Introduction

The **Bank Management System** is a Python- and MySQL-based console application designed to simplify and automate essential banking operations like account management, transactions, and loan processing. It enables both administrators and users to manage banking activities efficiently through an interactive menu-driven interface.

This project integrates **Python** (for logic and user interface) with **MySQL** (for persistent data storage), creating a robust and secure environment for financial data management. It focuses on user experience, security, and modularity — using separate modules for admin functions, user operations, transactions, feedback, and updates.

The system ensures accurate handling of financial data while reducing manual errors and paperwork. It can easily be extended to include advanced features like online access, encryption, and audit tracking. Designed as a learning-oriented project, it demonstrates how modern banking workflows can be efficiently managed through **Python programming and database connectivity**.

Evolution of Management Systems using Python and MySQL Connector

Earlier, bank operations relied heavily on **manual record-keeping** and paper-based systems, which led to inefficiencies and data loss risks. With technological evolution, **Database Management Systems (DBMS)** and **Object-Oriented Programming (OOP)** transformed how banks operate.

Using **Python**, a high-level programming language known for its readability and database integration capabilities, and **MySQL Connector**, which allows seamless communication between Python and relational databases, modern systems now offer automation, reliability, and scalability.

Over time, these management systems have evolved from simple command-line tools to advanced, integrated applications that can handle millions of records securely. Python's flexibility and MySQL's structured storage make them a perfect combination for building real-time, data-driven management systems. This project represents this evolution — combining Python scripts to manage logic and **MySQL tables** to ensure secure data storage for customers, transactions, and loans. It models a real-world digital banking environment.

Purpose of the Bank Management System

The primary purpose of this system is to:

- **Digitize** core banking processes like account creation, updates, and deletion.
- **Automate** transactions such as deposits, withdrawals, and transfers with balance validation.
- **Provide loan management**, including loan application, approval, and EMI calculations.
- **Enable admins** to oversee all accounts, feedback, and loans with control over data.
- **Enhance user interaction** by offering a feedback system and support desk simulation.
- **Ensure data security and integrity** through structured database management and restricted admin access.
- **Save time and reduce human error** by minimizing repetitive manual entries and automating calculations.
- **Provide a scalable foundation** for developing more advanced online banking applications in the future.

Overall, it bridges the gap between user accessibility and administrative control through a simple yet powerful software tool.

User and Admin Functions

Module: `admin.py`

1. **`auth()`** – Authenticates admin credentials by matching username and password in the database; grants access to admin dashboard upon success.
2. **`addData()`** – Adds a new account holder with unique account number, balance, and personal details into the **`accHolder`** table.
3. **`viewAll()`** – Displays all account holders' details from the database in a formatted report.
4. **`newAdmin()`** – Allows creation of new admin accounts after checking for duplicate usernames.
5. **`approveLoan()`** – Lets admins approve pending loans by updating their status to 'ACTIVE' in the Loans table.
6. **`deleteAccount()`** – Permanently removes an account from the system after admin confirmation.

Module: `dbConnect.py`

1. **`connect()`** – Establishes a connection to the MySQL **`banking_system`** database with proper credentials; returns a connection object or error message.

Module: `feedback.py`

1. **`new()`** – Allows users to submit textual feedback linked to their account number, stored in the feedback table.
2. **`view()`** – Displays all feedback entries for admin review, showing account numbers and comments.

Module: `loan.py`

1. **`applyLoan()`** – Accepts loan applications by calculating EMI based on amount and tenure; inserts data into Loans table with ‘PENDING’ status.
2. **`viewLoan()`** – Fetches and displays all loans for a particular account, including status, tenure, and EMI details.
3. **`checkStatus()`** – Displays all loan IDs and their current status for a given account number.
4. **`listDefaulters()`** – Lists all accounts with loans marked as 'DEFAULT', helping admins identify defaulters.

Module: `transactions.py`

1. **`balance()`** – Displays the current account balance of a user by fetching from the **`accHolder`** table.
2. **`user_transactions()`** – Handles deposit, withdrawal, and fund transfer operations between accounts with live balance updates.
3. **`user_transaction_history()`** – Fetches and prints all past transactions for a given account.
4. **`admin_view_transactions()`** – Provides admins with daily, monthly, or full transaction reports.
5. **`add_transaction()`** – Inserts new transaction records into the database with type, amount, and date.
6. **`get_transactions_by_account()`** – Retrieves all transactions for a specific account in reverse chronological order.
7. **`check_balance()`** – Returns the current balance of an account for internal verification during transactions.
8. **`update_balance()`** – Updates the account’s balance based on transaction type (deposit, withdrawal, transfer).

Module: `update.py`

1. **`name()`** – Updates the account holder's name in the `accHolder` table.
2. **`email()`** – Updates the registered email of a specific account.
3. **`phnum()`** – Updates the phone number of an account holder.
4. **`address()`** – Updates the permanent address for an account holder.

Module: `users.py`

1. **`viewData()`** – Displays complete details of a user's bank account including balance and loan status.
2. **`closeAccount()`** – Permanently closes a user's account by deleting related feedback, loan, and transaction records.

Module: `schema.sql (Database Schema)`

Defines tables for **`accHolder, admin_data, transactions, Loans, and feedback`**, including sample records for testing.
Ensures data integrity with **foreign key constraints** and **ON DELETE CASCADE** for automatic cleanup.

Tree Structure

```
banking-system/
├── main.py                 # Main entry point of the
application
├── README.md               # Project overview and
documentation
├── requirements.txt        # Python dependencies
├── python/                 # Core logic of the application
│   ├── admin.py            # Admin-related functionalities
│   ├── dbConnect.py        # MySQL connection handling
│   ├── feedback.py         # Feedback processing functions
│   ├── loan.py             # Loan management logic
│   ├── menu.py             # Interactive command-line menu
│   ├── transactions.py     # Deposit, withdrawal, and
transfer logic
│   ├── update.py           # User account info update
functions
│   └── user.py             # User-related functionalities
├── sql/                    # Database schema and setup
│   └── schema.sql          # SQL schema setup script
```

Syntax

dbConnect.py

```
import mysql.connector as con

# --- DATABASE CONNECTION FUNCTION ---
def connect():
    """Establishes and returns a database connection."""
    try:
        db = con.connect(
            host="localhost",
            user="root",
            password="mypass",
            database="banking_system"
        )
        return db
    except con.Error as err:
        print(f"Error connecting to database: {err}")
        print("Please ensure MySQL is running, the database
'bank_management_system' exists, and the credentials are
correct.")
        return None
```

menu.py

```
#making the main menu for banking-system

import update
#=======================================================
#                    MAIN MENU
#=======================================================

def mainMenu(): #mainmenu
    print('''
    ============================================================
==
                    BANK MANAGEMENT SYSTEM
    ============================================================
==
''')
    print('''
    ------------------MAIN MENU-------------------
```

```
    [1]     USER MANAGMENT
    [2]     ADMIN DASHBOARD
    ____________________________________________

    [3]     NEW FEEDBACK
    [4]     CUSTOMER SUPPORT / HELP DESK ☎
    [0]     EXIT
''')

    try:
        n = int(input("Choose a menu option: "))
        return n
    except ValueError:
        print("Invalid input, please enter a number.")
        return -1


def userMenu():     #usermenu
    print('''
    ------------------USER MENU--------------------
    [1]     ACCOUNT HOLDER DETAILS
    [2]     VIEW LOAN DETAILS
    [3]     BALANCE ENQUIRY
    [4]     UPDATE NAME
    [5]     UPDATE EMAIL
    [6]     UPDATE PHONE NUMBER
    [7]     UPDATE ADDRESS
    [8]     LOAN SERVICES (Apply / EMI / Check Status)
    [9]     CLOSE ACCOUNT
    [10]    TRANSACTIONS (Deposit, Withdraw, Transfer)
    [11]    TRANSACTION HISTROY

    ____________________________________________

    [13]    GIVE FEEDBACK
    [14]    CUSTOMER SUPPORT / HELP DESK
    [0]     EXIT
''')

    try:
        n = int(input("Choose a menu option: "))
        return n
    except ValueError:
        print("Invalid input, please enter a number.")
        return -1
```

```
def adminMenu():     #adminmenu
    print('''
    ------------------ADMIN MENU-------------------
    [1]     VIEW ALL ACCOUNT HOLDERS
    [2]     VIEW FEEDBACK
    [3]     ADD ACCOUNT HOLDER
    [4]     NEW ADMIN ACCOUNT
    [5]     APPROVE LOAN
    [6]     LOAN DEFAULTERS
    [7]     UPDATE ACCOUNT DETAILS
    [8]     DELETE ACCOUNT (Permanent)
    ____________________________________________

    [0]     EXIT
''')

    try:
        n = int(input("Choose a menu option: "))
        return n
    except ValueError:
        print("Invalid input, please enter a number.")
        return -1

def upgd():
    print('''
    ------------------UPDATE MENU-------------------
    [1]     UPDATE NAME
    [2]     UPDATE EMAIL
    [3]     UPDATE PHONE NUMBER
    [4]     UPDATE ADDRESS
    ''')
    try:
        n = int(input("Choose a menu option: "))
    except ValueError:
        print("Invalid input, please enter a number.")

    match n:
        case 1:
            update.name()
        case 2:
            update.email()
        case 3:
            update.phnum()
        case 4:
            update.address()
```

