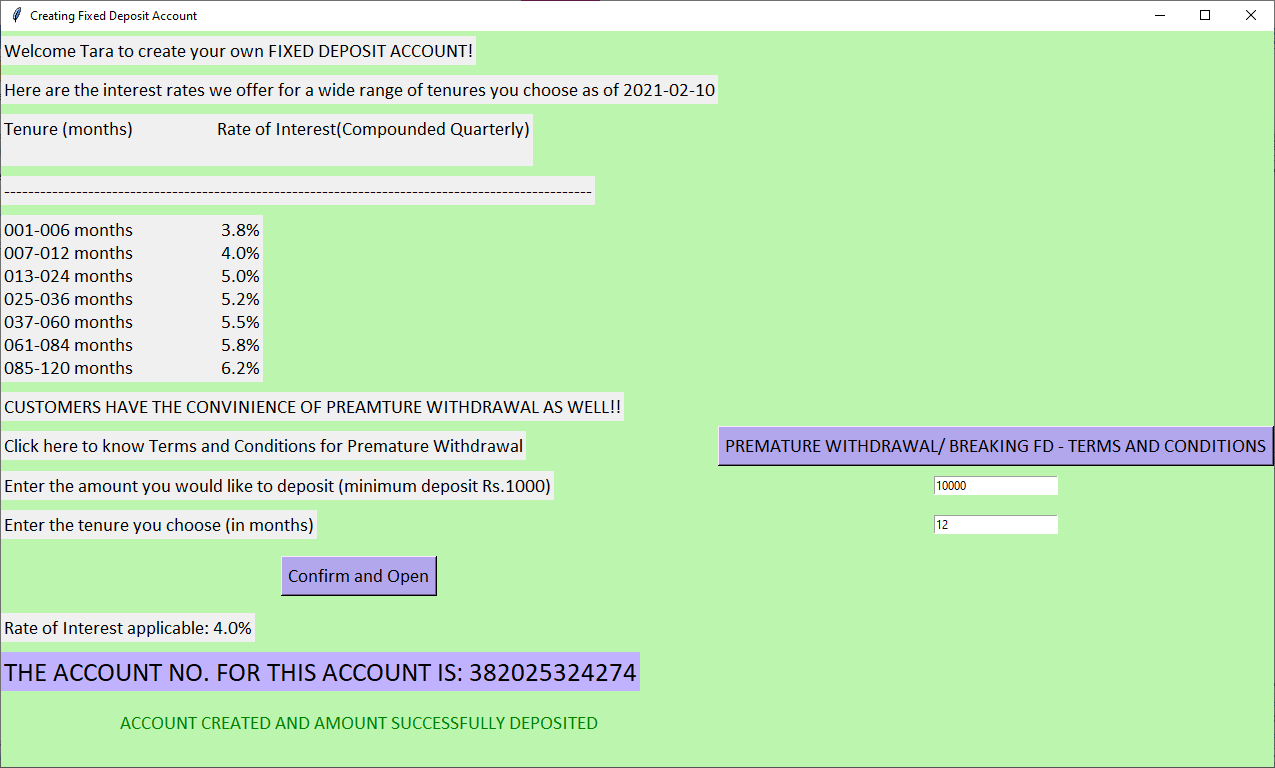


Error message is raised saying minimum deposit Rs.1000/- required.

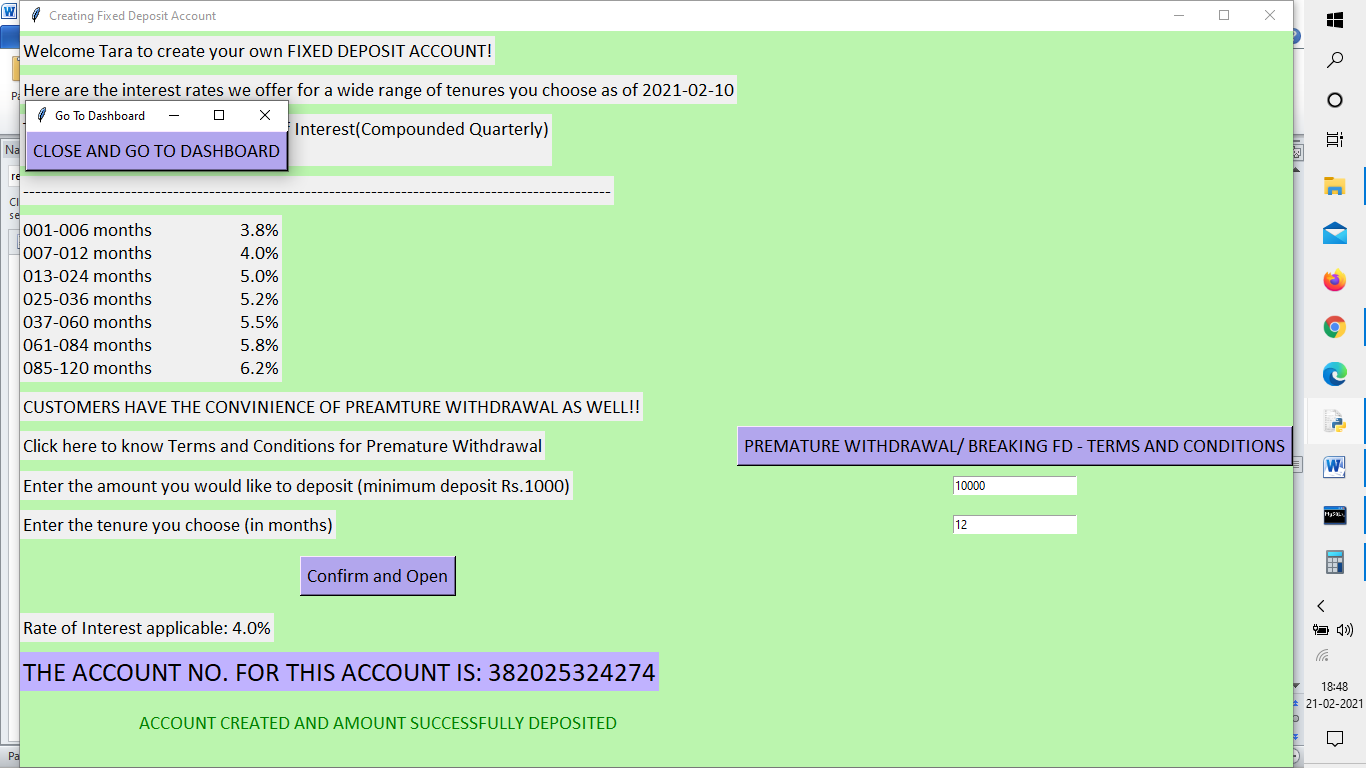
* Giving valid deposit amount, but invalid tenure: (Rs.10,000 for 240 months)



Now, giving valid input in both fields and clicking “CONFIRM AND OPEN”:

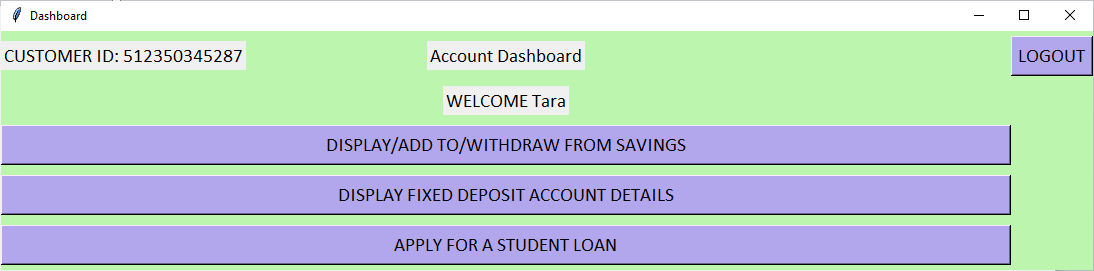


Another window also appears in the front giving option to close and go to dashboard:

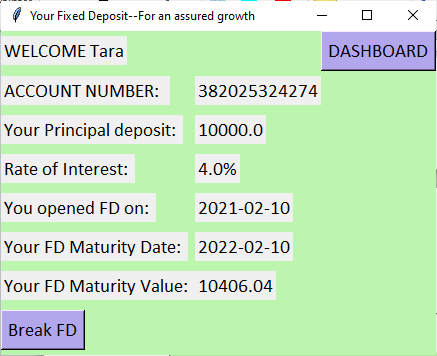


Clicking “CLOSE AND GO TO DASHBOARD”

Updated Dashboard appears:



Clicking “DISPLAY FIXED DEPOSIT ACCOUNT DETAILS:



**PREMATURE WITHDRAWAL**

Considering the case when customer Tara wants to break the Fixed Deposit, 6 months after creating the account.

That is, her FD tenure was originally 12 months, but she wishes to withdraw the deposit in 6 months.

In that case, she would proceed as below.

***Assumption:***

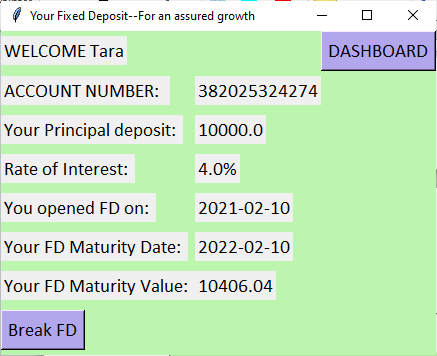
Today is 2021-08-10 (12th of August, 2021)

Tara opened her FD on 2021-02-10 (10th February, 2021)

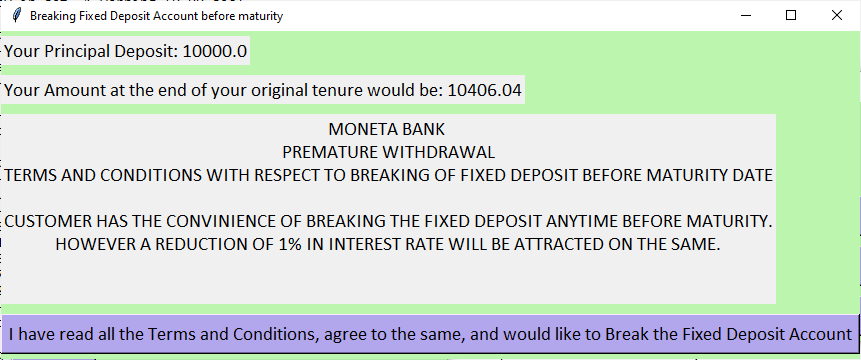
Her FD matures on 2022-02-10 (10th February, 2022)

Tara wishes to withdraw her deposit today.

Tara’s FD details as of 2021-08-10:

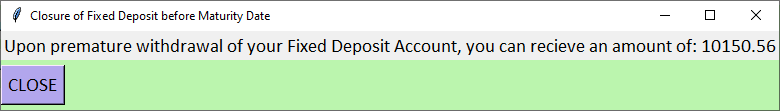


If she clicks “Break FD”: The following screen opens

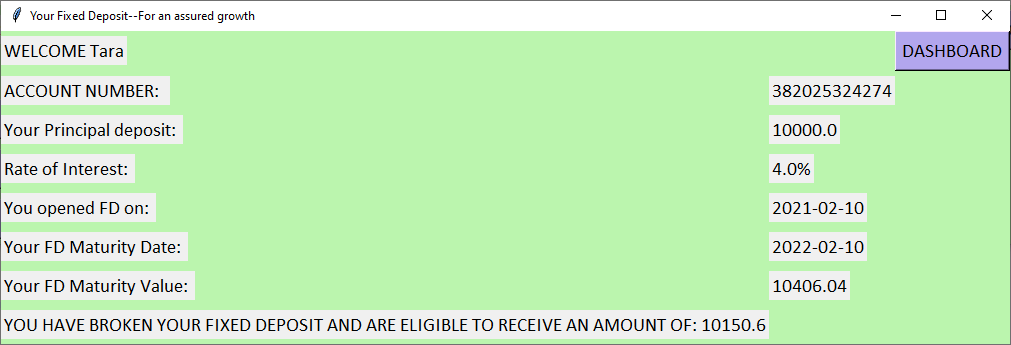


Customer Tara agrees to the terms and Conditions and clicks the button for Breaking FD.

The following Screen Appears:



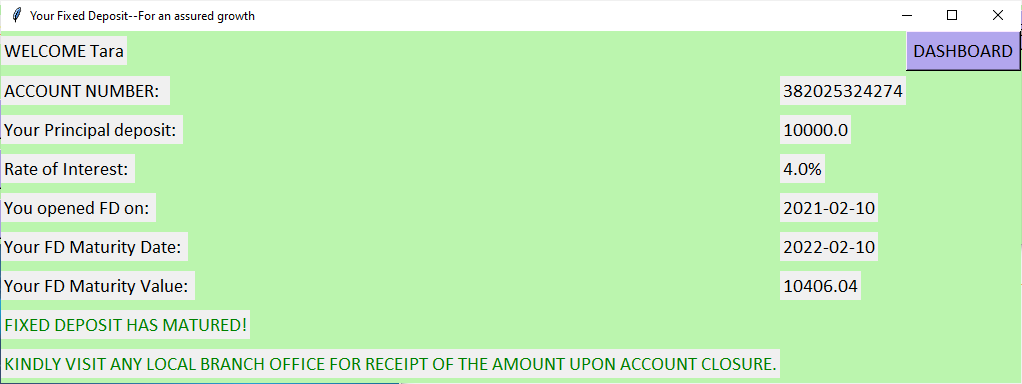
When she clicks “CLOSE”, it will lead her back to the “DASHBOARD” where she clicks “DISPLAY FIXED DEPOSIT ACCOUNT DETAILS”. The window appears as below:



Customer Tara is eligible to receive an amount of Rs.10,150.60/- upon premature withdrawal.

**WITHDRAWAL ON MATURITY: (IDEAL CASE)**

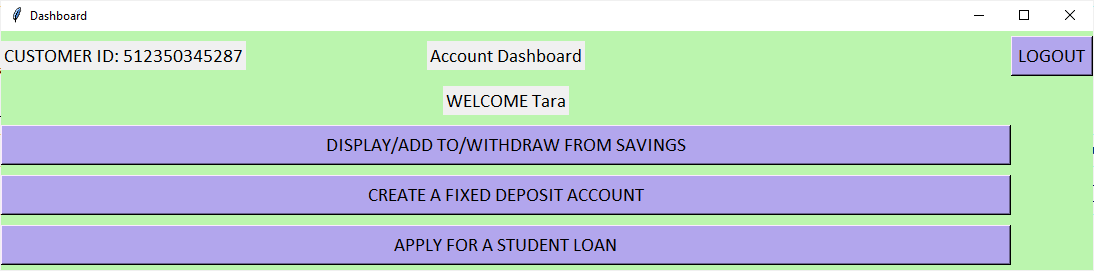
If Tara had opted to retain the FD till maturity, on (and after) 2022-02-10 (10th February 2022), her FD details would be as below.



Customer Tara is eligible to receive the complete maturity value of her fixed deposit.

Once the amount is received by Tara and an official statement of receipt has been given by her to the bank, her account details would be deleted from our bank’s running FIXED DEPOSIT table.

On completing all of these processes, this would be Tara’s dashboard:

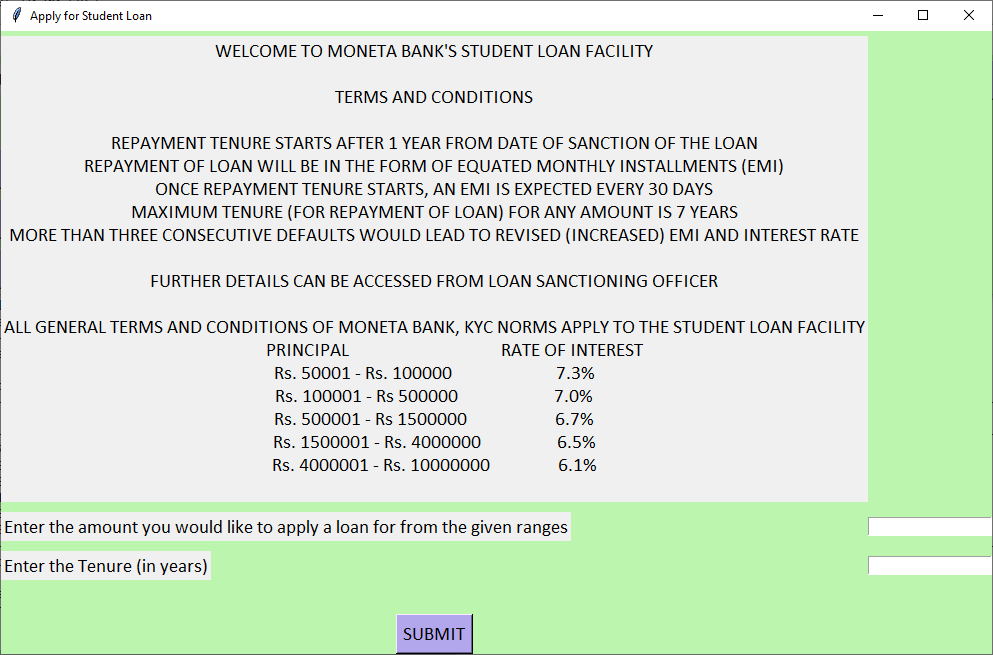


**STUDENT LOAN**

Assuming Tara is a student in the beginning of her 3rd year of college doing a 4 year course and requires a student loan for her final year expenditures.