admin.py

```
import dbConnect as con
import menu
import random


def auth(): #when the admin username & password is verified
    """Authenticate admin using username and password."""
    username = input("Enter admin username: ")
    password = input("Enter admin password: ")

    db = con.connect()
    if not db:
        return False

    cursor = db.cursor()
    try:
        query = "SELECT * FROM admin_data WHERE username = %s
AND password = %s"
        cursor.execute(query, (username, password))
        result = cursor.fetchone()

        if result:
            print("\n Admin Login Successful!\n")
            return True
        else:
            print("\n Invalid credentials. Access Denied.\n")
            return False

    except con.mysql.connector.Error as err:
        print(f"Database error: {err}")
        return False
    finally:
        db.close()

def addData():
    """Adds a new account holder to the system."""
    db = con.connect()
    if not db: return
    cu = db.cursor()
    try:
        acct_no = random.randint(1000000000, 9999999999)
        name = input("Enter the name of account holder: ")
        phone = input("Enter the phone number: ")
```

```
        email = input("Enter the email of account holder: ")
        address = input("Enter the address of account holder:
")
        balance = float(input("Enter the initial balance: "))
        loan_taken = input("Whether loan taken (yes/no): ")

        query = "INSERT INTO accHolder VALUES (%s, %s, %s,
%s, %s, %s, %s)"
        cu.execute(query, (acct_no, name, phone, email,
address, balance, loan_taken))
        db.commit()
        print(f"Successfully new account was added with A/C
No: {acct_no}")
    except ValueError:
        print("Invalid input for balance.")
    except con.Error as err:
        print(f"Database error: {err}")
    finally:
        db.close()

def viewAll():
    db = con.connect()
    if not db: return
    cu = db.cursor()
    try:
        query = "SELECT acct_no, holder_name, phone_no,
email, address, initial_balance, loan_taken FROM accHolder"
        cu.execute(query)
        data = cu.fetchall()
        if data:
            print("\n*************** ALL ACCOUNTS
***************")
            for row in data:
                print(f"Account Number: {row[0]}")
                print(f"Name: {row[1]}")
                print(f"Phone: {row[2]}")
                print(f"Email: {row[3]}")
                print(f"Address: {row[4]}")
                print(f"Balance: {row[5]}")
                print(f"Loan Taken: {row[6]}")
                print("-----------------------------------
---")
            print("*****************************************
**\n")
        else:
```

```
            print("No accounts found.")
    except con.Error as err:
        print(f"Database error: {err}")
    finally:
        db.close()

def newAdmin():
    """Allows creation of a new admin account."""
    username = input("Enter new admin username: ")
    password = input("Enter new admin password: ")

    db = con.connect()
    if not db:
        print("Database connection failed.")
        return

    cu = db.cursor()
    try:
        # Check if username already exists
        cu.execute("SELECT * FROM admin_data WHERE username =
%s", (username,))
        if cu.fetchone():
            print(" Username already exists. Choose a
different username.")
            return

        # Insert new admin into database
        query = "INSERT INTO admin_data (username, password)
VALUES (%s, %s)"
        cu.execute(query, (username, password))
        db.commit()

        print(f"\n New admin account created successfully
with username: {username}\n")

    except con.Error as err:
        print(f"Database error: {err}")
    finally:
        db.close()

def approveLoan():
    """For admins to approve pending loans"""
    db = con.connect()
    if not db: return
    cu = db.cursor()
```

```
    try:
        loan_id = int(input("Enter Loan ID to approve: "))
        query = "UPDATE Loans SET status = 'ACTIVE' WHERE
loan_id = %s AND status = 'PENDING'"
        cu.execute(query, (loan_id,))
        db.commit()

        if cu.rowcount > 0:
            print(f"Loan {loan_id} approved successfully ")
        else:
            print("No pending loan found with that ID ")

    except ValueError:
        print("Invalid input for loan ID.")
    except con.Error as err:
        print(f"Database error: {err}")
    finally:
        db.close()

def deleteAccount():
    #permanent delete account
    db = con.connect()
    if not db:
        return
    cu = db.cursor()
    try:
        acct_no = int(input("Enter the account number to
delete: "))
        confirm = input(f"Are you sure you want to
permanently delete account {acct_no}? (y/N):
").strip().lower()
        if confirm != "y":
            print("Deletion cancelled.")
            return

        query = "DELETE FROM accHolder WHERE acct_no = %s"
        cu.execute(query, (acct_no,))
        db.commit()

        if cu.rowcount > 0:
            print(f"Account {acct_no} permanently deleted by
admin.")
        else:
            print("No account found with that number.")
    except ValueError:
```

```
        print("Invalid input. Please enter a numeric account
number.")
    except con.Error as err:
        print(f"Database error: {err}")
    finally:
        db.close()
```

users.py

```
import dbConnect as con
import menu
import mysql.connector

def viewData():
    """Displays account holder's data based on account
number."""
    db = con.connect()
    if not db: return
    cu = db.cursor()
    try:
        number = int(input("Enter the account number: "))
        query = "SELECT * FROM accHolder WHERE acct_no =
{}".format(number)
        cu.execute(query)
        data = cu.fetchone()  # Fetch one record
        if data:
            print("\n*************** ACCOUNT DETAILS
***************")
            print("Account number: ", data[0])
            print("Name of account holder: ", data[1])
            print("Phone number: ", data[2])
            print("Email: ", data[3])
            print("Address: ", data[4])
            print("Initial balance: ", data[5])
            print("Loan Taken: ", data[6])
            print("****************************************
***\n")
        else:
            print("Account not found.")
    except ValueError:
        print("Invalid input. Please enter a numeric account
number.")
    except con.Error as err:
```

```
        print(f"Database error: {err}")
    finally:
        db.close()

def closeAccount():
    db = con.connect()
    if not db:
        return
    cu = db.cursor()
    try:
        acct_no = int(input("Enter your account number: "))
        confirm = input("Are you sure you want to close your
account? (Y/N): ").strip().lower()
        if confirm != "y":
            print("Account closure cancelled.")
            return

        # Delete all feedback linked to this account
        cu.execute("DELETE FROM feedback WHERE acct_no=%s",
(acct_no,))

        cu.execute("DELETE FROM loan_acct WHERE acct_no=%s",
(acct_no,))
        cu.execute("DELETE FROM transaction WHERE
acct_no=%s", (acct_no,))
        db.commit()


        # Delete from accHolder table
        query = "DELETE FROM accHolder WHERE acct_no = %s"
        cu.execute(query, (acct_no,))
        db.commit()

        if cu.rowcount > 0:
            print(f"Account {acct_no} closed successfully.")
        else:
            print("No account found with that number.")
    except ValueError:
        print("Invalid input. Please enter a numeric account
number.")
    except mysql.connector.Error as err:
        print(f"Database error: {err}")
    finally:
        db.close()
```

update.py

```
import dbConnect as con
import menu

db=con.connect()
def name():
    """Updates the name of an account holder."""
    if not db: return
    cu = db.cursor()
    try:
        number = int(input("Enter the account number: "))
        holder = input("Enter the updated name: ")
        query = "UPDATE accHolder SET holder_name = '{}'
WHERE acct_no = {}".format(holder, number)
        cu.execute(query)
        db.commit()
        if cu.rowcount > 0:
            print("Name successfully updated!")
        else:
            print("Account not found or name is the same.")
    except ValueError:
        print("Invalid input for account number.")
    except con.Error as err:
        print(f"Database error: {err}")
    finally:
        db.close()

def email():
    """Updates the email of an account holder."""
    if not db: return
    cu = db.cursor()
    try:
        number = int(input("Enter the account number: "))
        email = input("Enter the updated email: ")
        query = "UPDATE accHolder SET email = '{}' WHERE
acct_no = {}".format(email, number)
        cu.execute(query)
        db.commit()
        if cu.rowcount > 0:
            print("Email successfully updated!")
        else:
            print("Account not found or email is the same.")
    except ValueError:
        print("Invalid input for account number.")
```

```
    except con.Error as err:
        print(f"Database error: {err}")
    finally:
        db.close()

def phnum():
    """Updates the phone number of an account holder."""
    if not db: return
    cu = db.cursor()
    try:
        number = int(input("Enter the account number: "))
        phone = input("Enter the updated phone number: ")
        query = "UPDATE accHolder SET phone_no = '{}' WHERE
acct_no = {}".format(phone, number)
        cu.execute(query)
        db.commit()
        if cu.rowcount > 0:
            print("Phone number successfully updated!")
        else:
            print("Account not found or phone number is the
same.")
    except ValueError:
        print("Invalid input for account number.")
    except con.Error as err:
        print(f"Database error: {err}")
    finally:
        db.close()

def address():
    """Updates the address of an account holder."""
    if not db: return
    cu = db.cursor()
    try:
        number = int(input("Enter the account number: "))
        address = input("Enter the updated address: ")
        query = "UPDATE accHolder SET address = '{}' WHERE
acct_no = {}".format(address, number)
        cu.execute(query)
        db.commit()
        if cu.rowcount > 0:
            print("Address was updated successfully!")
        else:
            print("Account not found or address is the
same.")
    except ValueError:
```

```
        print("Invalid input for account number.")
    except con.Error as err:
        print(f"Database error: {err}")
    finally:
        db.close()
```

transactions.py

```
import dbConnect as con
import menu
from datetime import date

# ========== BALANCE ENQUIRY ==========

def balance():
    """Gives the current balance in the account"""
    db = con.connect()
    if not db:
        return
    cu = db.cursor()
    try:
        acct_no = int(input("Enter the account number: "))
        query = "SELECT acct_no, holder_name, initial_balance
FROM accHolder WHERE acct_no = %s"
        cu.execute(query, (acct_no,))
        data = cu.fetchone()
        if data:
            print("\n********** BALANCE ENQUIRY
**********")
            print("Account number: ", data[0])
            print("Name of account holder: ", data[1])
            print("Balance: ", data[2])
            print("**************************************
***\n")
        else:
            print("Account not found.")
    except ValueError:
        print("Invalid input. Please enter a numeric account
number.")
    except con.Error as err:
        print(f"Database error: {err}")
    finally:
```

```
        db.close()


# ========== USER MENU FUNCTIONS ==========

# [6] TRANSACTIONS (Deposit, Withdraw, Transfer)
def user_transactions():
    acct_no = int(input("Enter your account number: "))

    print("\n===== TRANSACTIONS =====")
    print("1. Deposit")
    print("2. Withdraw")
    print("3. Transfer")
    choice = int(input("Enter choice: "))

    if choice == 1:  # Deposit
        amt = float(input("Enter amount to deposit: "))
        add_transaction(acct_no, "DEPOSIT", amt)
        update_balance(acct_no, amt, "DEPOSIT")

    elif choice == 2:  # Withdraw
        amt = float(input("Enter amount to withdraw: "))
        if check_balance(acct_no) >= amt:
            add_transaction(acct_no, "WITHDRAW", amt)
            update_balance(acct_no, amt, "WITHDRAW")
        else:
            print("Insufficient Balance!")

    elif choice == 3:  # Transfer
        to_acct = int(input("Enter recipient account no: "))
        amt = float(input("Enter amount to transfer: "))
        if check_balance(acct_no) >= amt:
            add_transaction(acct_no, "TRANSFER_OUT", amt)
            update_balance(acct_no, amt, "WITHDRAW")
            add_transaction(to_acct, "TRANSFER_IN", amt)
            update_balance(to_acct, amt, "DEPOSIT")
            print(f"Transferred {amt} from {acct_no} to
{to_acct}")
        else:
            print("Insufficient Balance!")
    else:
        print("Invalid Choice!")


# [7] TRANSACTION HISTORY
def user_transaction_history():
```

```
    acct_no = int(input("Enter your account number: "))
    get_transactions_by_account(acct_no)


# ========== ADMIN MENU FUNCTIONS ==========

# [3] VIEW ALL TRANSACTIONS (Daily / Monthly Reports)
def admin_view_transactions():
    print("\n===== TRANSACTION REPORTS =====")
    print("1. Daily Report")
    print("2. Monthly Report")
    print("3. Full Report")
    choice = int(input("Enter choice: "))

    db = con.connect()
    if db is None:
        return
    cursor = db.cursor()

    if choice == 1:
        today = date.today()
        query = "SELECT * FROM transaction WHERE
transaction_date = %s"
        cursor.execute(query, (today,))
        print(f"\n Daily Report for {today}:")
    elif choice == 2:
        month = int(input("Enter month (1-12): "))
        year = int(input("Enter year (YYYY): "))
        query = """SELECT * FROM transaction
                   WHERE MONTH(transaction_date) = %s AND
YEAR(transaction_date) = %s"""
        cursor.execute(query, (month, year))
        print(f"\n Monthly Report for {month}/{year}:")
    else:
        query = "SELECT * FROM transaction ORDER BY
transaction_date DESC"
        cursor.execute(query)
        print("\n Full Report:")

    records = cursor.fetchall()
    for row in records:
        print(f"ID: {row[0]}, Acc: {row[1]}, Type: {row[2]},
Amt: {row[3]}, Date: {row[4]}")

    cursor.close()
    db.close()
```

```python
# ========== HELPER FUNCTIONS (all take input inside)
==========

def add_transaction(acct_no=None, transaction_type=None,
amount=None):
    if acct_no is None:
        acct_no = int(input("Enter account number: "))
    if transaction_type is None:
        transaction_type = input("Enter transaction type:
").upper()
    if amount is None:
        amount = float(input("Enter amount: "))

    db = con.connect()
    if db is None:
        return
    cursor = db.cursor()

    query = """
        INSERT INTO transactions (acct_no, transaction_type,
amount, transaction_date)
        VALUES (%s, %s, %s, %s)
    """
    values = (acct_no, transaction_type, amount,
date.today())

    cursor.execute(query, values)
    db.commit()
    cursor.close()
    db.close()


def get_transactions_by_account(acct_no=None):
    if acct_no is None:
        acct_no = int(input("Enter account number: "))

    db = con.connect()
    if db is None:
        return
    cursor = db.cursor()

    query = "SELECT * FROM transactions WHERE acct_no = %s
ORDER BY transaction_date DESC"
    cursor.execute(query, (acct_no,))
```

```
    records = cursor.fetchall()

    print(f"\n Transactions for Account {acct_no}:")
    for row in records:
        print(f"ID: {row[0]}, Type: {row[2]}, Amount:
{row[3]}, Date: {row[4]}")

    cursor.close()
    db.close()


def check_balance(acct_no=None):
    if acct_no is None:
        acct_no = int(input("Enter account number: "))

    db = con.connect()
    if db is None:
        return 0
    cursor = db.cursor()

    query = "SELECT initial_balance FROM accHolder WHERE
acct_no = %s"  # FIXED
    cursor.execute(query, (acct_no,))
    result = cursor.fetchone()
    cursor.close()
    db.close()

    return result[0] if result else 0


def update_balance(acct_no=None, amount=None,
transaction_type=None):
    if acct_no is None:
        acct_no = int(input("Enter account number: "))
    if amount is None:
        amount = float(input("Enter amount: "))
    if transaction_type is None:
        transaction_type = input("Enter type
(DEPOSIT/WITHDRAW/TRANSFER_IN/TRANSFER_OUT): ").upper()

    db = con.connect()
    if db is None:
        return
    cursor = db.cursor()

    if transaction_type in ("DEPOSIT", "TRANSFER_IN"):
```

```
        query = "UPDATE accHolder SET initial_balance =
initial_balance + %s WHERE acct_no = %s"
    else:  # WITHDRAW, TRANSFER_OUT
        query = "UPDATE accHolder SET initial_balance =
initial_balance - %s WHERE acct_no = %s"

    cursor.execute(query, (amount, acct_no))
    db.commit()
    cursor.close()
    db.close()
```