She clicks “APPLY FOR A STUDENT LOAN”

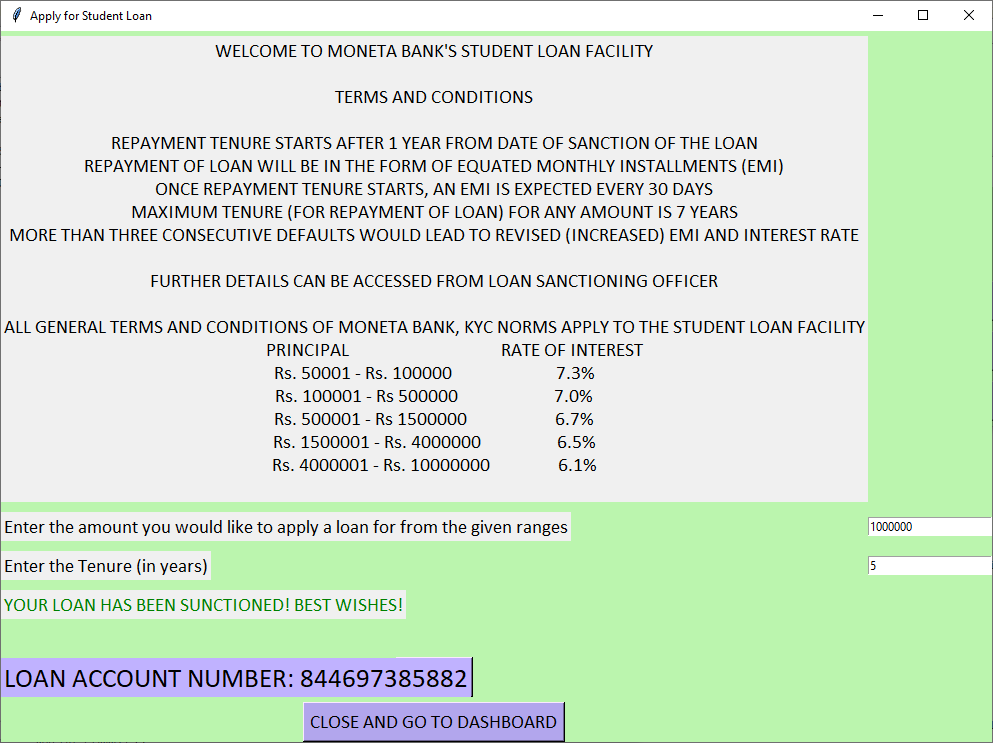
The window below opens



Suppose Tara applies for a loan with an amount of Rs.10,00,000/- and a tenure of 5 years.

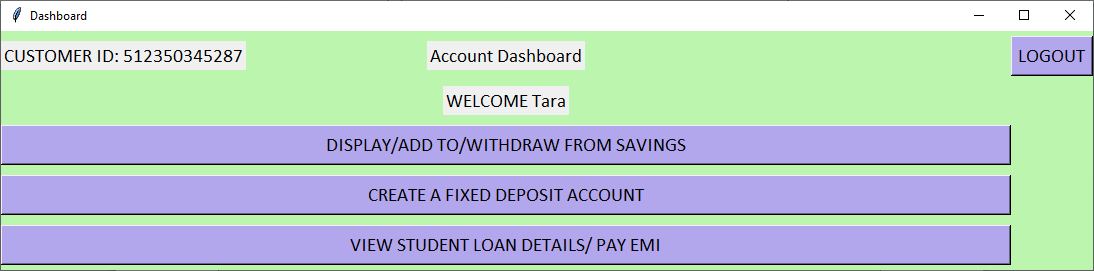
She **reads the Terms and Conditions carefully**, enters her specifications and clicks “SUBMIT”.

A message stating that her loan has been sanctioned would appear as below.



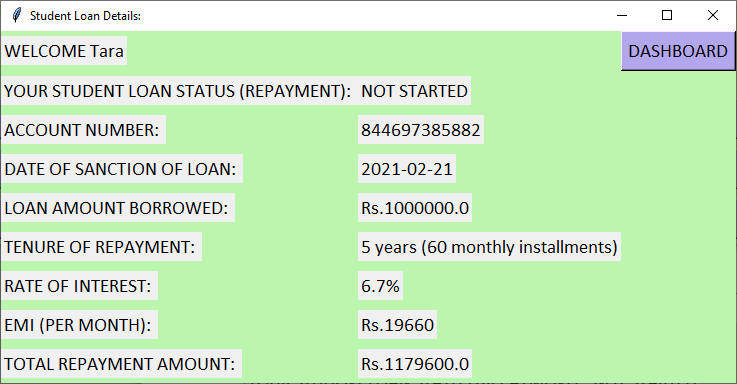
She notes down her loan account number and clicks “CLOSE AND GO TO DASHBOARD”.

The Dashboard opens again, with the third option saying “VIEW STUDENT LOAN DETAILS/ PAY EMI”



Tara clicks “VIEW STUDENT LOAN DETAILS/ PAY EMI”

The following window appears:



These are the details of Tara’s student loan. The Repayment Status shows “NOT STARTED” as it hasn’t been 1 year since she received the loan.

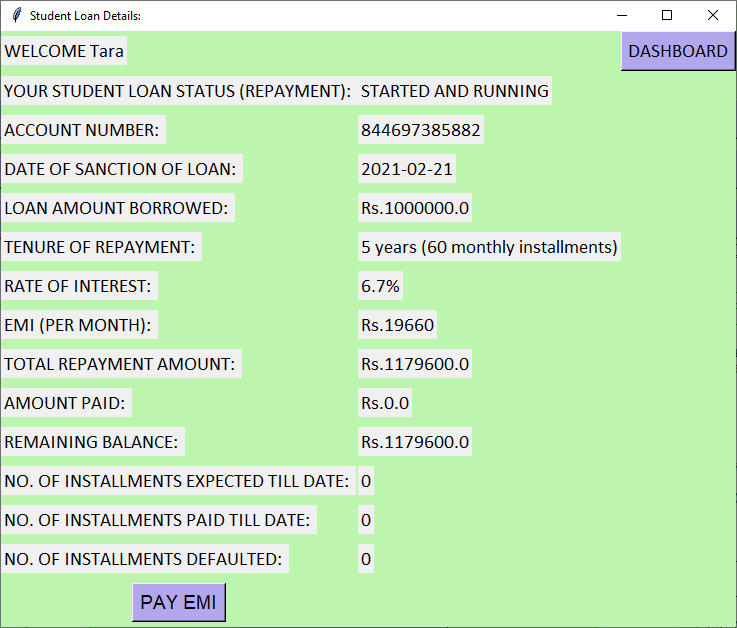
Hence, the “PAY EMI” option is not yet available for her.

She can start paying her EMI only from 2022-02-21.

The above window will display the same details from 2021-02-21 till 2022-02-20.

Once Repayment Tenure starts, she would have to pay Rs.19,660 as EMI for 60 months.

On, 2022-02-21, her window Student Loan details screen would appear as follows.



Note:

Loan repayment Status has changed to “STARTED AND RUNNING”.

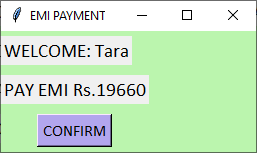
New Columns are being displayed now: starting from AMOUNT PAID till NO. OF INSTALLMENTS DEFAULTED.

(These columns were not displayed when the status was “NOT STARTED”)

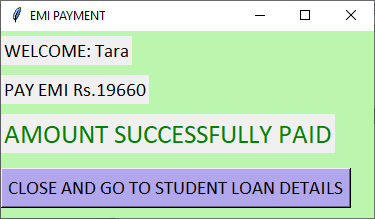
No. of Instalments expected till date is still zero. It would become 1 only after 30 more days since the bank expects the payment of the instalment only by the end of a month.

Suppose Tara decides to make a payment now. She clicks “PAY EMI”

A small window opens as follows:

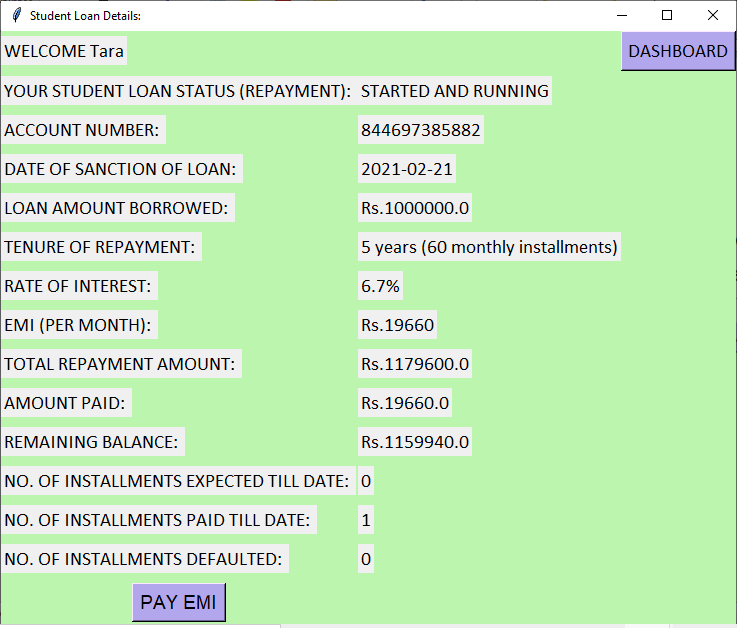


She clicks CONFIRM. A message appears in the same window.



She clicks CLOSE AND GO TO STUDENT LOAN DETAILS.

The Student Loan details window now reopens, refreshed.



Amount paid has become: Rs.19660/-. Total repayment amount has become: Rs.1179600.0 – Rs.19660.0 = Rs.1159940.0/-

No. of Instalments Expected is still zero (as it is expected only at the end of the month (on 2022-03-23), but Tara has paid it in the beginning itself)

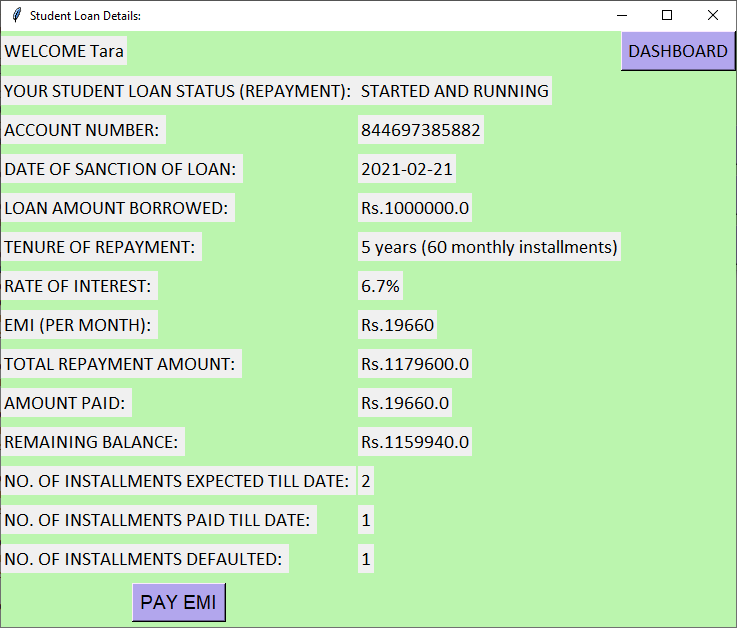
No. of Instalments Paid has become 1.

There aren’t any defaults yet.

Suppose, Tara doesn’t pay her EMI for about, the next 2 months, i.e. till 2022-05-15, she was expected to pay 2 instalments in total (one, on or before 2022-03-23 and the next, on or before 2022-04-22)

She has paid the first instalment but has defaulted on the second one.

Thus, the student loan details page would be as follows in May 2022.



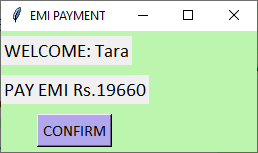
Moneta bank increases EMI and Interest Rate only when a customer defaults more than 3 consecutive times.

Tara has defaulted one instalment.

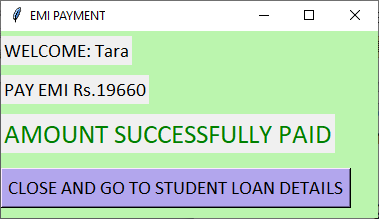
Assuming she pays that skipped instalment on 2022-05-15:

(Now, *today is 2022-05-15*)

Tara clicks PAY EMI.



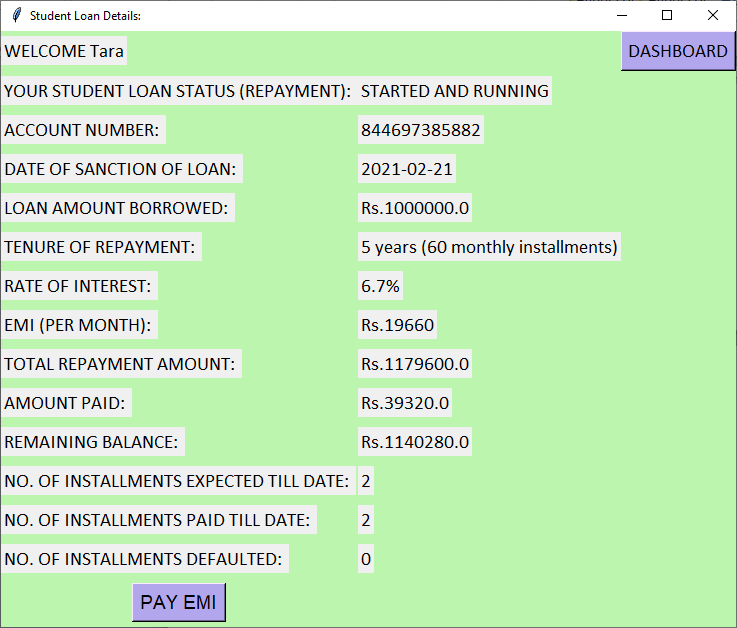
She clicks CONFIRM.



Confirmation message for payment is displayed.

She clicks “CLOSE AND GO TO STUDENT LOAN DETAILS”.

Immediately, the refreshed student Loan details appear as follows:



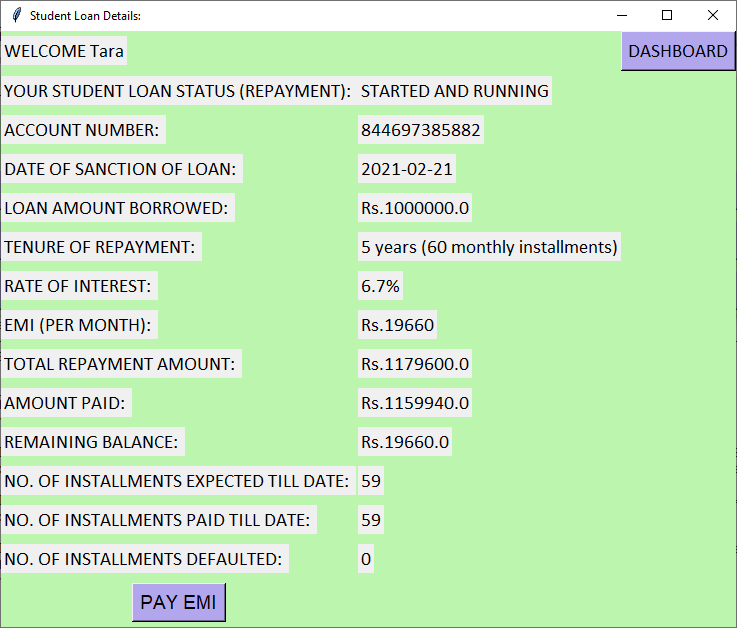
No. of Instalments Expected and Paid are both 2 and hence No. of Defaults has become zero.

Amount Paid and Remaining Balance also change accordingly.

Note: The next EMI is expected by 2022-05-23.

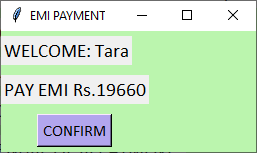
Assuming Tara pays the following instalments on time.

After paying 59 instalments, her Student Loan details would be as follows.

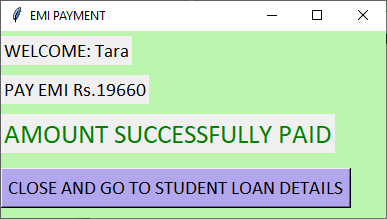


Tara now decides to pay her last instalment and complete the repayment of her loan.

She clicks “PAY EMI”.