loan.py

```
import dbConnect as con

def applyLoan(acct_no, loan_amount, tenure_months):
    """Apply for a new loan"""
    db = con.connect()
    if not db: return
    cu = db.cursor()


    rate = 0.075 / 12   # monthly 7.5% annual interest
    n = tenure_months
    P = loan_amount
    emi = (P*rate*(1+rate)**n) / ((1+rate)**n - 1)

    query = "INSERT INTO Loans (acct_no, loan_amount,
tenure_months, emi_amount) VALUES (%s,%s,%s,%s)"
    cu.execute(query, (acct_no, loan_amount, tenure_months,
round(emi,2)))
    db.commit()
    print(f"Loan request submitted. EMI = {emi:.2f}")
    db.close()

def viewLoan(acct_no=None):
    """View all loans for a user"""
    if acct_no is None:
        try:
            acct_no = int(input("Enter account number: "))
        except ValueError:
```

```
            print("Invalid input. Please enter a numeric
account number.")
            return
    db = con.connect()
    if not db: return
    cu = db.cursor()
    try:
        cu.execute("SELECT loan_id, acct_no, loan_amount,
interest_rate, tenure_months, emi_amount, status, issued_date
FROM Loans WHERE acct_no = %s", (acct_no,))
        loans = cu.fetchall()
        if loans:
            print("\n*************** LOAN DETAILS
***************")
            for loan in loans:
                print(f"Loan ID: {loan[0]}")
                print(f"Account Number: {loan[1]}")
                print(f"Loan Amount: {float(loan[2]):.2f}")
                print(f"Interest Rate: 7.50%")  # Fixed rate
as per applyLoan()
                print(f"Tenure (Months): {loan[3]}")
                print(f"EMI Amount: {float(loan[4]):.2f}")
                print(f"Status: {loan[5]}")
                print(f"Application Date: {loan[6]}")
                print("-----------------------------------
---")
            print("****************************************
**\n")
        else:
            print("No loans found for this account.")
    except con.errors.Error as err:
        print(f"Database error: {err}")
    finally:
        cu.close()
        db.close()

def checkStatus(acct_no):
    """Check loan status"""
    db = con.connect()
    cu = db.cursor()
    cu.execute("SELECT loan_id, status FROM Loans WHERE
acct_no = %s", (acct_no,))
    for l in cu.fetchall():
        print(f"Loan {l[0]}: {l[1]}")
    db.close()
```

```
def listDefaulters():
    """Admin: view loan defaulters"""
    db = con.connect()
    cu = db.cursor()
    cu.execute("""SELECT l.loan_id, a.holder_name,
l.loan_amount, l.status
                  FROM Loans l
                  JOIN accHolder a ON l.acct_no=a.acct_no
                  WHERE l.status='DEFAULT'""")
    for row in cu.fetchall():
        print(row)
    db.close()
```

feedback.py

```
import dbConnect as con
import mysql.connector
import menu
from mysql.connector import Error


def new():
    """Allows a user to give feedback."""

    db=con.connect()
    if not db: return
    cu = db.cursor()
    try:
        number = int(input("Enter your account number: "))
        feed = input("Enter your feedback: ")
        query = "INSERT INTO feedback (acct_no,
feedback_text) VALUES ({}, '{}')".format(number, feed)
        cu.execute(query)
        db.commit()
        print("Thank you for your feedback!")
    except ValueError:
        print("Invalid input for account number.")
    except mysql.connector.Error as err:
        print(f"Database error: Account Number may not
exist!!")
    finally:
        db.close()
```

```
def view():
    """Displays all user feedback."""
    db=con.connect()
    if not db: return
    cu = db.cursor()
    try:
        query = "SELECT * FROM feedback"
        cu.execute(query)
        data = cu.fetchall()
        if data:
            print("\n*************** USER FEEDBACKS
***************")
            for row in data:
                print(f"Account Number: {row[0]}")
                print(f"Feedback: {row[1]}")
                print("-----------------------------------
---")
            print("*****************************************
****\n")
        else:
            print("No feedback found.")
    except con.Error as err:
        print(f"Database error: {err}")
    finally:
        db.close()
```

schema.sql

```sql
CREATE DATABASE if NOT EXISTS banking_system;

USE banking_system;


CREATE TABLE IF NOT EXISTS accHolder (
  acct_no BIGINT PRIMARY KEY,
  holder_name VARCHAR(50) NOT NULL,
  phone_no VARCHAR(15),
  email VARCHAR(50),
  address VARCHAR(100),
  initial_balance DECIMAL(15, 2),
  loan_taken VARCHAR(5)
);

-- Insert some sample account holders
INSERT INTO accHolder (acct_no, holder_name, phone_no, email,
address, initial_balance, loan_taken) VALUES
(1234567890, 'John Doe', '9876543210', 'john.doe@email.com',
'123 River Road', 50000.00, "YES"),
(1234567891, 'Jane Smith', '9876543211',
'jane.smith@email.com', '456 Oak Avenue', 75000.00, "NO"),
(1234567892, 'Peter Jones', '9876543212',
'peter.jones@email.com', '789 Pine Lane', 120000.00, "YES")
ON DUPLICATE KEY UPDATE holder_name=VALUES(holder_name);

CREATE TABLE IF NOT EXISTS feedback (
  acct_no BIGINT,
  feedback_text varchar(5000),
  FOREIGN KEY (acct_no) REFERENCES accHolder(acct_no)
  );
--@block


--@block
INSERT INTO accHolder (acct_no, holder_name, phone_no, email,
address, initial_balance, loan_taken) VALUES
(1234567893, 'Alice Brown', '9876543213',
'alice.brown@email.com', '12 Maple Street', 45000.00, 'NO'),
(1234567894, 'David Wilson', '9876543214',
'david.wilson@email.com', '34 Elm Avenue', 62000.00, 'YES'),
(1234567895, 'Sophia Taylor', '9876543215',
'sophia.taylor@email.com', '56 Cedar Road', 83000.00, 'NO'),
```

```
(1234567896, 'Michael Johnson', '9876543216',
'michael.johnson@email.com', '78 Birch Lane', 100000.00,
'YES'),
(1234567897, 'Emily Davis', '9876543217',
'emily.davis@email.com', '90 Willow Court', 27000.00, 'NO'),
(1234567898, 'Daniel Martinez', '9876543218',
'daniel.martinez@email.com', '11 Chestnut Blvd', 94000.00,
'YES'),
(1234567899, 'Olivia Anderson', '9876543219',
'olivia.anderson@email.com', '22 Poplar Street', 53000.00,
'NO'),
(1234567900, 'James Thomas', '9876543220',
'james.thomas@email.com', '33 Ash Avenue', 12000.00, 'NO'),
(1234567901, 'Isabella Moore', '9876543221',
'isabella.moore@email.com', '44 Cypress Lane', 74000.00,
'YES'),
(1234567902, 'William Harris', '9876543222',
'william.harris@email.com', '55 Redwood Drive', 66000.00,
'NO'),
(1234567903, 'Mia Clark', '9876543223',
'mia.clark@email.com', '66 Spruce Road', 89000.00, 'YES'),
(1234567904, 'Ethan Lewis', '9876543224',
'ethan.lewis@email.com', '77 Palm Street', 31000.00, 'NO')
ON DUPLICATE KEY UPDATE holder_name=VALUES(holder_name);

--@block

-- Create admin_data table
CREATE TABLE IF NOT EXISTS admin_data (
  username VARCHAR(30) PRIMARY KEY,
  password VARCHAR(30) NOT NULL
);

-- Insert admin accounts (with ON DUPLICATE KEY to avoid
duplicates)
INSERT INTO admin_data (username, password) VALUES
('admin', 'password123'),
('manager', 'securePass!'),
('officer', 'bank@321')
ON DUPLICATE KEY UPDATE password = VALUES(password);

--@block
-- Create transaction table
CREATE TABLE IF NOT EXISTS transactions (
  transaction_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
  acct_no BIGINT,
  transaction_type VARCHAR(20) NOT NULL,
  amount DECIMAL(15, 2) NOT NULL,
  transaction_date DATE DEFAULT (CURRENT_DATE),
  FOREIGN KEY (acct_no) REFERENCES accHolder(acct_no) ON
DELETE CASCADE
);


INSERT INTO transactions (acct_no, transaction_type, amount,
transaction_date) VALUES
(1234567890, 'deposit', 5000.00, '2025-09-01'),
(1234567890, 'withdrawal', 1500.00, '2025-09-02'),
(1234567891, 'deposit', 10000.00, '2025-09-02'),
(1234567891, 'withdrawal', 2000.00, '2025-09-03'),
(1234567892, 'deposit', 25000.00, '2025-09-04'),
(1234567892, 'withdrawal', 5000.00, '2025-09-05'),
(1234567890, 'withdrawal', 1000.00, '2025-09-06'),
(1234567891, 'deposit', 3000.00, '2025-09-06'),
(1234567892, 'deposit', 7000.00, '2025-09-07'),
(1234567890, 'deposit', 12000.00, '2025-09-08');

--@block
CREATE TABLE Loans (
    loan_id INT PRIMARY KEY AUTO_INCREMENT,
    acct_no BIGINT NOT NULL,
    loan_amount DECIMAL(12,2) NOT NULL,
    interest_rate DECIMAL(5,2) NOT NULL DEFAULT 7.5,
    tenure_months INT NOT NULL,
    emi_amount DECIMAL(12,2),
    status ENUM('PENDING','ACTIVE','CLOSED','DEFAULT')
DEFAULT 'PENDING',
    issued_date DATE DEFAULT (CURRENT_DATE),
    FOREIGN KEY (acct_no) REFERENCES accHolder(acct_no) ON
DELETE CASCADE
);

INSERT INTO Loans (acct_no, loan_amount, interest_rate,
tenure_months, emi_amount, status, issued_date)
VALUES
(1234567890, 200000, 7.5, 24, 8996.97, 'ACTIVE', '2024-05-
15'),
(1234567891, 500000, 8.0, 60, 10134.20, 'ACTIVE', '2023-11-
20'),
(1234567892, 150000, 7.5, 12, 12964.71, 'PENDING', '2025-01-
10'),
```

```
(1234567893, 300000, 9.0, 36, 9560.58, 'ACTIVE', '2024-03-
05'),
(1234567894, 100000, 7.5, 24, 4493.98, 'CLOSED', '2022-07-
18'),
(1234567895, 75000, 8.5, 18, 4568.29, 'ACTIVE', '2023-06-
21'),
(1234567896, 120000, 7.5, 36, 3727.37, 'DEFAULT', '2022-12-
30'),
(1234567897, 250000, 9.0, 48, 6269.57, 'ACTIVE', '2024-08-
11'),
(1234567898, 50000, 7.0, 12, 4322.58, 'CLOSED', '2021-11-
01'),
(1234567899, 400000, 8.5, 60, 8213.47, 'ACTIVE', '2024-01-
29'),
(1234567900, 175000, 7.5, 24, 7874.34, 'DEFAULT', '2023-09-
10'),
(1234567901, 220000, 8.0, 36, 6889.63, 'ACTIVE', '2025-02-
25'),
(1234567902, 95000, 7.5, 18, 5561.65, 'PENDING', '2025-03-
14'),
(1234567903, 600000, 9.5, 72, 11003.65, 'ACTIVE', '2022-10-
17'),
(1234567904, 130000, 7.0, 24, 5811.56, 'CLOSED', '2021-05-
19');

--@block
SELECT * FROM loans
```