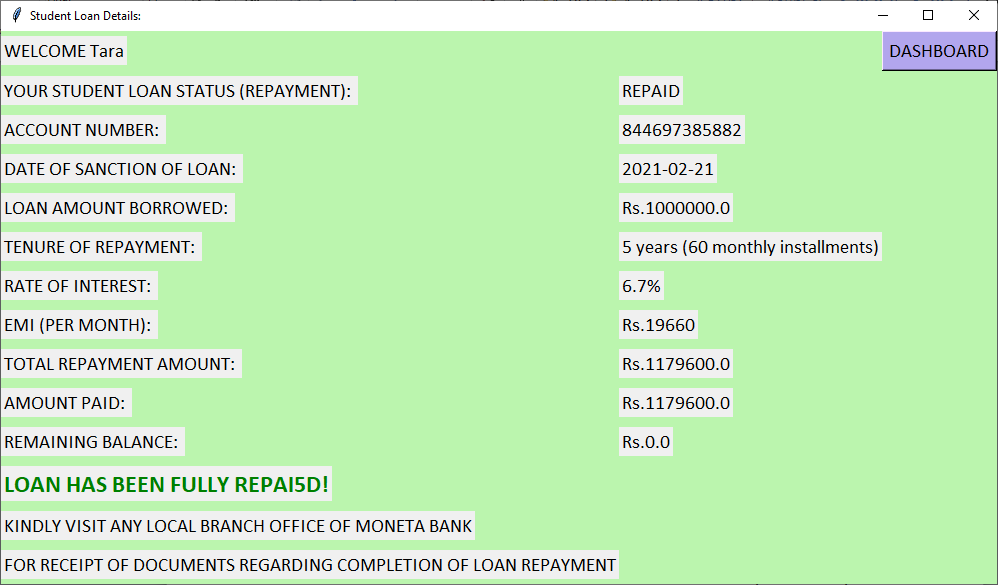
In the new window that appears as above, she clicks CONFIRM.



Confirmation of successful payment appears in the same window.

She clicks “CLOSE AND GO TO STUDENT LOAN DETAILS”

The refreshed Student Loan details window opens:



The customer has paid the total repayment amount and remaining balance to be paid is zero

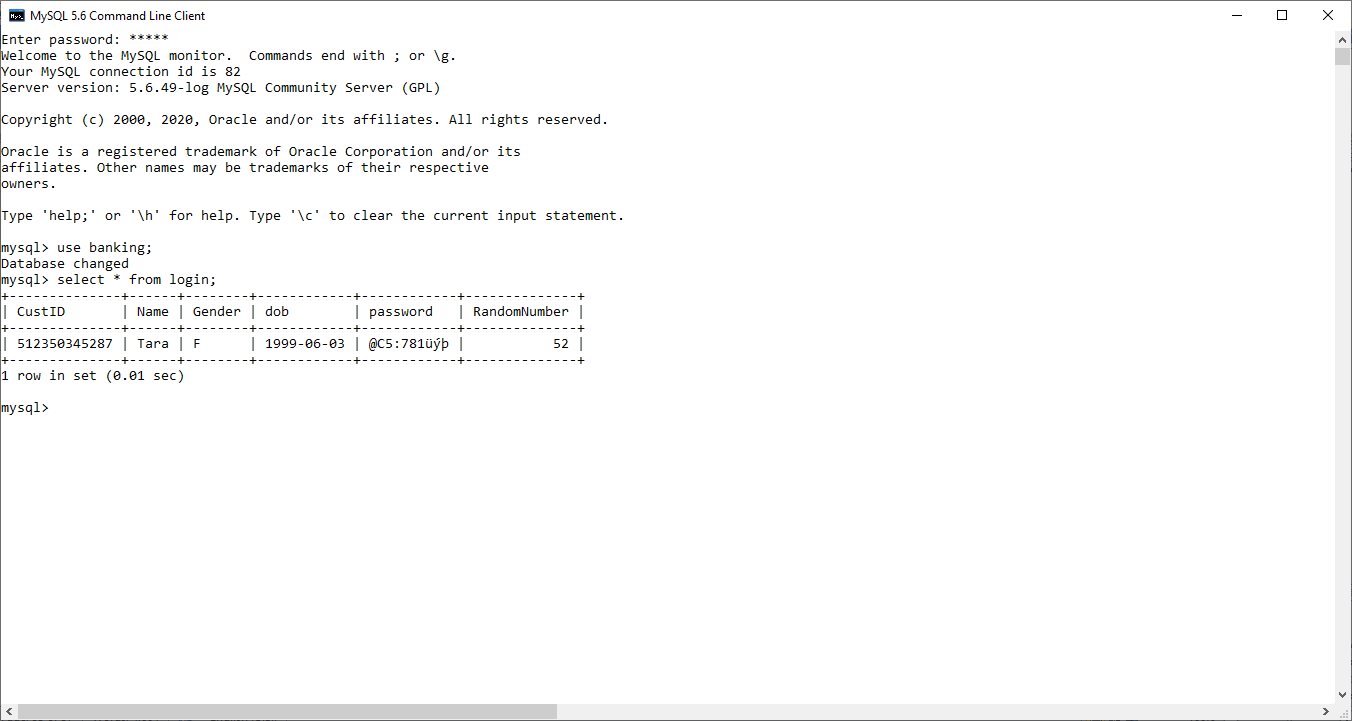
Customer Tara has successfully completed her loan payment!

She visits her local branch office of Moneta Bank and completes the post-repayment formalities through a quick and transparent process, after which the process is termed complete and the customer would be eligible for borrowing another student loan as well following the same processes as above.

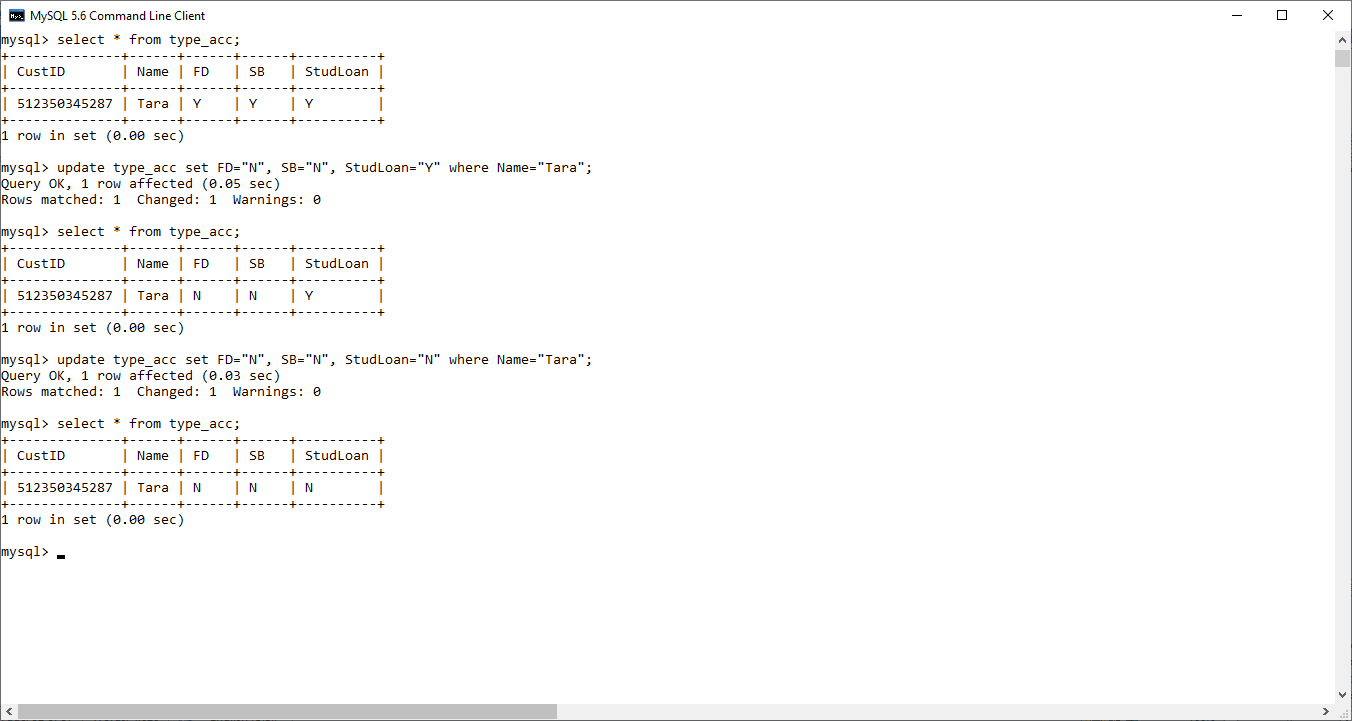
**Screenshots in MySQL**

**Login**

1. On creating Customer Tara’s account in the Python Platform, the MySQL table “login” appears as below;

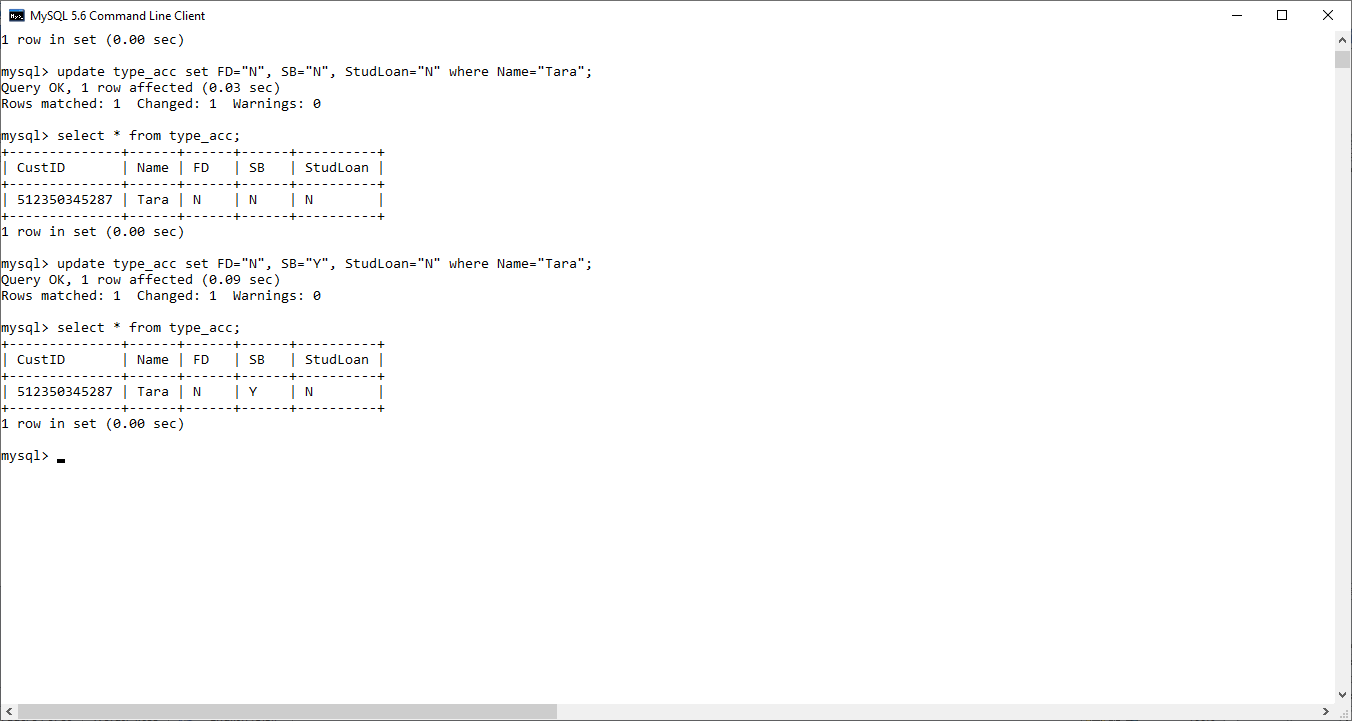


1. The table type\_acc appears as below: (None of the features of the Bank are availed by the customer yet).



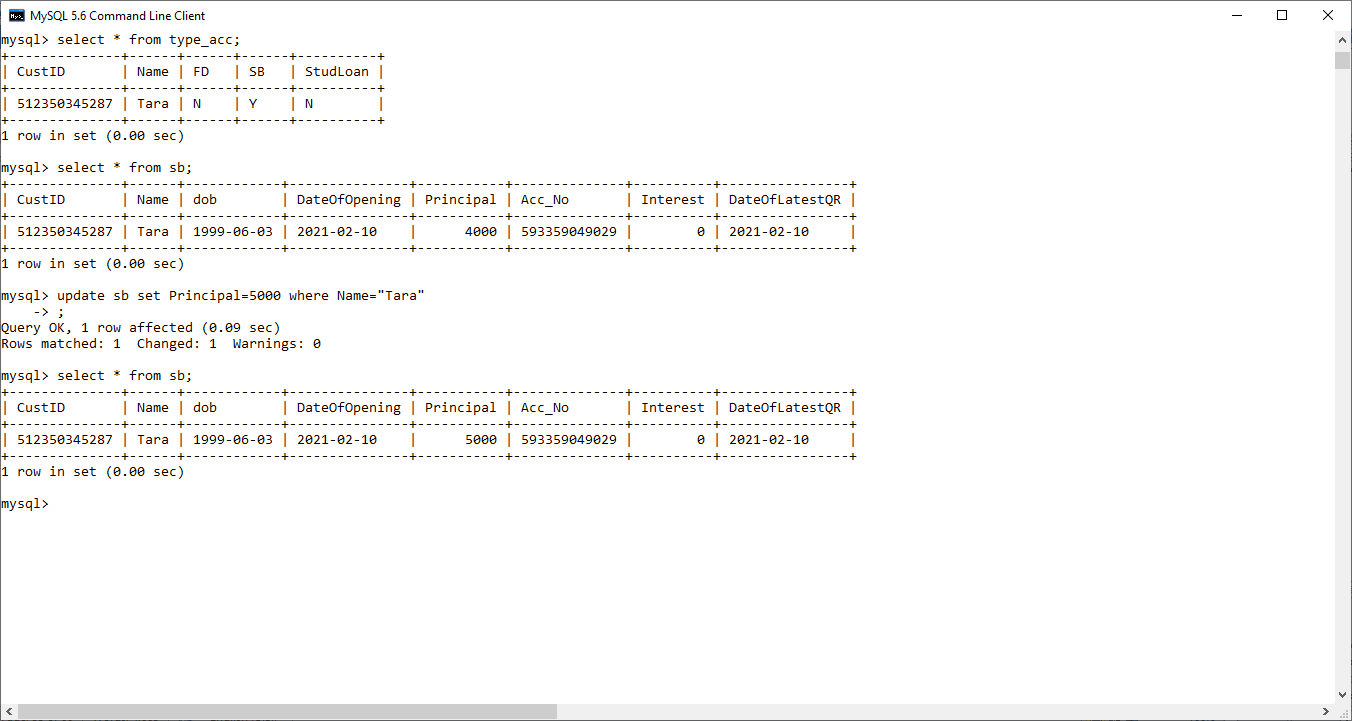
**Creation of Savings Bank Account:**

1. On creating Savings Bank Account with Rs.5000/- , the table type\_acc appears as below:



Value ‘Y’ in SB column implies Savings Account has been created.

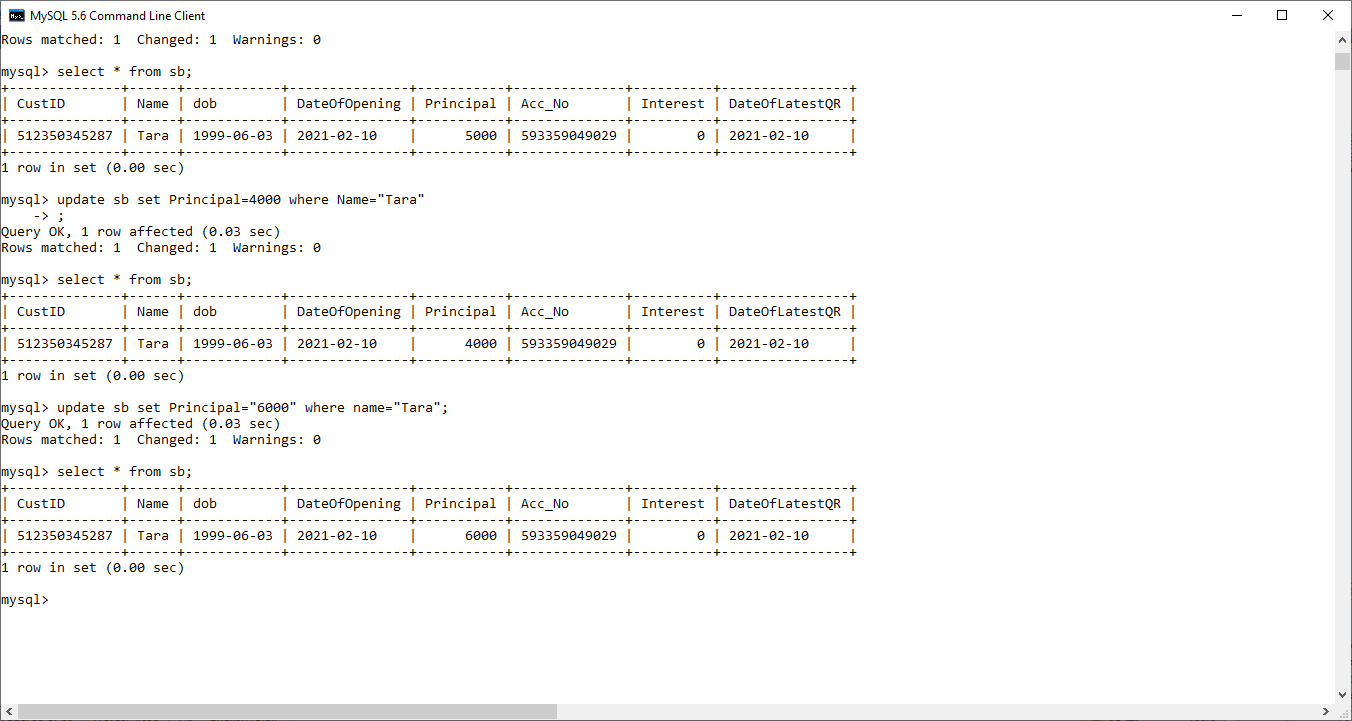
Table ‘sb’ appears as below (Initially the table was empty)