Databases

```
mysql> use banking_system;
Database changed
mysql> show tables;
+---------------------------+
| Tables_in_banking_system  |
+---------------------------+
| accholder                 |
| admin_data                |
| feedback                  |
| loan_acct                 |
| loans                     |
| transaction               |
+---------------------------+
6 rows in set (0.02 sec)

mysql>
```

Data in account holder table:

```
mysql> select * from accholder;
+-----------+---------------+-----------+--------------------------+-----------------+-----------------+------------+
| acct_no   | holder_name   | phone_no  | email                    | address         | initial_balance | loan_taken |
+-----------+---------------+-----------+--------------------------+-----------------+-----------------+------------+
| 1234567890 | John Doe      | 9876543210 | john.doe@email.com       | 123 River Road  |        50000.00 | YES        |
| 1234567891 | Jane Smith    | 9876543211 | jane.smith@email.com     | 456 Oak Avenue  |        75000.00 | NO         |
| 1234567892 | Peter Jones   | 9876543212 | sh@gmail.com             | 789 Pine Lane   |       126699.00 | YES        |
| 1234567893 | Alice Brown   | 9876543213 | alice.brown@email.com    | 12 Maple Street |        45000.00 | NO         |
| 1234567894 | David Wilson  | 9876543214 | david.wilson@email.com   | 34 Elm Avenue   |        62000.00 | YES        |
| 1234567895 | Sophia Taylor | 9876543215 | sophia.taylor@email.com  | 56 Cedar Road   |        83000.00 | NO         |
| 1234567896 | Michael Johnson | 9876543216 | michael.johnson@email.com | 78 Birch Lane   |       100000.00 | YES        |
| 1234567897 | Emily Davis   | 9876543217 | emily.davis@email.com    | 90 Willow Court |        27000.00 | NO         |
| 1234567898 | Daniel Martinez | 9876543218 | daniel.martinez@email.com | 11 Chestnut Blvd |        94000.00 | YES        |
| 1234567899 | Olivia Anderson | 9876543219 | olivia.anderson@email.com | 22 Poplar Street |        53000.00 | NO         |
| 1234567900 | James Thomas  | 9876543220 | james.thomas@email.com   | 33 Ash Avenue   |        12000.00 | NO         |
| 1234567901 | Isabella Moore | 9876543221 | isabella.moore@email.com | 44 Cypress Lane |        74000.00 | YES        |
| 1234567902 | William Harris | 9876543222 | william.harris@email.com | 55 Redwood Drive |        66000.00 | NO         |
| 1234567903 | Mia Clark     | 9876543223 | mia.clark@email.com      | 66 Spruce Road  |        89000.00 | YES        |
| 1234567904 | Ethan Lewis   | 9876543224 | ethan.lewis@email.com    | 77 Palm Street  |        31000.00 | NO         |
+-----------+---------------+-----------+--------------------------+-----------------+-----------------+------------+
15 rows in set (0.00 sec)

mysql>
```

Data in admin table:

```
mysql> select * from admin_data;
+----------+-------------+
| username | password    |
+----------+-------------+
| admin    | password123 |
| shazra   | mypass      |
+----------+-------------+
2 rows in set (0.04 sec)

mysql>
```

Describe ADMIN table:

```
mysql> desc admin_data;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| username | varchar(30) | NO   | PRI | NULL    |       |
| password | varchar(30) | NO   |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
2 rows in set (0.02 sec)

mysql>
```

Describe FEEDBACK table:

```
mysql> desc feedback;
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| acct_no       | bigint        | YES  | MUL | NULL    |       |
| feedback_text | varchar(5000) | YES  |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

Show feedback of users:

```
mysql> select * from feedback;
+------------+-----------------------------------------------------------------------------+
| acct_no    | feedback_text                                                               |
+------------+-----------------------------------------------------------------------------+
| 1234567890 | Very good system. I am certain the developers are of the greatest minds! |
+------------+-----------------------------------------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

Describe Transaction Table:

```
mysql> desc transaction;
+------------------+---------------+------+-----+---------+----------------+
| Field            | Type          | Null | Key | Default | Extra          |
+------------------+---------------+------+-----+---------+----------------+
| transaction_id   | int           | NO   | PRI | NULL    | auto_increment |
| acct_no          | bigint        | YES  | MUL | NULL    |                |
| transaction_type | varchar(20)   | YES  |     | NULL    |                |
| amount           | decimal(15,2) | YES  |     | NULL    |                |
| transaction_date | date          | YES  |     | NULL    |                |
+------------------+---------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql>
```

Show data in Transaction Table:

```
mysql> select * from transaction;
+----------------+------------+------------------+----------+------------------+
| transaction_id | acct_no    | transaction_type | amount   | transaction_date |
+----------------+------------+------------------+----------+------------------+
|              5 | 1234567892 | deposit          | 25000.00 | 2025-09-04       |
|              6 | 1234567892 | withdrawal       |  5000.00 | 2025-09-05       |
|              9 | 1234567892 | deposit          |  7000.00 | 2025-09-07       |
|             15 | 1234567892 | deposit          | 25000.00 | 2025-09-04       |
|             16 | 1234567892 | withdrawal       |  5000.00 | 2025-09-05       |
|             19 | 1234567892 | deposit          |  7000.00 | 2025-09-07       |
|             21 | 1234567892 | DEPOSIT          |  6969.00 | 2025-09-06       |
|             22 | 1234567892 | DEPOSIT          |  1000.00 | 2025-09-06       |
|             26 | 1234567892 | TRANSFER_IN      |  6699.00 | 2025-09-06       |
|             27 | 1234567890 | deposit          |  5000.00 | 2025-09-01       |
|             28 | 1234567890 | withdrawal       |  1500.00 | 2025-09-02       |
|             29 | 1234567891 | deposit          | 10000.00 | 2025-09-02       |
|             30 | 1234567891 | withdrawal       |  2000.00 | 2025-09-03       |
|             31 | 1234567892 | deposit          | 25000.00 | 2025-09-04       |
|             32 | 1234567892 | withdrawal       |  5000.00 | 2025-09-05       |
|             33 | 1234567890 | withdrawal       |  1000.00 | 2025-09-06       |
|             34 | 1234567891 | deposit          |  3000.00 | 2025-09-06       |
|             35 | 1234567892 | deposit          |  7000.00 | 2025-09-07       |
|             36 | 1234567890 | deposit          | 12000.00 | 2025-09-08       |
|             37 | 1234567890 | deposit          |  5000.00 | 2025-09-01       |
|             38 | 1234567890 | withdrawal       |  1500.00 | 2025-09-02       |
|             39 | 1234567891 | deposit          | 10000.00 | 2025-09-02       |
|             40 | 1234567891 | withdrawal       |  2000.00 | 2025-09-03       |
|             41 | 1234567892 | deposit          | 25000.00 | 2025-09-04       |
|             42 | 1234567892 | withdrawal       |  5000.00 | 2025-09-05       |
|             43 | 1234567890 | withdrawal       |  1000.00 | 2025-09-06       |
|             44 | 1234567891 | deposit          |  3000.00 | 2025-09-06       |
|             45 | 1234567892 | deposit          |  7000.00 | 2025-09-07       |
|             46 | 1234567890 | deposit          | 12000.00 | 2025-09-08       |
|             47 | 1234567890 | deposit          |  5000.00 | 2025-09-01       |
|             48 | 1234567890 | withdrawal       |  1500.00 | 2025-09-02       |
|             49 | 1234567891 | deposit          | 10000.00 | 2025-09-02       |
|             50 | 1234567891 | withdrawal       |  2000.00 | 2025-09-03       |
|             51 | 1234567892 | deposit          | 25000.00 | 2025-09-04       |
|             52 | 1234567892 | withdrawal       |  5000.00 | 2025-09-05       |
|             53 | 1234567890 | withdrawal       |  1000.00 | 2025-09-06       |
|             54 | 1234567891 | deposit          |  3000.00 | 2025-09-06       |
|             55 | 1234567892 | deposit          |  7000.00 | 2025-09-07       |
|             56 | 1234567890 | deposit          | 12000.00 | 2025-09-08       |
+----------------+------------+------------------+----------+------------------+
39 rows in set (0.00 sec)

mysql>
```

Describe account table:

```
mysql> desc accholder;
+-----------------+---------------+------+-----+---------+-------+
| Field           | Type          | Null | Key | Default | Extra |
+-----------------+---------------+------+-----+---------+-------+
| acct_no         | bigint        | NO   | PRI | NULL    |       |
| holder_name     | varchar(50)   | NO   |     | NULL    |       |
| phone_no        | varchar(15)   | YES  |     | NULL    |       |
| email           | varchar(50)   | YES  |     | NULL    |       |
| address         | varchar(100)  | YES  |     | NULL    |       |
| initial_balance | decimal(15,2) | YES  |     | NULL    |       |
| loan_taken      | varchar(5)    | YES  |     | NULL    |       |
+-----------------+---------------+------+-----+---------+-------+
7 rows in set (0.00 sec)

mysql>
```

Data in loans table:

```
mysql> select * from loans;
+---------+-----------+-------------+---------------+---------------+------------+---------+-------------+
| loan_id | acct_no   | loan_amount | interest_rate | tenure_months | emi_amount | status  | issued_date |
+---------+-----------+-------------+---------------+---------------+------------+---------+-------------+
|       1 | 1234567890 |   200000.00 |          7.50 |            24 |    8996.97 | ACTIVE  | 2024-05-15  |
|       2 | 1234567891 |   500000.00 |          8.00 |            60 |   10134.20 | ACTIVE  | 2023-11-20  |
|       3 | 1234567892 |   150000.00 |          7.50 |            12 |   12964.71 | PENDING | 2025-01-10  |
|       4 | 1234567893 |   300000.00 |          9.00 |            36 |    9560.58 | ACTIVE  | 2024-03-05  |
|       5 | 1234567894 |   100000.00 |          7.50 |            24 |    4493.98 | CLOSED  | 2022-07-18  |
|       6 | 1234567895 |    75000.00 |          8.50 |            18 |    4568.29 | ACTIVE  | 2023-06-21  |
|       7 | 1234567896 |   120000.00 |          7.50 |            36 |    3727.37 | DEFAULT | 2022-12-30  |
|       8 | 1234567897 |   250000.00 |          9.00 |            48 |    6269.57 | ACTIVE  | 2024-08-11  |
|       9 | 1234567898 |    50000.00 |          7.00 |            12 |    4322.58 | CLOSED  | 2021-11-01  |
|      10 | 1234567899 |   400000.00 |          8.50 |            60 |    8213.47 | ACTIVE  | 2024-01-29  |
|      11 | 1234567900 |   175000.00 |          7.50 |            24 |    7874.34 | DEFAULT | 2023-09-10  |
|      12 | 1234567901 |   220000.00 |          8.00 |            36 |    6889.63 | ACTIVE  | 2025-02-25  |
|      13 | 1234567902 |    95000.00 |          7.50 |            18 |    5561.65 | PENDING | 2025-03-14  |
|      14 | 1234567903 |   600000.00 |          9.50 |            72 |   11003.65 | ACTIVE  | 2022-10-17  |
|      15 | 1234567904 |   130000.00 |          7.00 |            24 |    5811.56 | CLOSED  | 2021-05-19  |
|      16 | 1234567890 |   200000.00 |          7.50 |            24 |    8996.97 | ACTIVE  | 2024-05-15  |
|      17 | 1234567891 |   500000.00 |          8.00 |            60 |   10134.20 | ACTIVE  | 2023-11-20  |
|      18 | 1234567892 |   150000.00 |          7.50 |            12 |   12964.71 | PENDING | 2025-01-10  |
|      19 | 1234567893 |   300000.00 |          9.00 |            36 |    9560.58 | ACTIVE  | 2024-03-05  |
|      20 | 1234567894 |   100000.00 |          7.50 |            24 |    4493.98 | CLOSED  | 2022-07-18  |
|      21 | 1234567895 |    75000.00 |          8.50 |            18 |    4568.29 | ACTIVE  | 2023-06-21  |
|      22 | 1234567896 |   120000.00 |          7.50 |            36 |    3727.37 | DEFAULT | 2022-12-30  |
|      23 | 1234567897 |   250000.00 |          9.00 |            48 |    6269.57 | ACTIVE  | 2024-08-11  |
|      24 | 1234567898 |    50000.00 |          7.00 |            12 |    4322.58 | CLOSED  | 2021-11-01  |
|      25 | 1234567899 |   400000.00 |          8.50 |            60 |    8213.47 | ACTIVE  | 2024-01-29  |
|      26 | 1234567900 |   175000.00 |          7.50 |            24 |    7874.34 | DEFAULT | 2023-09-10  |
|      27 | 1234567901 |   220000.00 |          8.00 |            36 |    6889.63 | ACTIVE  | 2025-02-25  |
|      28 | 1234567902 |    95000.00 |          7.50 |            18 |    5561.65 | PENDING | 2025-03-14  |
|      29 | 1234567903 |   600000.00 |          9.50 |            72 |   11003.65 | ACTIVE  | 2022-10-17  |
|      30 | 1234567904 |   130000.00 |          7.00 |            24 |    5811.56 | CLOSED  | 2021-05-19  |
|      31 | 1234567901 |    10000.00 |          7.50 |            12 |     867.57 | PENDING | 2025-09-13  |
+---------+-----------+-------------+---------------+---------------+------------+---------+-------------+
31 rows in set (0.00 sec)

mysql>
```

Describe loans table:

```
mysql> desc loans;
+--------------+-----------------------------------------+------+-----+-----------+------------------+
| Field        | Type                                    | Null | Key | Default   | Extra            |
+--------------+-----------------------------------------+------+-----+-----------+------------------+
| loan_id      | int                                     | NO   | PRI | NULL      | auto_increment   |
| acct_no      | bigint                                  | NO   | MUL | NULL      |                  |
| loan_amount  | decimal(12,2)                           | NO   |     | NULL      |                  |
| interest_rate| decimal(5,2)                            | NO   |     | 7.50      |                  |
| tenure_months| int                                     | NO   |     | NULL      |                  |
| emi_amount   | decimal(12,2)                           | YES  |     | NULL      |                  |
| status       | enum('PENDING','ACTIVE','CLOSED','DEFAULT') | YES  |     | PENDING   |                  |
| issued_date  | date                                    | YES  |     | curdate() | DEFAULT_GENERATED |
+--------------+-----------------------------------------+------+-----+-----------+------------------+
8 rows in set (0.00 sec)

mysql>
```

Describe account table

```
mysql> desc accholder;
+-----------------+---------------+------+-----+---------+-------+
| Field           | Type          | Null | Key | Default | Extra |
+-----------------+---------------+------+-----+---------+-------+
| acct_no         | bigint        | NO   | PRI | NULL    |       |
| holder_name     | varchar(50)   | NO   |     | NULL    |       |
| phone_no        | varchar(15)   | YES  |     | NULL    |       |
| email           | varchar(50)   | YES  |     | NULL    |       |
| address         | varchar(100)  | YES  |     | NULL    |       |
| initial_balance | decimal(15,2) | YES  |     | NULL    |       |
| loan_taken      | varchar(5)    | YES  |     | NULL    |       |
+-----------------+---------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

USER MANAGEMENT

```
============================================================
                BANK MANAGEMENT SYSTEM
============================================================


------------------MAIN MENU--------------------
[1]     USER MANAGMENT
[2]     ADMIN DASHBOARD

-------------------------------------------

[3]         NEW FEEDBACK
[4]         CUSTOMER SUPPORT / HELP DESK ☎
[0]         EXIT

Choose a menu option: 1
```

Account Holder Details

```
------------------USER MENU--------------------
[1]         ACCOUNT HOLDER DETAILS
[2]         VIEW LOAN DETAILS
[3]         BALANCE ENQUIRY
[4]         UPDATE NAME
[5]         UPDATE EMAIL
[6]         UPDATE PHONE NUMBER
[7]         UPDATE ADDRESS
[8]         LOAN SERVICES (Apply / EMI / Check Status)
[9]         CLOSE ACCOUNT
[10]        TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]    TRANSACTION HISTROY