1. After withdrawal of Rs.1000/- the table ‘sb’ appears as below:

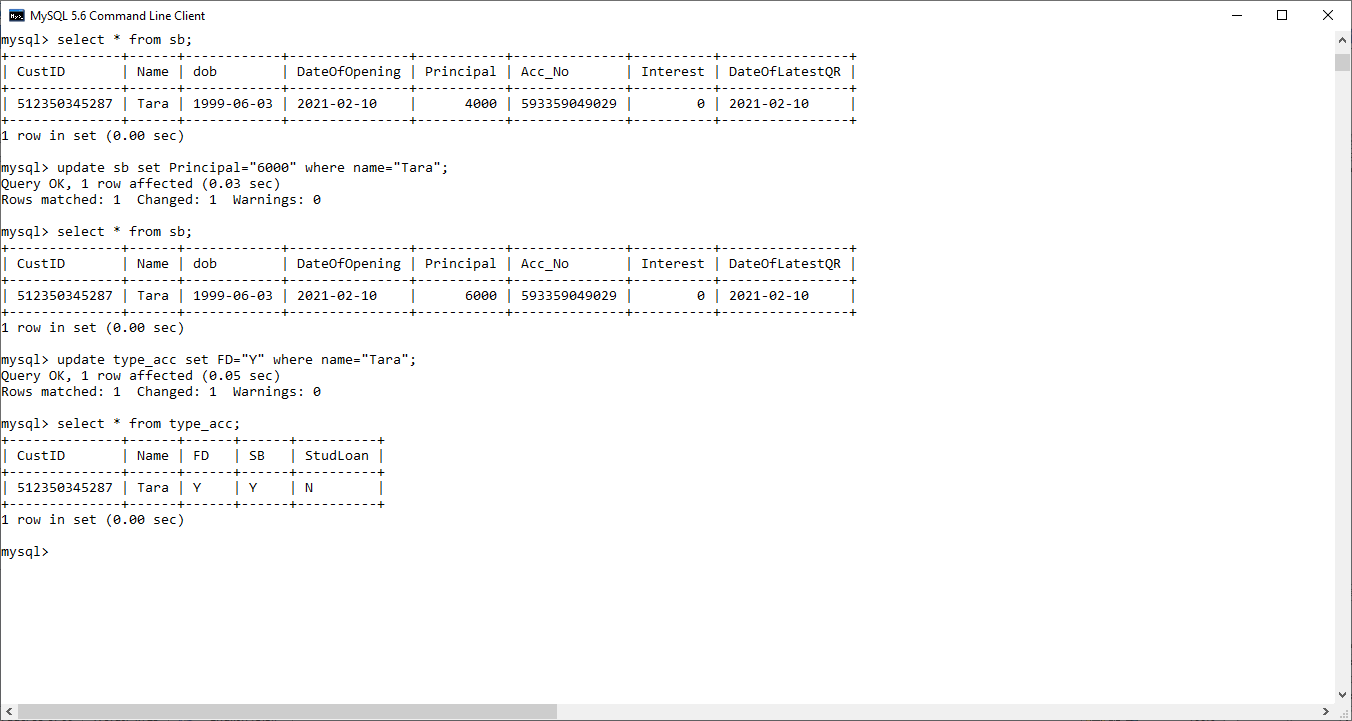


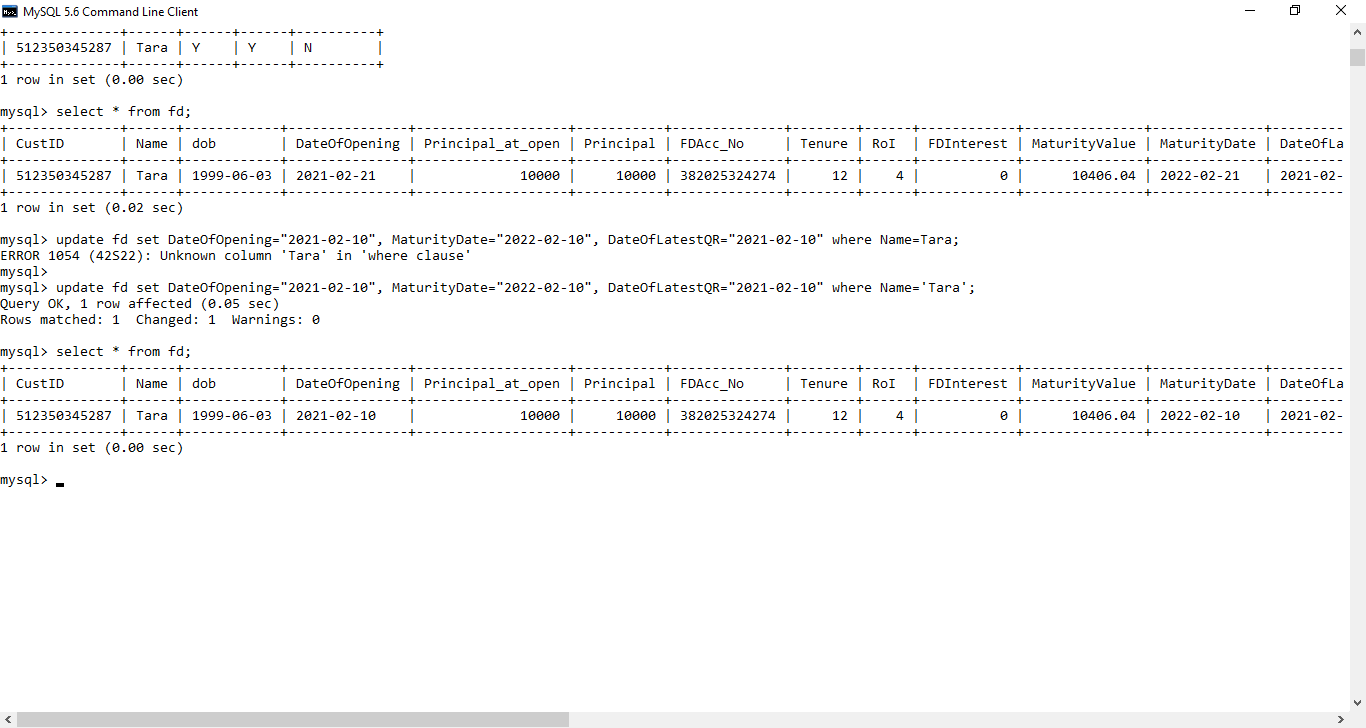
1. After deposit of Rs.2000/- from this status, the table ‘sb’ appears as below:



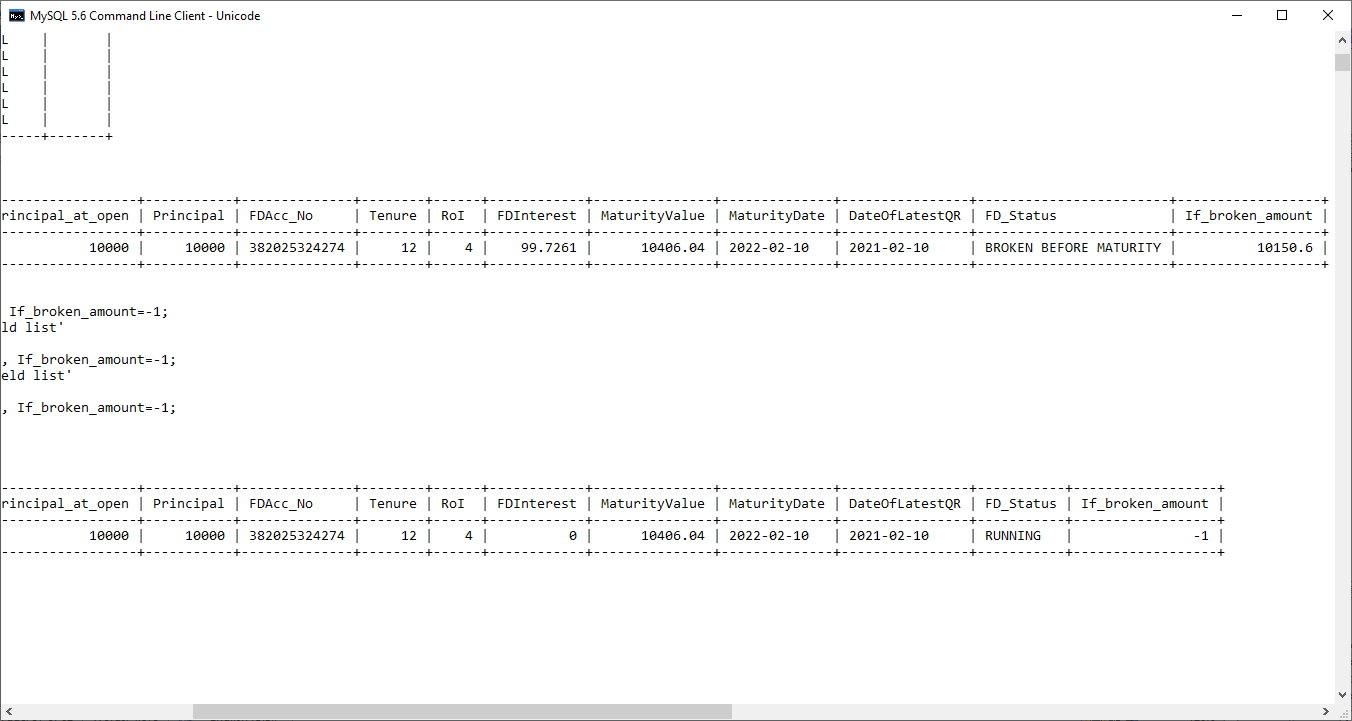
**Creation Of Fixed Deposit Account:**

1. On Creating Fixed Deposit Account for Customer Tara, with deposit amount Rs.10,000/- for a tenure of 12 months, tables ‘type\_acc’ and ‘fd’ appear as below:

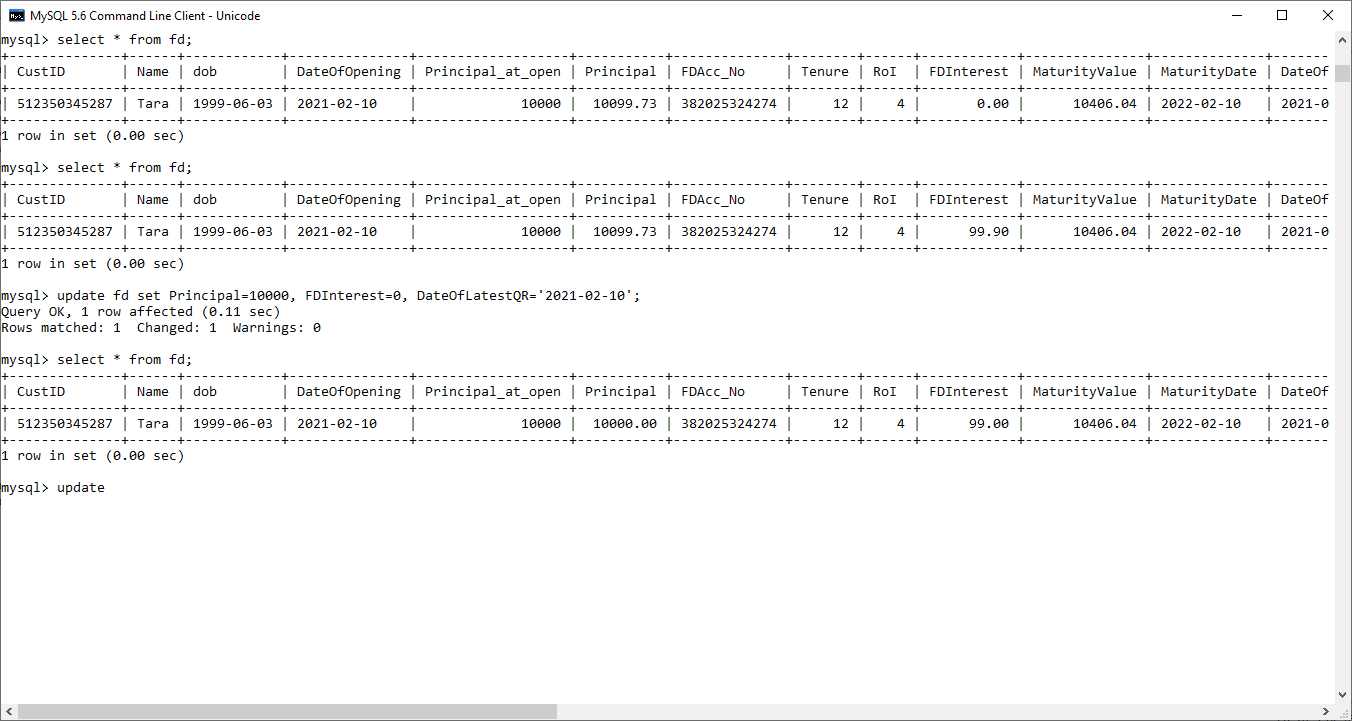


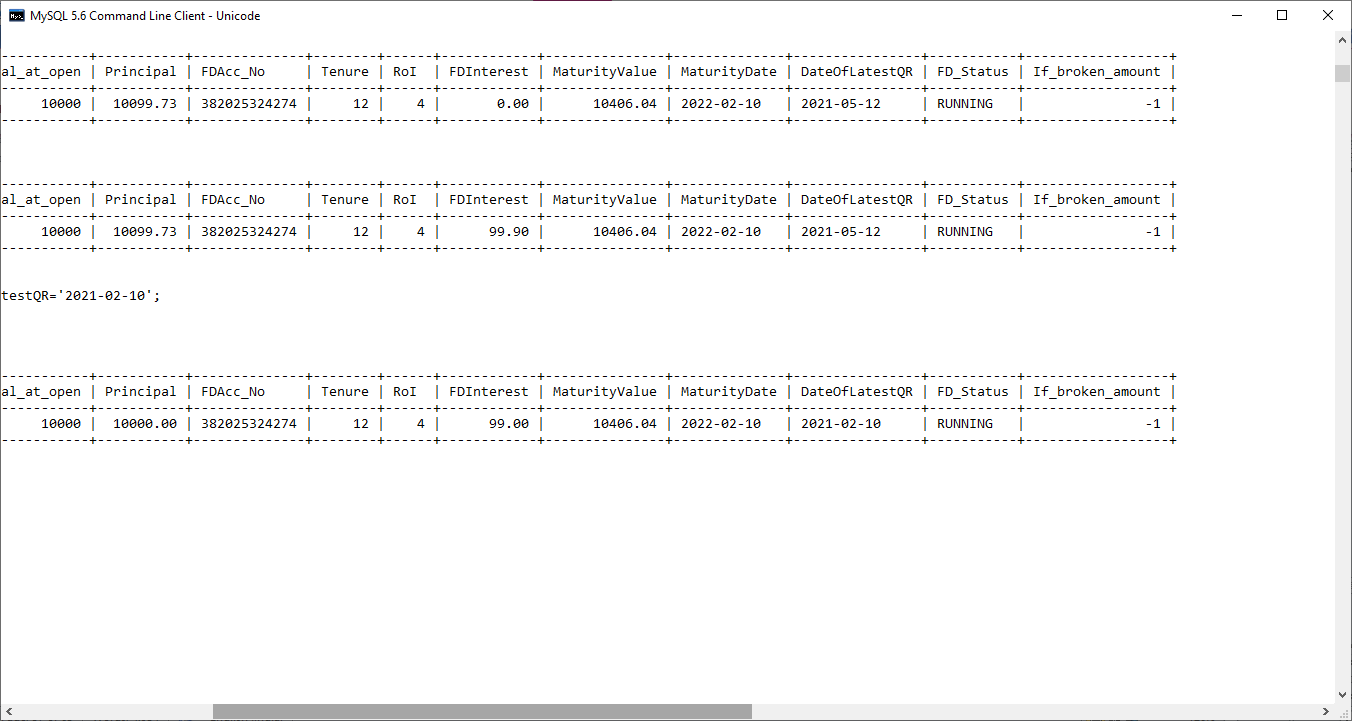


(Continuation of ‘fd’ table….)