-------------------------------------------

[13]        GIVE FEEDBACK
[14]        CUSTOMER SUPPORT / HELP DESK
[0]         EXIT

Choose a menu option: 1
Enter the account number: 1234567890
Initial balance:  50010.00
Loan Taken:  YES
*******************************************
```

View Loan Details

```
------------------USER MENU--------------------
[1]         ACCOUNT HOLDER DETAILS
[2]         VIEW LOAN DETAILS
[3]         BALANCE ENQUIRY
[4]         UPDATE NAME
[5]         UPDATE EMAIL
[6]         UPDATE PHONE NUMBER
[7]         UPDATE ADDRESS
[8]         LOAN SERVICES (Apply / EMI / Check Status)
[9]         CLOSE ACCOUNT
[10]        TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]    TRANSACTION HISTROY

-----------------------------------------

[13]        GIVE FEEDBACK
[14]        CUSTOMER SUPPORT / HELP DESK
[0]         EXIT

Choose a menu option: 2
Enter your account number: 1234567890

*************** LOAN DETAILS ***************
Loan ID: 1
Account Number: 1234567890
Loan Amount: 200000.00
Interest Rate: 7.50%
Tenure (Months): 7.50
EMI Amount: 24.00
Status: 8996.97
Application Date: ACTIVE
----------------------------------------
****************************************
```

Balance Enquiry

```
------------------USER MENU--------------------
[1]         ACCOUNT HOLDER DETAILS
[2]         VIEW LOAN DETAILS
[3]         BALANCE ENQUIRY
[4]         UPDATE NAME
[5]         UPDATE EMAIL
[6]         UPDATE PHONE NUMBER
[7]         UPDATE ADDRESS
[8]         LOAN SERVICES (Apply / EMI / Check Status)
[9]         CLOSE ACCOUNT
[10]        TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]    TRANSACTION HISTROY

-----------------------------------------

[13]        GIVE FEEDBACK
[14]        CUSTOMER SUPPORT / HELP DESK
[0]         EXIT

Choose a menu option: 3
Enter the account number: 1234567890

*************** BALANCE ENQUIRY ***************
Account number:  1234567890
Name of account holder:  Klein
Balance:  50010.00
*****************************************
```

Update Name

```
------------------USER MENU--------------------
[1]          ACCOUNT HOLDER DETAILS
[2]          VIEW LOAN DETAILS
[3]          BALANCE ENQUIRY
[4]          UPDATE NAME
[5]          UPDATE EMAIL
[6]          UPDATE PHONE NUMBER
[7]          UPDATE ADDRESS
[8]          LOAN SERVICES (Apply / EMI / Check Status)
[9]          CLOSE ACCOUNT
[10]         TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]     TRANSACTION HISTROY

-----------------------------------------------

[13]         GIVE FEEDBACK
[14]         CUSTOMER SUPPORT / HELP DESK
[0]          EXIT

Choose a menu option: 4
Enter the account number: 1234567890
Enter the updated name: Klein Moretti
Name successfully updated!
```

Update Email

```
------------------USER MENU--------------------
[1]         ACCOUNT HOLDER DETAILS
[2]         VIEW LOAN DETAILS
[3]         BALANCE ENQUIRY
[4]         UPDATE NAME
[5]         UPDATE EMAIL
[6]         UPDATE PHONE NUMBER
[7]         UPDATE ADDRESS
[8]         LOAN SERVICES (Apply / EMI / Check Status)
[9]         CLOSE ACCOUNT
[10]        TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]    TRANSACTION HISTROY

-----------------------------------------

[13]        GIVE FEEDBACK
[14]        CUSTOMER SUPPORT / HELP DESK
[0]         EXIT

Choose a menu option: 5
Enter the account number: 1234567890
Enter the updated email: kleinthefool@gmail.com
Email successfully updated!
```

Update Phone Number

```
------------------USER MENU--------------------
[1]         ACCOUNT HOLDER DETAILS
[2]         VIEW LOAN DETAILS
[3]         BALANCE ENQUIRY
[4]         UPDATE NAME
[5]         UPDATE EMAIL
[6]         UPDATE PHONE NUMBER
[7]         UPDATE ADDRESS
[8]         LOAN SERVICES (Apply / EMI / Check Status)
[9]         CLOSE ACCOUNT
[10]        TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]    TRANSACTION HISTROY

-----------------------------------------------

[13]        GIVE FEEDBACK
[14]        CUSTOMER SUPPORT / HELP DESK
[0]         EXIT

Choose a menu option: 6
Enter the account number: 1234567890
Enter the updated phone number: 9330724909
Phone number successfully updated!
```

Update Address

```
------------------USER MENU--------------------
[1]         ACCOUNT HOLDER DETAILS
[2]         VIEW LOAN DETAILS
[3]         BALANCE ENQUIRY
[4]         UPDATE NAME
[5]         UPDATE EMAIL
[6]         UPDATE PHONE NUMBER
[7]         UPDATE ADDRESS
[8]         LOAN SERVICES (Apply / EMI / Check Status)
[9]         CLOSE ACCOUNT
[10]        TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]    TRANSACTION HISTROY

-----------------------------------------

[13]        GIVE FEEDBACK
[14]        CUSTOMER SUPPORT / HELP DESK
[0]         EXIT

Choose a menu option: 7
Enter the account number: 1234567890
Enter the updated address: Unit 2 Daffodil Street, Tingen City
Address was updated successfully!
```

\\

LOAN SERVICES

Apply Loan

```
------------------USER MENU--------------------
[1]         ACCOUNT HOLDER DETAILS
[2]         VIEW LOAN DETAILS
[3]         BALANCE ENQUIRY
[4]         UPDATE NAME
[5]         UPDATE EMAIL
[6]         UPDATE PHONE NUMBER
[7]         UPDATE ADDRESS
[8]         LOAN SERVICES (Apply / EMI / Check Status)
[9]         CLOSE ACCOUNT
[10]        TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]    TRANSACTION HISTROY

-----------------------------------------------

[13]        GIVE FEEDBACK
[14]        CUSTOMER SUPPORT / HELP DESK
[0]         EXIT

Choose a menu option: 8
[1] Apply Loan
[2] View Loan Details
[3] Check Loan Status
Enter choice: 1
Enter your account number: 1234567890
Loan amount: 10000
Tenure (months): 6
Loan request submitted. EMI = 1703.31
```

View Loan Details

```
------------------USER MENU--------------------
[1]         ACCOUNT HOLDER DETAILS
[2]         VIEW LOAN DETAILS
[3]         BALANCE ENQUIRY
[4]         UPDATE NAME
[5]         UPDATE EMAIL
[6]         UPDATE PHONE NUMBER
[7]         UPDATE ADDRESS
[8]         LOAN SERVICES (Apply / EMI / Check Status)
[9]         CLOSE ACCOUNT
[10]        TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]    TRANSACTION HISTROY

-----------------------------------------

[13]        GIVE FEEDBACK
[14]        CUSTOMER SUPPORT / HELP DESK
[0]         EXIT

Choose a menu option: 8
[1] Apply Loan
[2] View Loan Details
[3] Check Loan Status
Enter choice: 2
Enter your account number: 1234567890

*************** LOAN DETAILS ***************
Loan ID: 1
Account Number: 1234567890
Loan Amount: 200000.00
Interest Rate: 7.50%
Tenure (Months): 7.50
EMI Amount: 24.00
Status: 8996.97
Application Date: ACTIVE
----------------------------------------
Loan ID: 16
Account Number: 1234567890
Loan Amount: 10000.00
Interest Rate: 7.50%
Tenure (Months): 7.50
EMI Amount: 6.00
Status: 1703.31
Application Date: PENDING
----------------------------------------
****************************************
```

Check Loan Status

```
------------------USER MENU--------------------
[1]         ACCOUNT HOLDER DETAILS
[2]         VIEW LOAN DETAILS
[3]         BALANCE ENQUIRY
[4]         UPDATE NAME
[5]         UPDATE EMAIL
[6]         UPDATE PHONE NUMBER
[7]         UPDATE ADDRESS
[8]         LOAN SERVICES (Apply / EMI / Check Status)
[9]         CLOSE ACCOUNT
[10]        TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]    TRANSACTION HISTROY

-----------------------------------------

[13]        GIVE FEEDBACK
[14]        CUSTOMER SUPPORT / HELP DESK
[0]         EXIT

Choose a menu option: 8
[1] Apply Loan
[2] View Loan Details
[3] Check Loan Status
Enter choice: 3
Enter your account number: 1234567890
Loan 1: ACTIVE
Loan 16: PENDING
```

Close Account

```
------------------USER MENU--------------------
[1]         ACCOUNT HOLDER DETAILS
[2]         VIEW LOAN DETAILS
[3]         BALANCE ENQUIRY
[4]         UPDATE NAME
[5]         UPDATE EMAIL
[6]         UPDATE PHONE NUMBER
[7]         UPDATE ADDRESS
[8]         LOAN SERVICES (Apply / EMI / Check Status)
[9]         CLOSE ACCOUNT
[10]        TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]    TRANSACTION HISTROY