1. After 3 months on 2021-05-11 (Just before first quarterly addition of Principal to Interest)

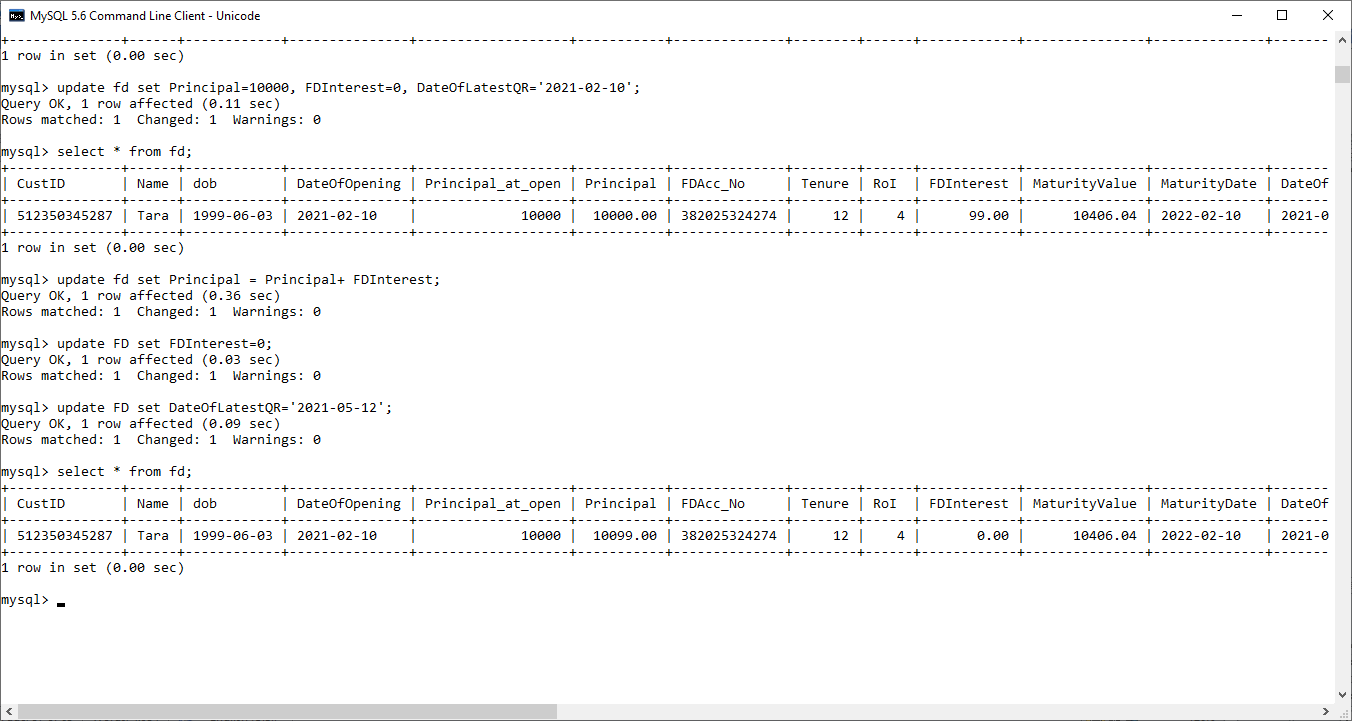


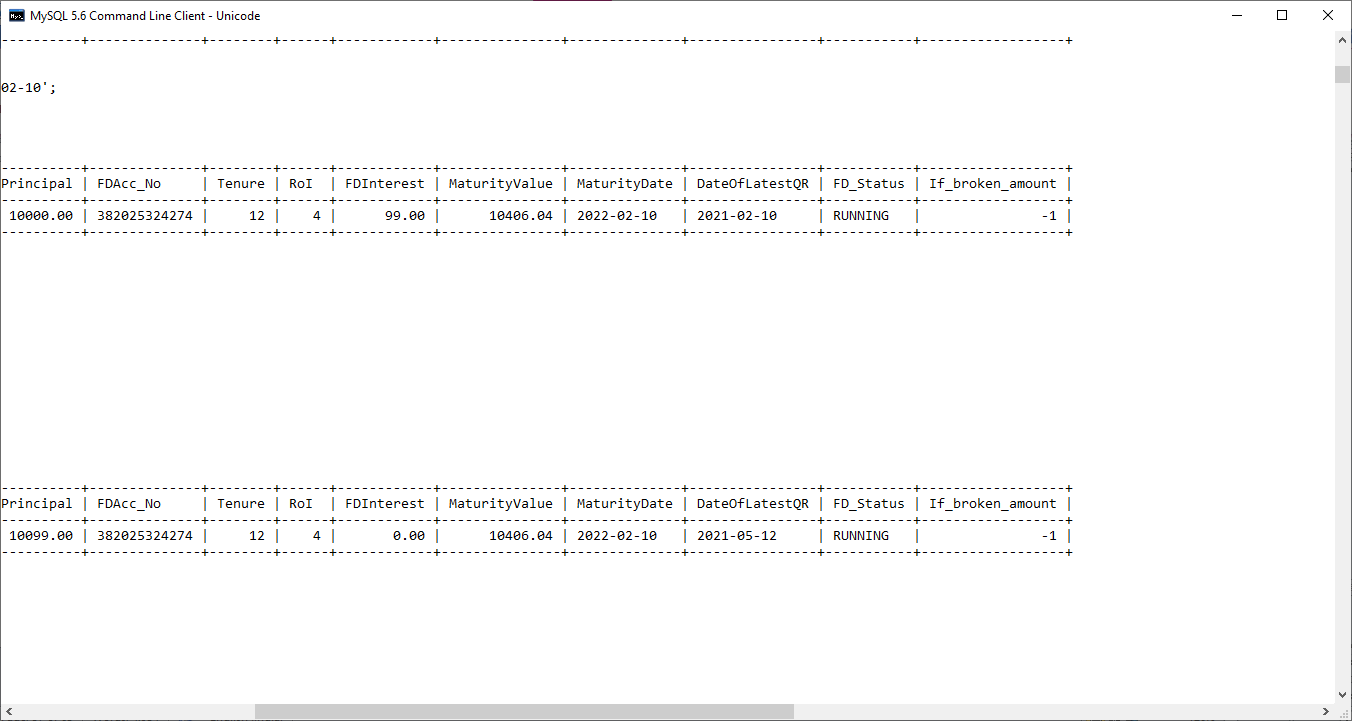


FD has increased over 3 months.

Date of Latest Quarterly Addition (DateOfLatetQR) is still unchanged.

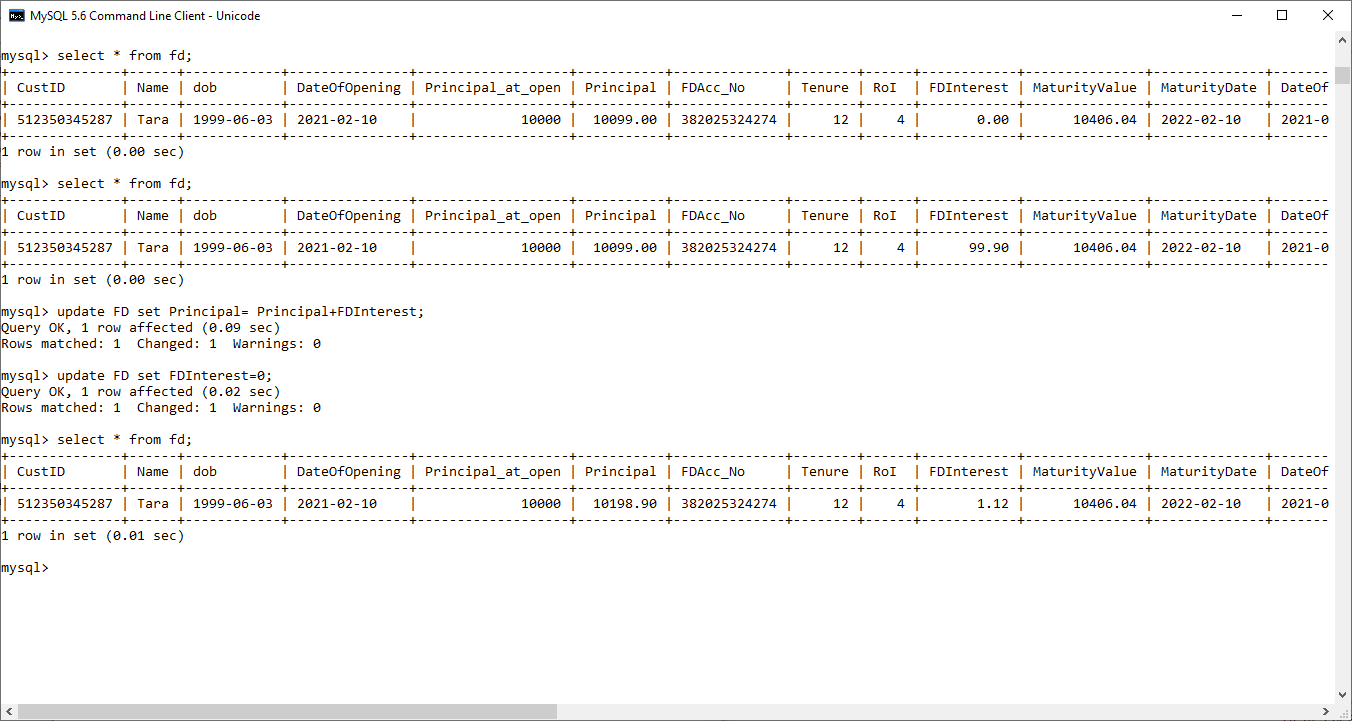
The change will be visible the next day as follows:

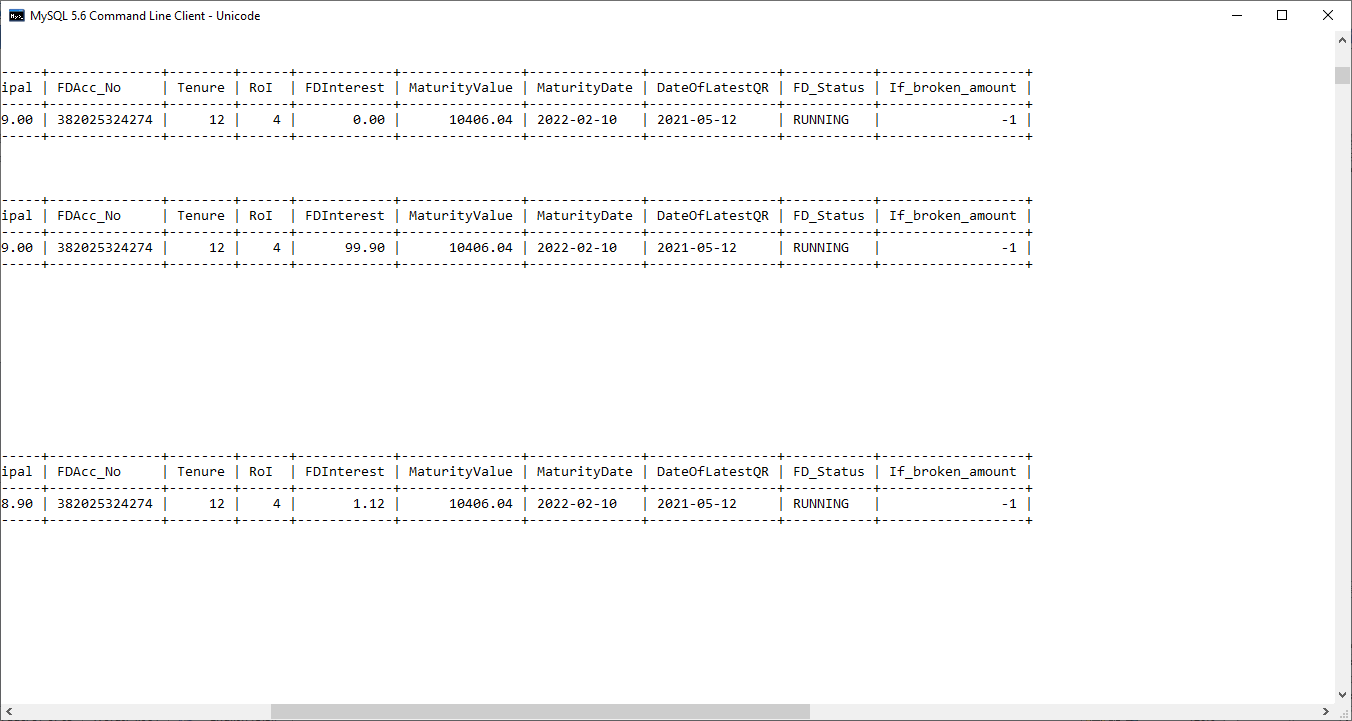




Current Principal (Principal) has been revised. FD has become zero and the Date Of Latest Quarterly addition is also updated.

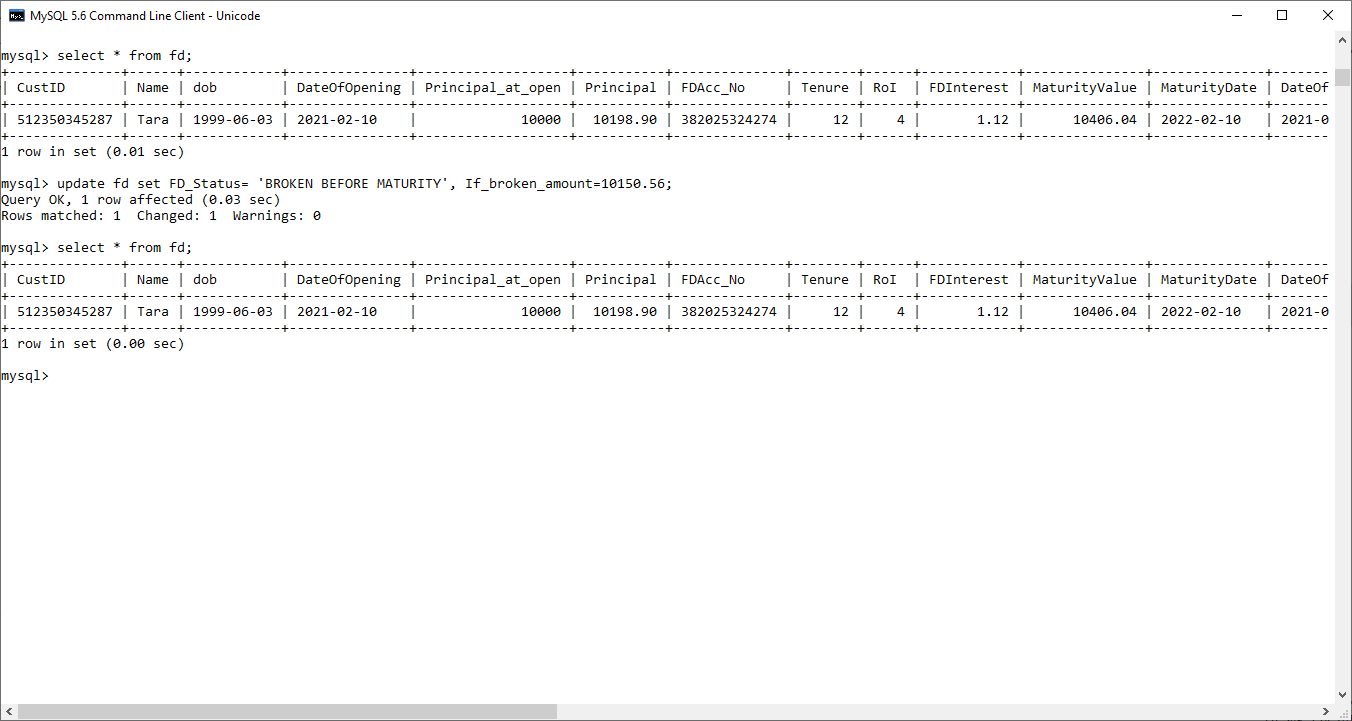
1. Customer Tara’s FD details on 2021-08-10:

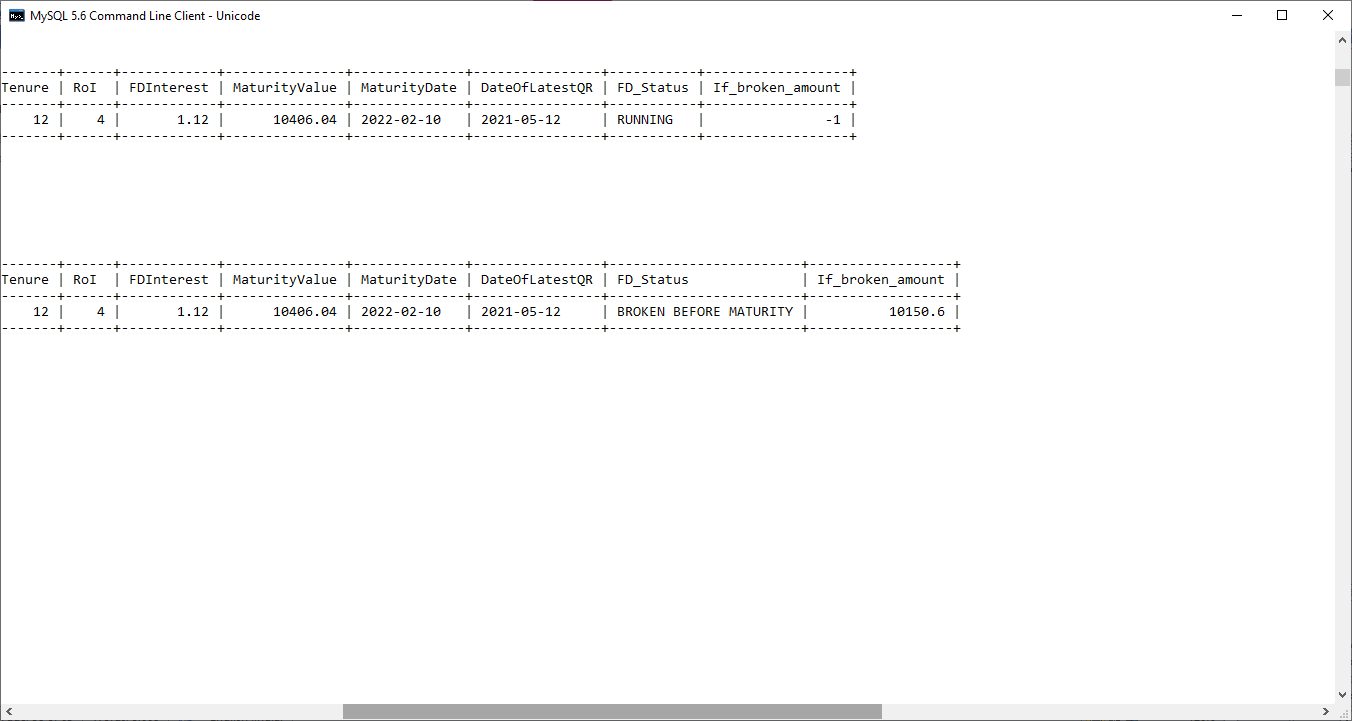




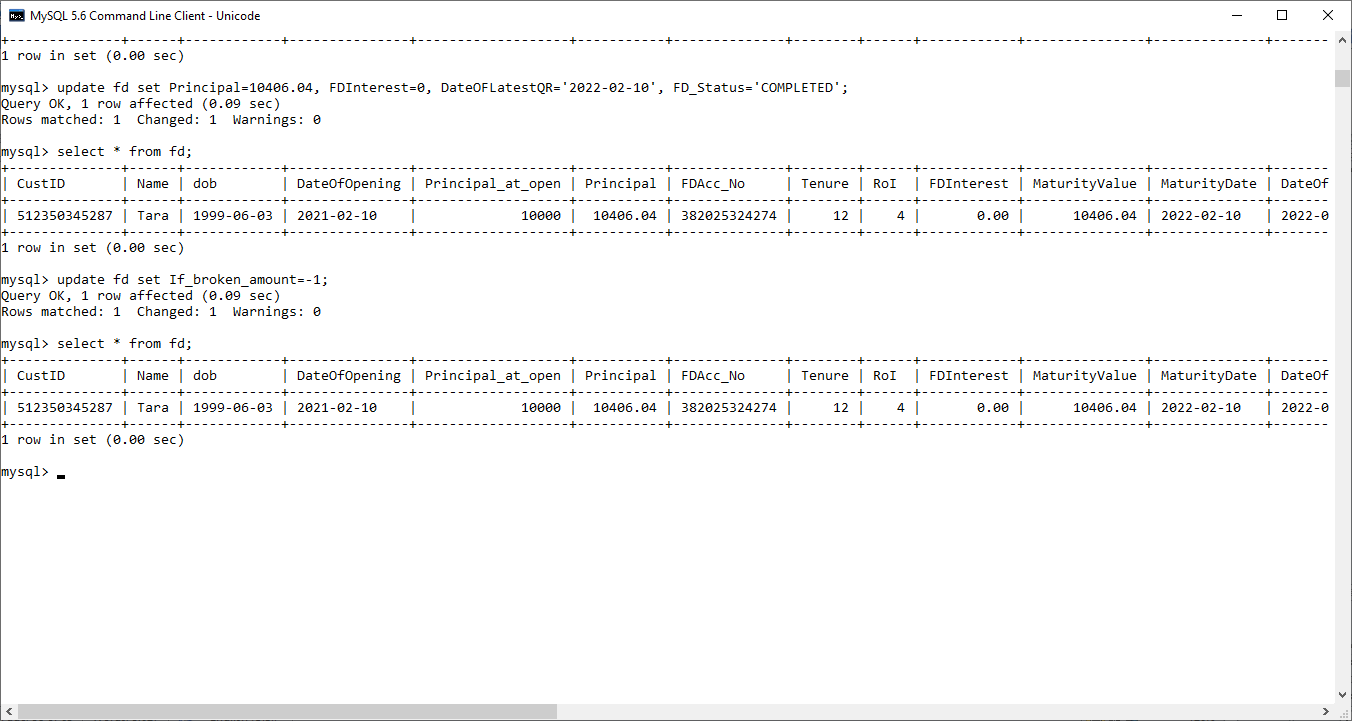
1. In case the customer decides to break her FD on 2021-08-10 (the case shown in Python Screenshot), the amount that she will receive on doing so is calculated by compound interest on the principal deposit **at an interest rate 1% less than the rate applicable** during deposit.

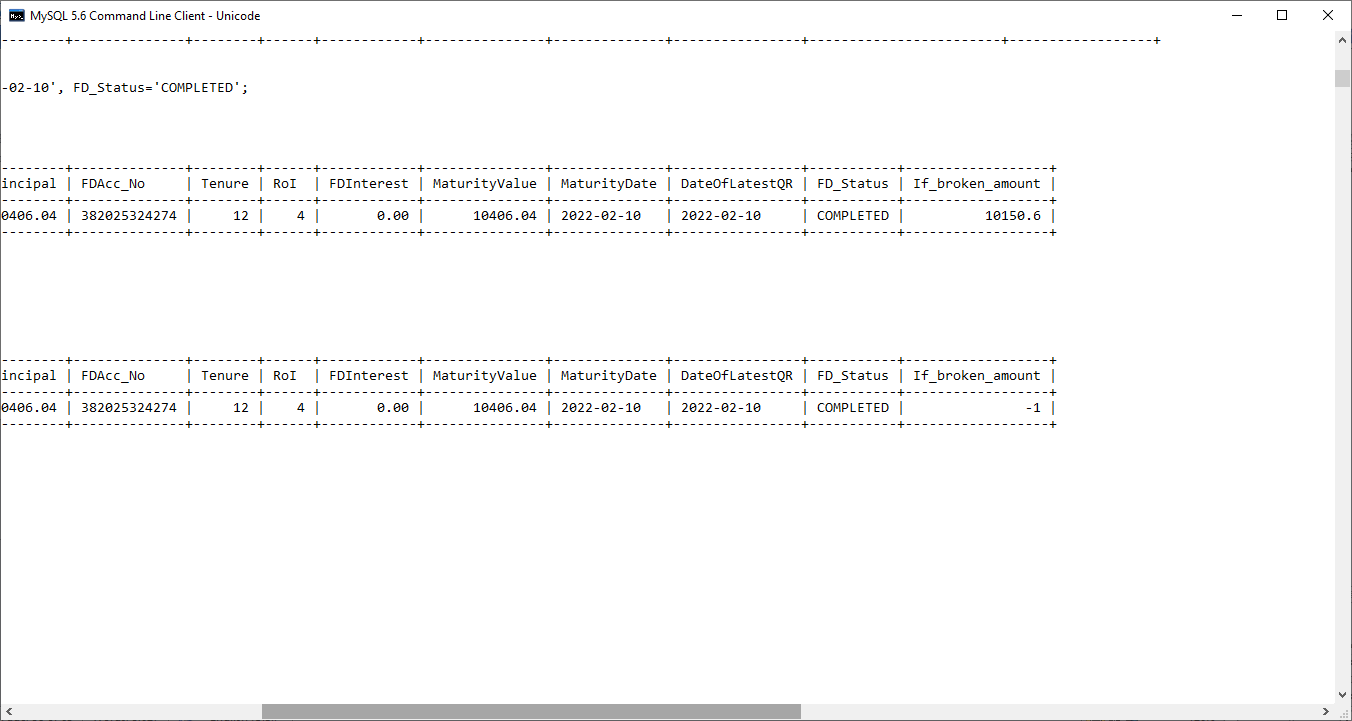
If she breaks her FD on 2021-08-10 she receives an amount of Rs.10,150.56 and this amount will go into ‘If\_broken\_amount’ field and FD\_Status becomes ‘BROKEN BEFORE MATURITY’.





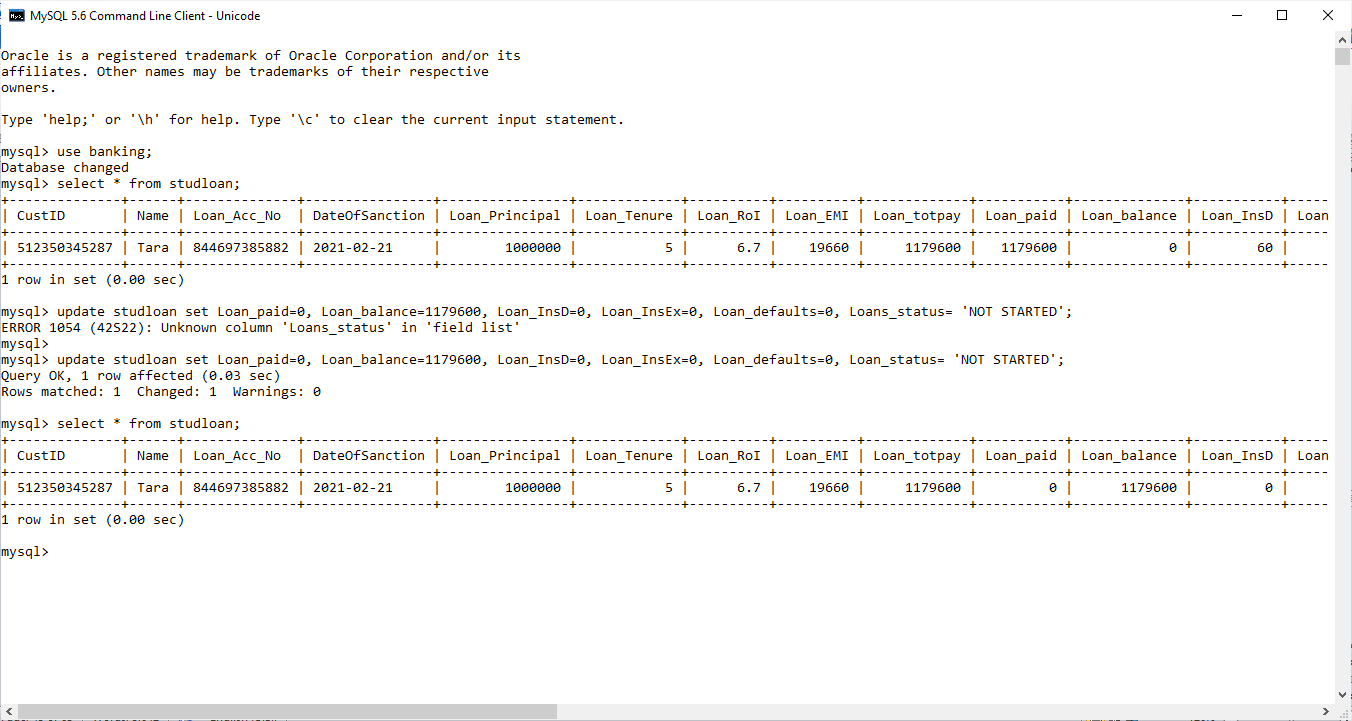
1. If she doesn’t break her FD and retains it till maturity her FD details will finally appear as below.

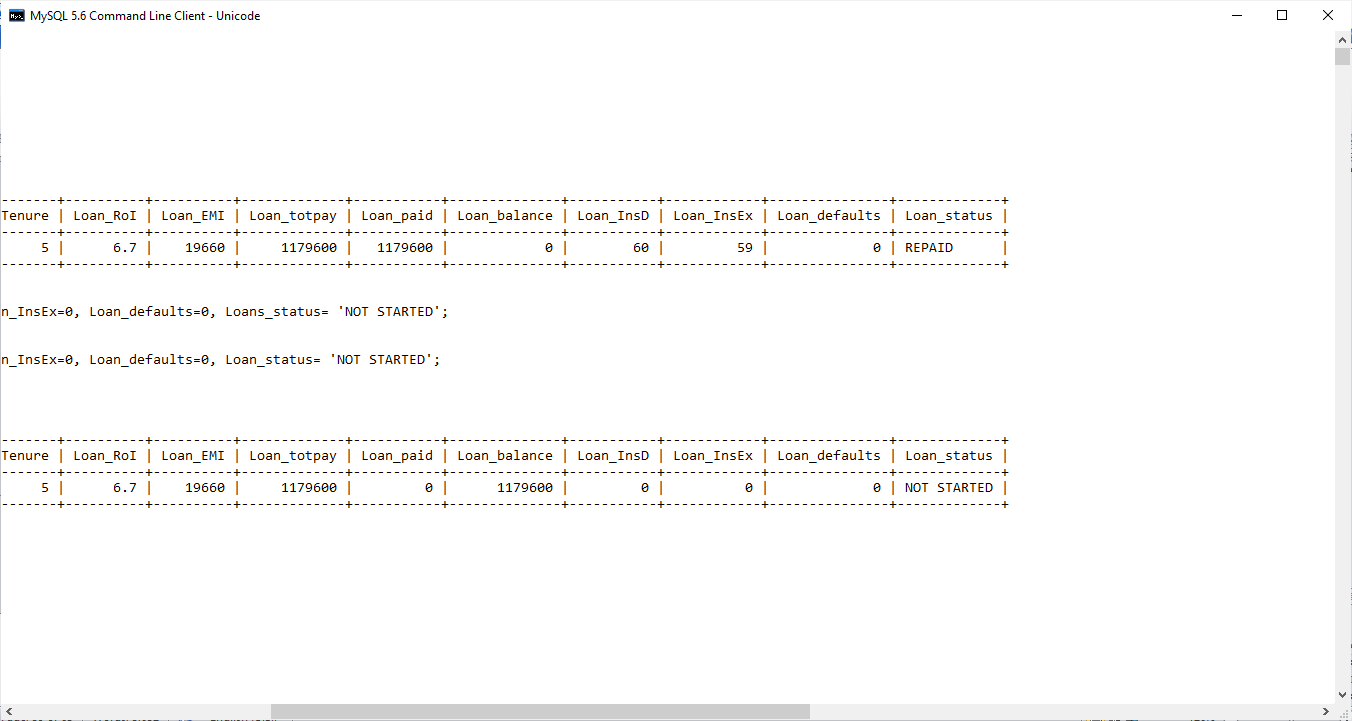




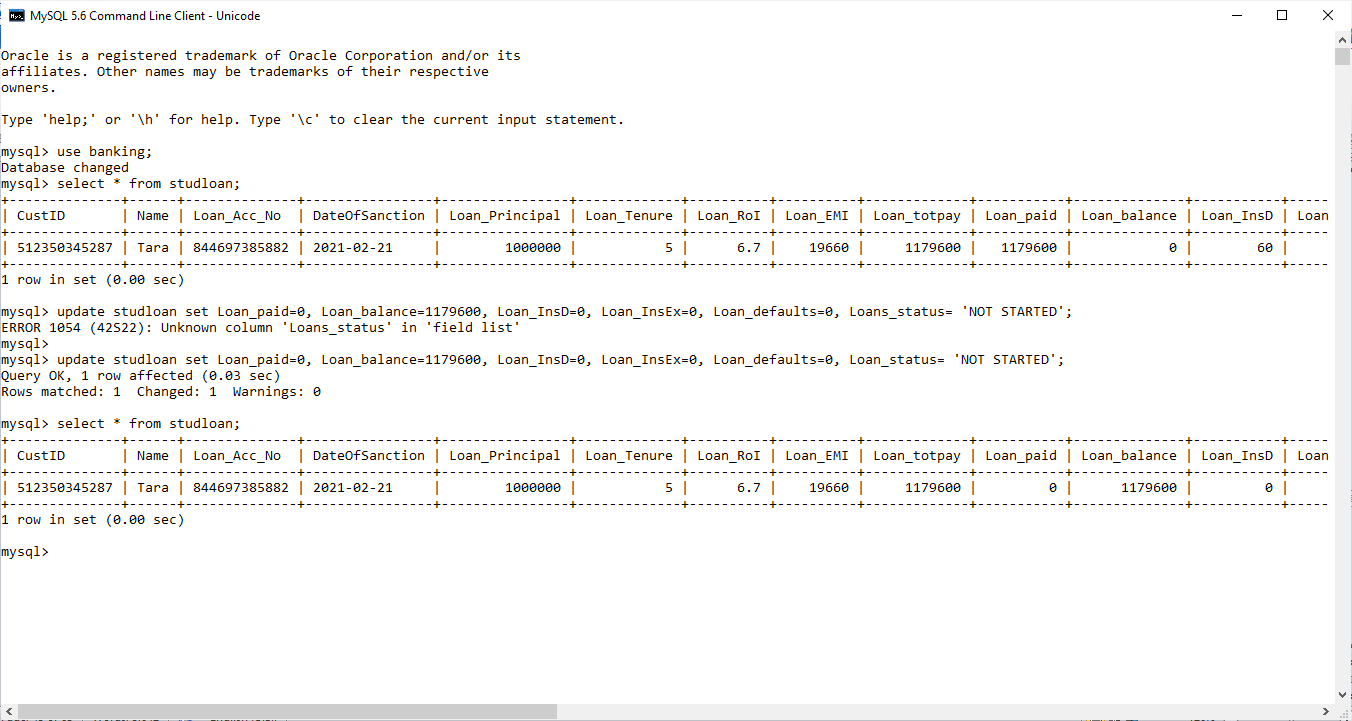
**Applying for Student Loan**

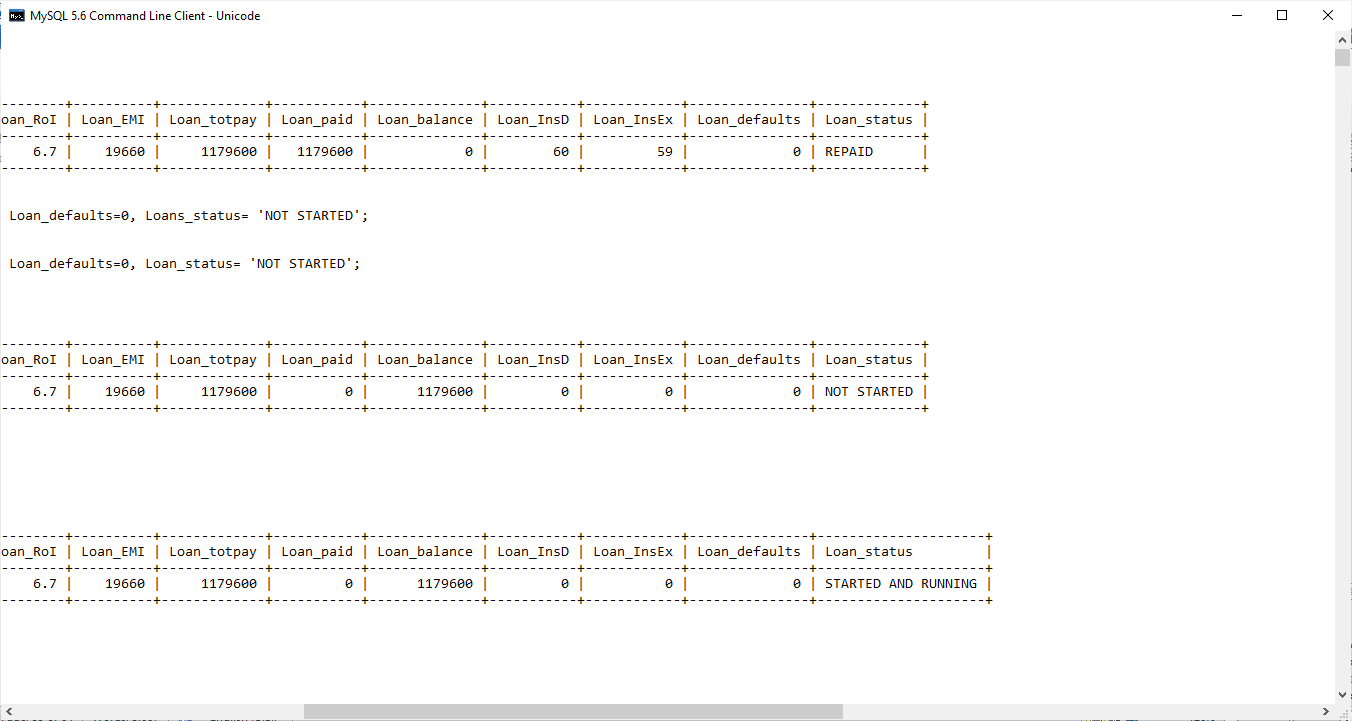
1. Table studloan appears as below for Customer Tara on applying for student loan on 2021-02-21:





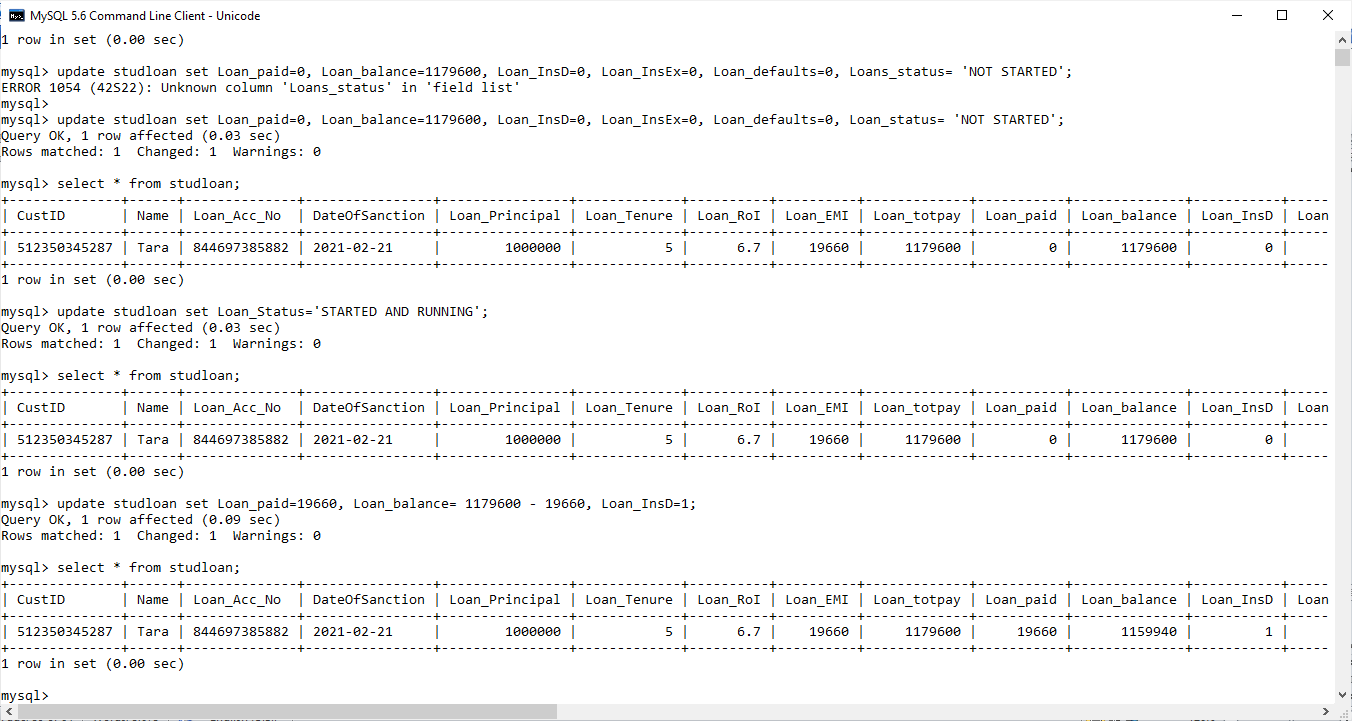
1. After 1 year, on 2022-02-21, the table appears as below:

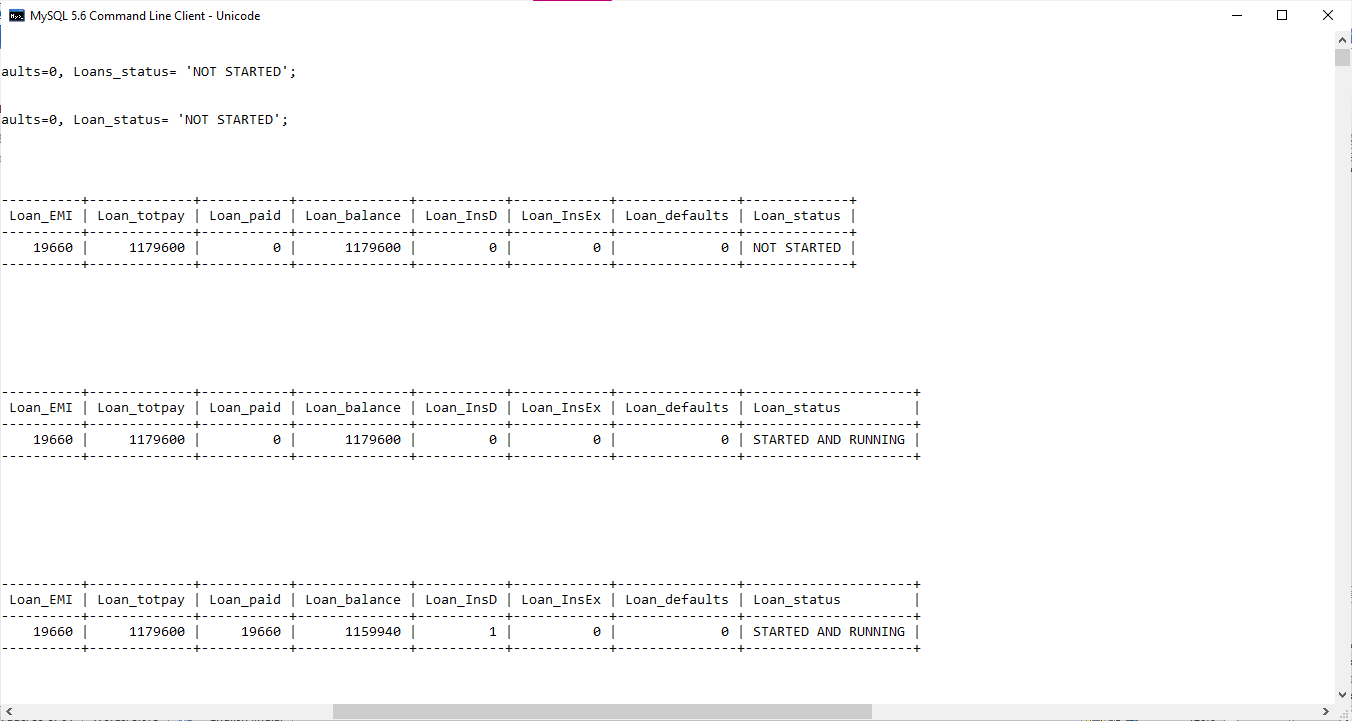




(Status of Loan has become ‘STARTED AND RUNNING’)

1. After paying one installment:

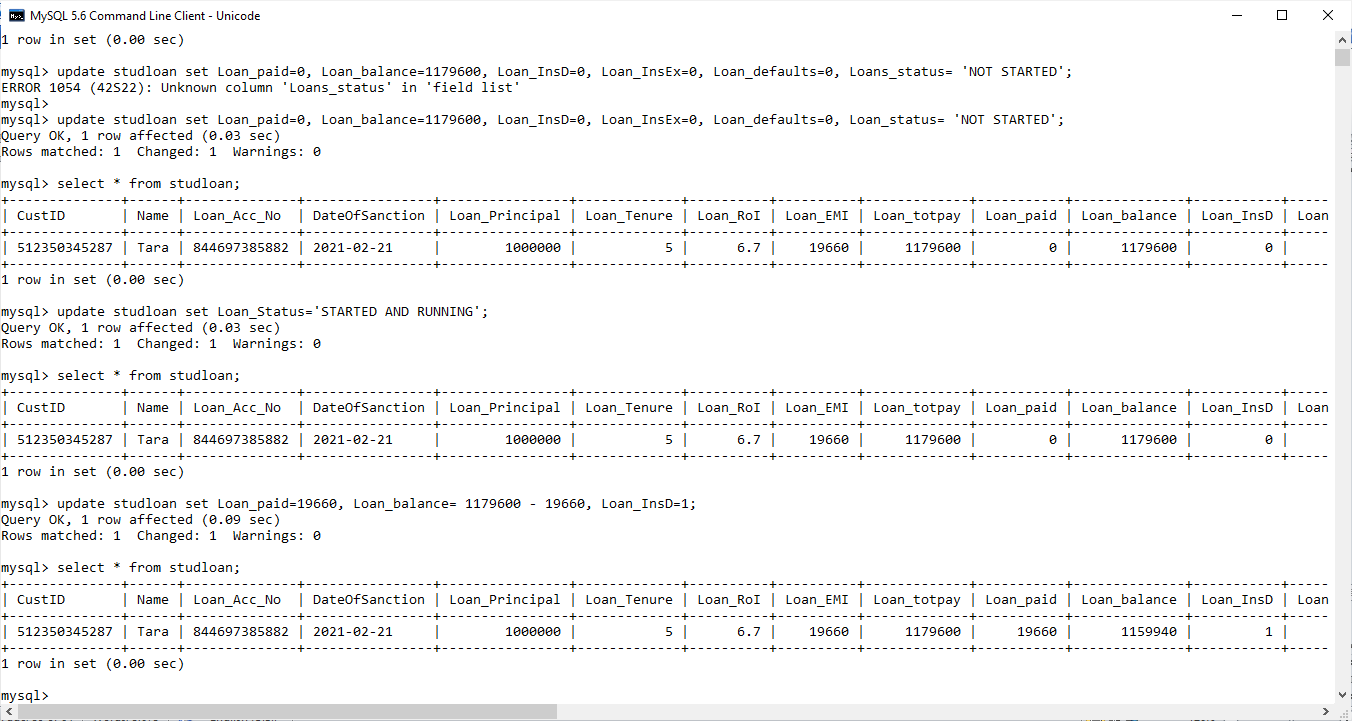


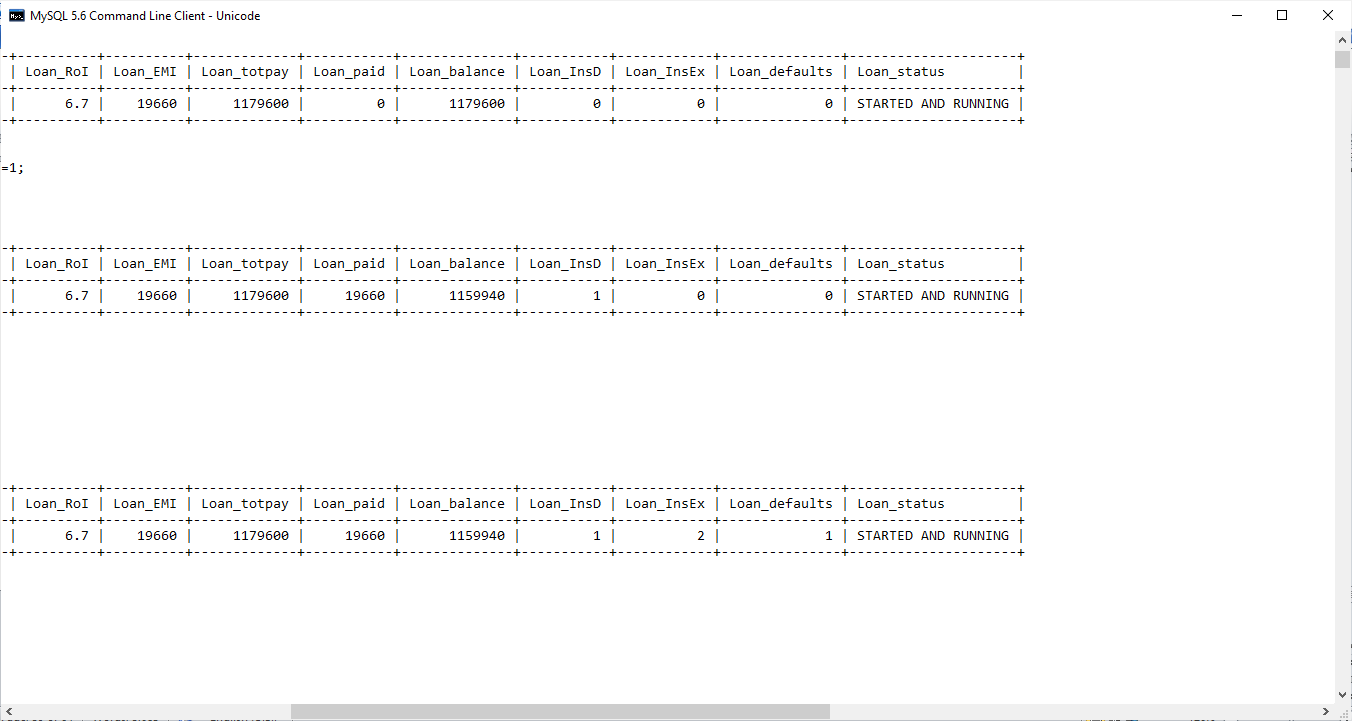


Number of Installments Done (Loan\_insD)=1

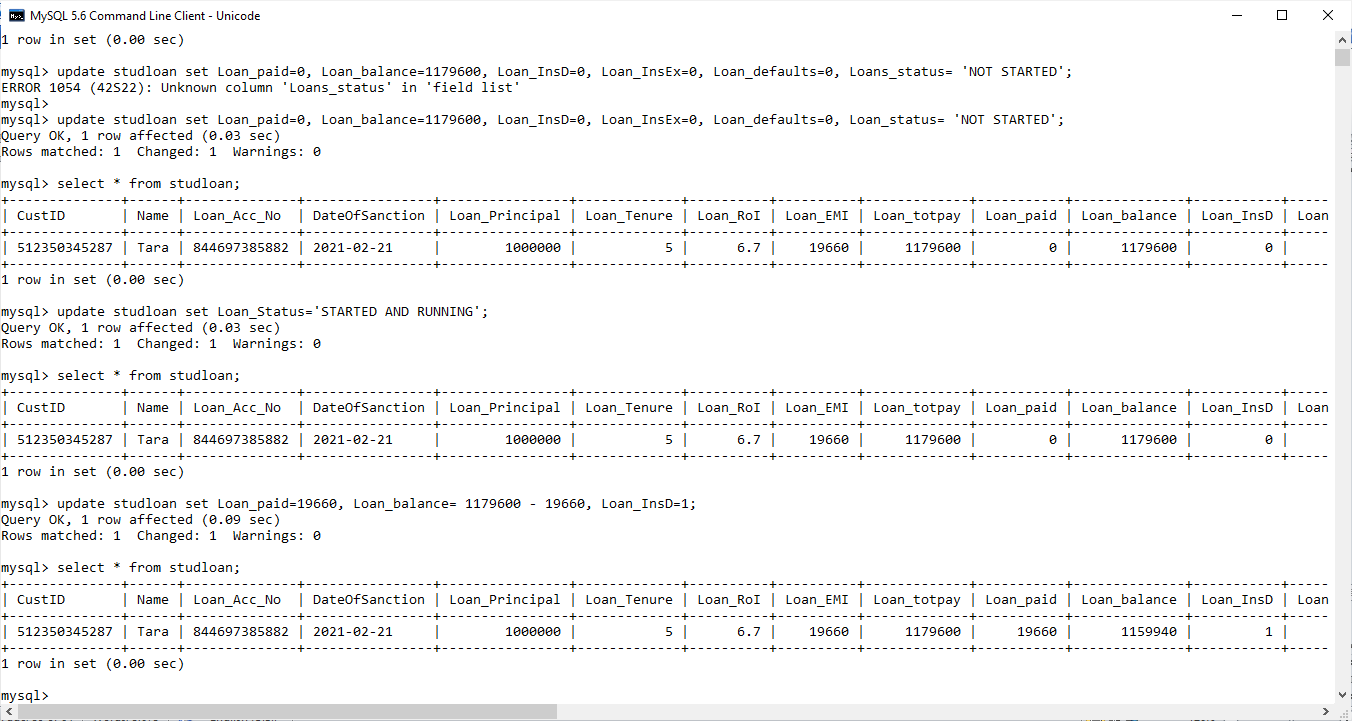
Amount paid and balanced have increased and decreased respectively by one EMI amount.

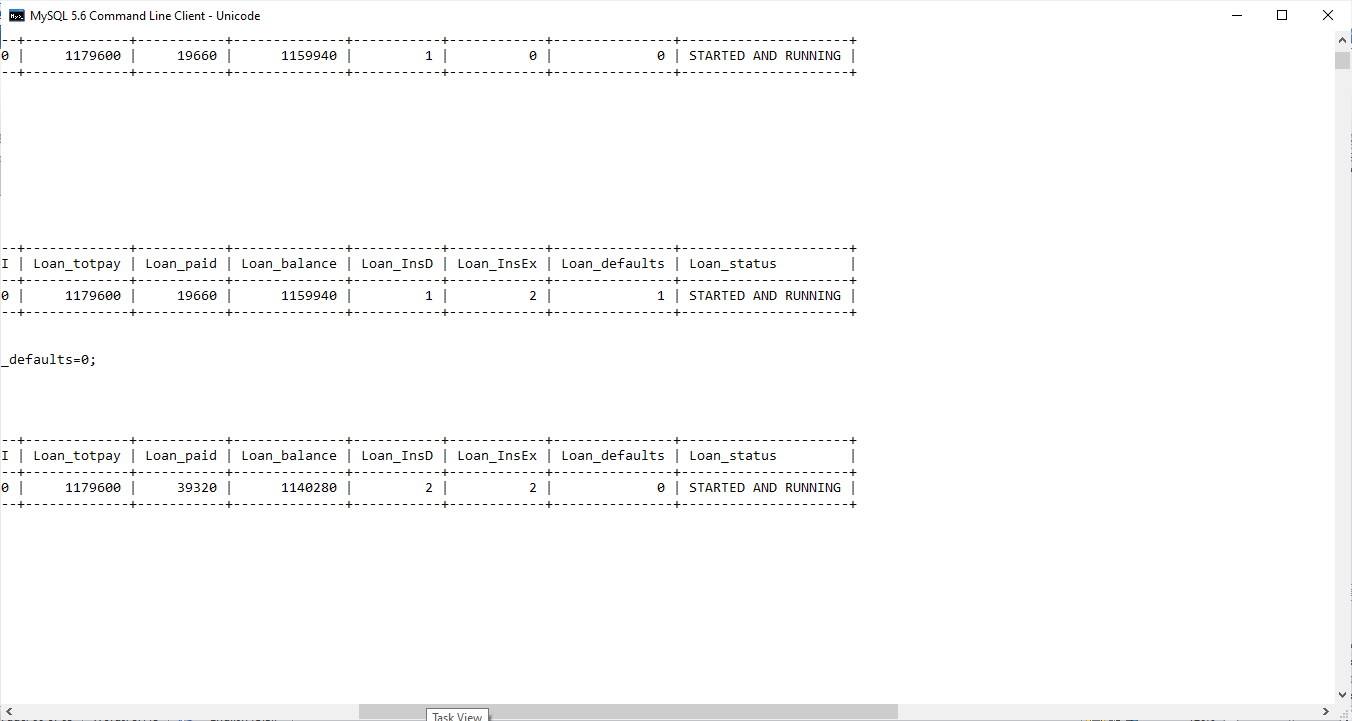
1. After a default of one installment:



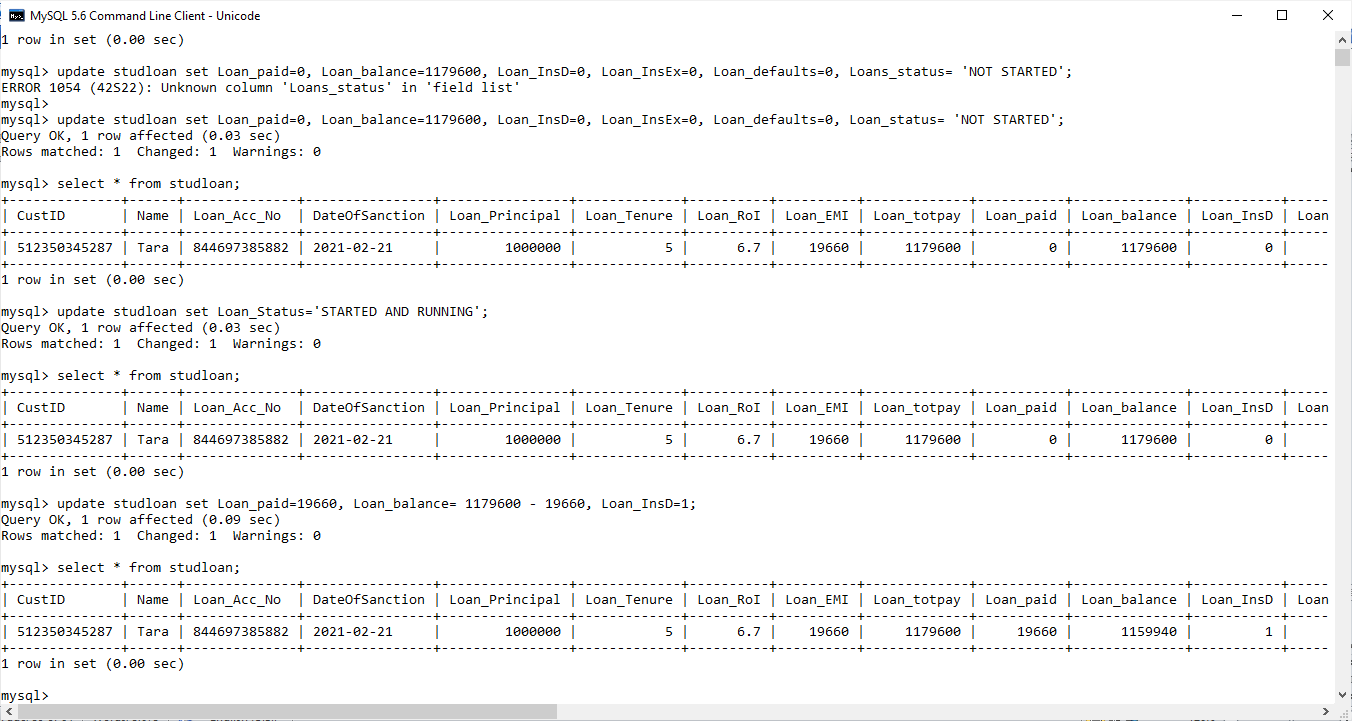


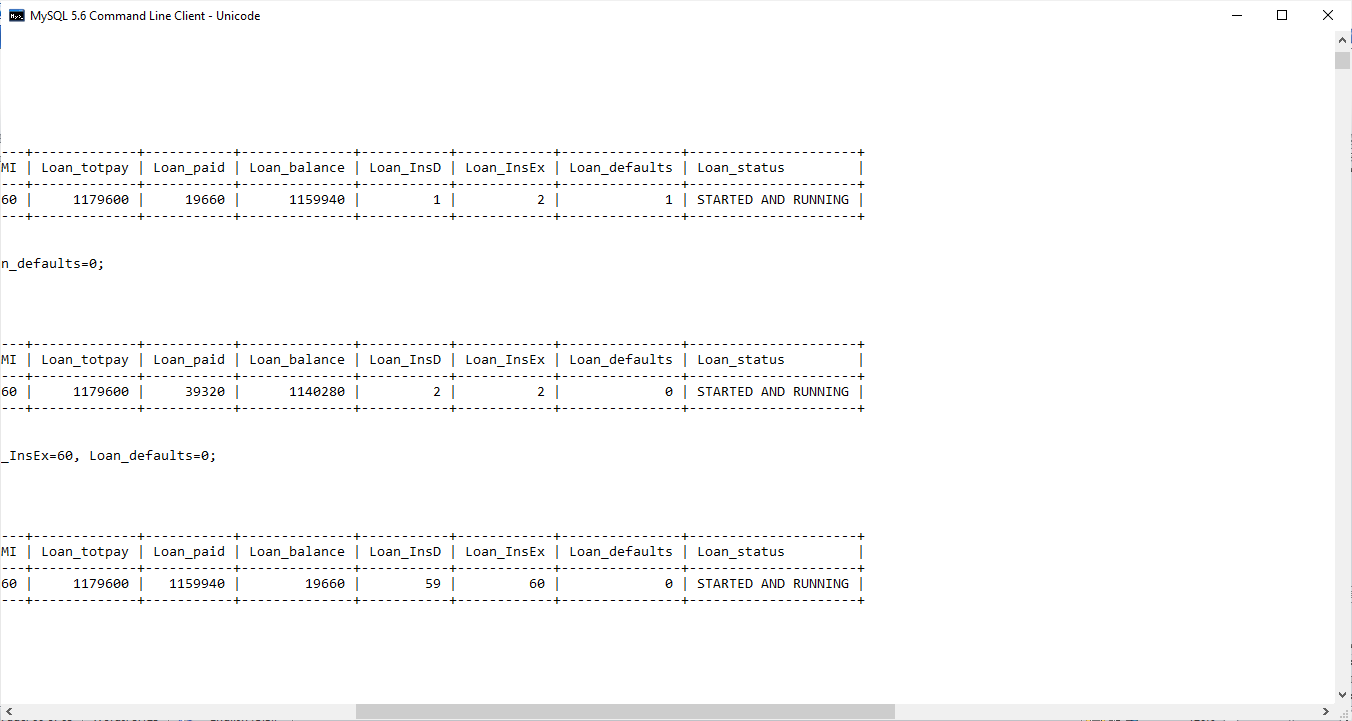
1. On repayment of the default:



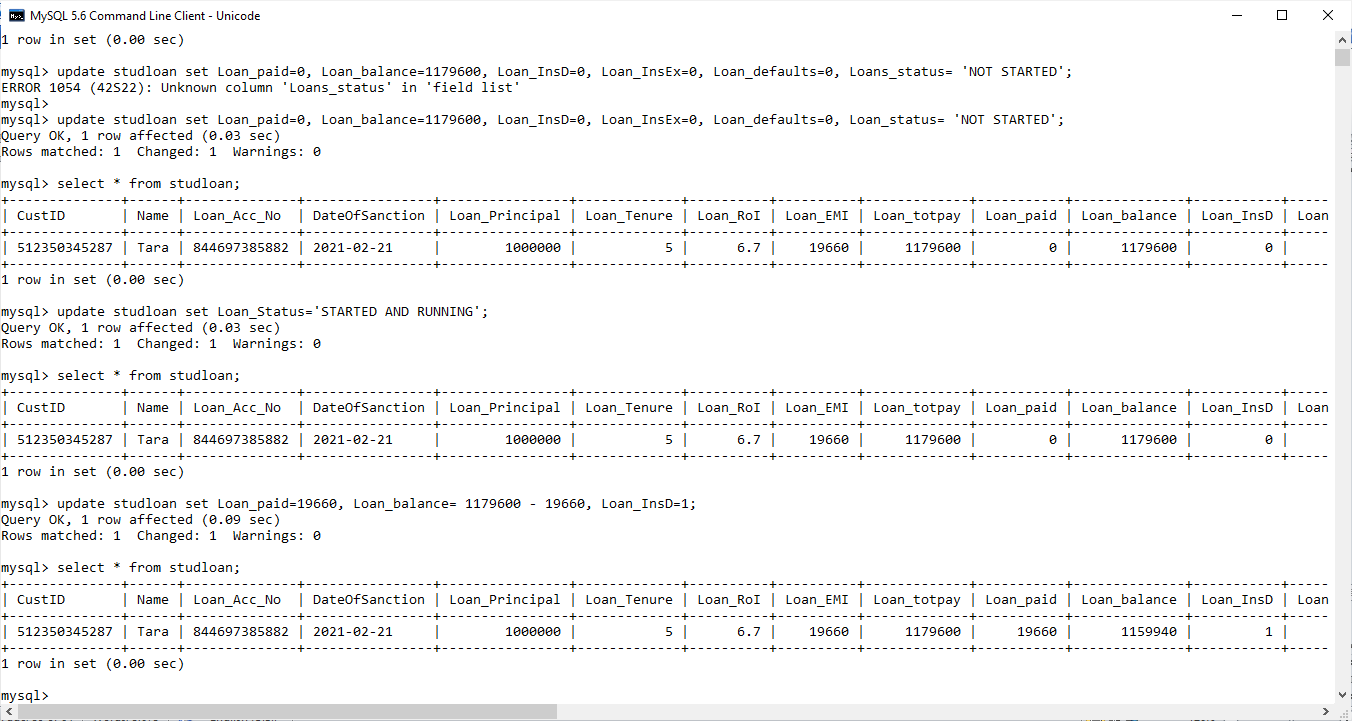


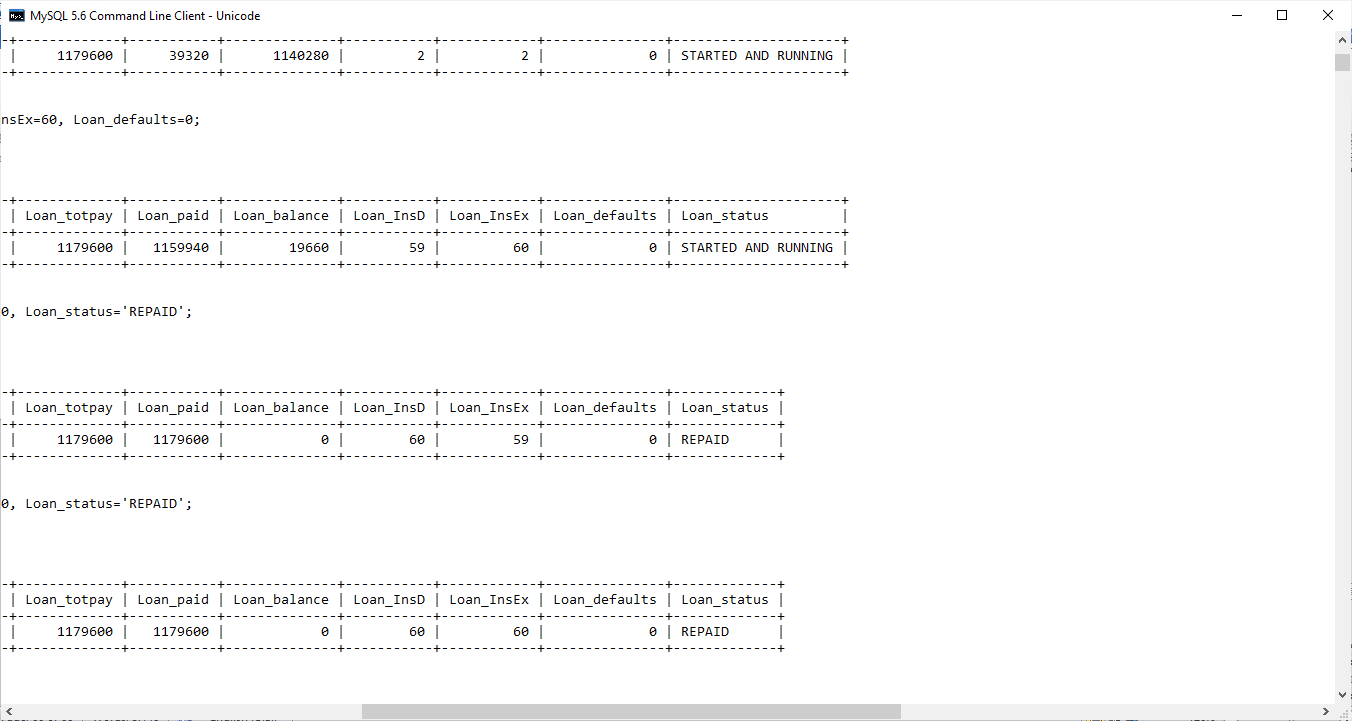
1. On paying all the upcoming installments without any defaults:





1. On Paying final Installment in February 2027:





After paying final installment:

Paid amount = Total amount to be paid,

Remaining Balance=0,

Number of Installments paid and expected till February 2027 = 60 (12 installments per year x 5 years)

Loan Status is ‘REPAID’

# **Future Improvements**

* Introducing Credit Cards
* Sponsoring Personal Loans
* Auto Loans
* Home Loans

# **Bibliography**

Computer Science with Python – Class XII, By Sumita Arora

<https://www.youtube.com/watch?v=Xt6SqWuMSA8>

[www.geeksforgeeks.org](http://www.geeksforgeeks.org http://stackoverflow.com/)

[http://stackoverflow.com/](http://www.geeksforgeeks.org http://stackoverflow.com/)

[www.quora.com](http://www.quora.com)

emicalculator.net

[www.bankbazaar.com](http://www.bankbazaar.com)

[www.myloancare.in](http://www.myloancare.in )

**THANK YOU**

**&**

**HAPPY BANKING!**