-----------------------------------------------

[13]        GIVE FEEDBACK
[14]        CUSTOMER SUPPORT / HELP DESK
[0]         EXIT

Choose a menu option: 9
Enter your account number: 1234567899
Are you sure you want to close your account? (Y/N): N
Account closure cancelled.
```

TRANSACTIONS

1. Deposit

```
------------------USER MENU--------------------
[1]          ACCOUNT HOLDER DETAILS
[2]          VIEW LOAN DETAILS
[3]          BALANCE ENQUIRY
[4]          UPDATE NAME
[5]          UPDATE EMAIL
[6]          UPDATE PHONE NUMBER
[7]          UPDATE ADDRESS
[8]          LOAN SERVICES (Apply / EMI / Check Status)
[9]          CLOSE ACCOUNT
[10]         TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]     TRANSACTION HISTROY

-----------------------------------------------

[13]         GIVE FEEDBACK
[14]         CUSTOMER SUPPORT / HELP DESK
[0]          EXIT

Choose a menu option: 10
Enter your account number: 1234567890

===== TRANSACTIONS =====
1. Deposit
2. Withdraw
3. Transfer
Enter choice: 1
Enter amount to deposit: 10000
```

2. Withdraw

```
------------------USER MENU--------------------
[1]          ACCOUNT HOLDER DETAILS
[2]          VIEW LOAN DETAILS
[3]          BALANCE ENQUIRY
[4]          UPDATE NAME
[5]          UPDATE EMAIL
[6]          UPDATE PHONE NUMBER
[7]          UPDATE ADDRESS
[8]          LOAN SERVICES (Apply / EMI / Check Status)
[9]          CLOSE ACCOUNT
[10]         TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]     TRANSACTION HISTROY

-----------------------------------------------

[13]         GIVE FEEDBACK
[14]         CUSTOMER SUPPORT / HELP DESK
[0]          EXIT

Choose a menu option: 10
Enter your account number: 1234567890

===== TRANSACTIONS =====
1. Deposit
2. Withdraw
3. Transfer
Enter choice: 2
Enter amount to withdraw: 10000
```

3. TRANSFER

```
------------------USER MENU--------------------
[1]         ACCOUNT HOLDER DETAILS
[2]         VIEW LOAN DETAILS
[3]         BALANCE ENQUIRY
[4]         UPDATE NAME
[5]         UPDATE EMAIL
[6]         UPDATE PHONE NUMBER
[7]         UPDATE ADDRESS
[8]         LOAN SERVICES (Apply / EMI / Check Status)
[9]         CLOSE ACCOUNT
[10]        TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]    TRANSACTION HISTROY

-----------------------------------------------

[13]        GIVE FEEDBACK
[14]        CUSTOMER SUPPORT / HELP DESK
[0]         EXIT

Choose a menu option: 10
Enter your account number: 1234567890

===== TRANSACTIONS =====
1. Deposit
2. Withdraw
3. Transfer
Enter choice: 3
Enter recipient account no: 1234567899
Enter amount to transfer: 10000
Transferred 10000.0 from 1234567890 to 1234567899
```

Transaction History

```
------------------USER MENU--------------------
[1]         ACCOUNT HOLDER DETAILS
[2]         VIEW LOAN DETAILS
[3]         BALANCE ENQUIRY
[4]         UPDATE NAME
[5]         UPDATE EMAIL
[6]         UPDATE PHONE NUMBER
[7]         UPDATE ADDRESS
[8]         LOAN SERVICES (Apply / EMI / Check Status)
[9]         CLOSE ACCOUNT
[10]        TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]    TRANSACTION HISTROY

-----------------------------------------------

[13]        GIVE FEEDBACK
[14]        CUSTOMER SUPPORT / HELP DESK
[0]         EXIT

Choose a menu option: 11
Enter your account number: 1234567890

 Transactions for Account 1234567890:
ID: 13, Type: DEPOSIT, Amount: 10000.00, Date: 2025-10-24
ID: 14, Type: WITHDRAW, Amount: 10000.00, Date: 2025-10-24
ID: 15, Type: TRANSFER_OUT, Amount: 10000.00, Date: 2025-10-24
ID: 12, Type: DEPOSIT, Amount: 10.00, Date: 2025-10-16
ID: 10, Type: deposit, Amount: 12000.00, Date: 2025-09-08
ID: 7, Type: withdrawal, Amount: 1000.00, Date: 2025-09-06
ID: 2, Type: withdrawal, Amount: 1500.00, Date: 2025-09-02
ID: 1, Type: deposit, Amount: 5000.00, Date: 2025-09-01
```

Feedback

```
------------------USER MENU--------------------
[1]         ACCOUNT HOLDER DETAILS
[2]         VIEW LOAN DETAILS
[3]         BALANCE ENQUIRY
[4]         UPDATE NAME
[5]         UPDATE EMAIL
[6]         UPDATE PHONE NUMBER
[7]         UPDATE ADDRESS
[8]         LOAN SERVICES (Apply / EMI / Check Status)
[9]         CLOSE ACCOUNT
[10]        TRANSACTIONS (Deposit, Withdraw, Transfer)
[11]    TRANSACTION HISTROY

-----------------------------------------

[13]        GIVE FEEDBACK
[14]        CUSTOMER SUPPORT / HELP DESK
[0]         EXIT

Choose a menu option: 13
Enter your account number: 1234567890
Enter your feedback: Very good!
Thank you for your feedback!
```

ADMIN DASHBOARD

```
============================================================
                BANK MANAGEMENT SYSTEM
============================================================


------------------MAIN MENU--------------------
[1]     USER MANAGMENT
[2]     ADMIN DASHBOARD
-----------------------------------------------

[3]         NEW FEEDBACK
[4]         CUSTOMER SUPPORT / HELP DESK ☎
[0]         EXIT

Choose a menu option: 2
Enter admin username: officer
Enter admin password: bank@321

 Admin Login Successful!
```

View All Account Holders

```
------------------ADMIN MENU--------------------
[1]     VIEW ALL ACCOUNT HOLDERS
[2]     VIEW FEEDBACK
[3]     ADD ACCOUNT HOLDER
[4]     NEW ADMIN ACCOUNT
[5]     APPROVE LOAN
[6]     LOAN DEFAULTERS
[7]     UPDATE ACCOUNT DETAILS
[8]     DELETE ACCOUNT (Permanent)
------------------------------------------------

[0]         EXIT

Choose a menu option: 1

*************** ALL ACCOUNTS ***************
Account Number: 1234567890
Name: Klein Moretti
Phone: 9330724909
Email: kleinthefool@gmail.com
Address: Unit 2 Daffodil Street, Tingen City
Balance: 40010.00
Loan Taken: YES
----------------------------------------
Account Number: 1234567891
Name: Jane Smith
Phone: 9876543211
Email: jane.smith@email.com
Address: 456 Oak Avenue
Balance: 75000.00
Loan Taken: NO
```

```
----------------------------------------
Account Number: 1234567892
Name: Peter Jones
Phone: 9876543212
Email: peter.jones@email.com
Address: 789 Pine Lane
Balance: 120000.00
Loan Taken: YES
----------------------------------------
Account Number: 1234567893
Name: Alice Brown
Phone: 9876543213
Email: alice.brown@email.com
Address: 12 Maple Street
Balance: 45000.00
Loan Taken: NO
----------------------------------------
Account Number: 1234567894
Name: David Wilson
Phone: 9876543214
Email: david.wilson@email.com
Address: 34 Elm Avenue
Balance: 62000.00
Loan Taken: YES
----------------------------------------
Account Number: 1234567895
Name: Sophia Taylor
Phone: 9876543215
Email: sophia.taylor@email.com
Address: 56 Cedar Road
Balance: 83000.00
Loan Taken: NO
```

View Feedback

```
------------------ADMIN MENU--------------------
[1]     VIEW ALL ACCOUNT HOLDERS
[2]     VIEW FEEDBACK
[3]     ADD ACCOUNT HOLDER
[4]     NEW ADMIN ACCOUNT
[5]     APPROVE LOAN
[6]     LOAN DEFAULTERS
[7]     UPDATE ACCOUNT DETAILS
[8]     DELETE ACCOUNT (Permanent)
------------------------------------------------

[0]         EXIT

Choose a menu option: 2

*************** USER FEEDBACKS ***************
Account Number: 1234567890
Feedback: Very good!
----------------------------------------
************************************************
```

ADD ACCOUNT HOLDER

```
------------------ADMIN MENU--------------------
[1]     VIEW ALL ACCOUNT HOLDERS
[2]     VIEW FEEDBACK
[3]     ADD ACCOUNT HOLDER
[4]     NEW ADMIN ACCOUNT
[5]     APPROVE LOAN
[6]     LOAN DEFAULTERS
[7]     UPDATE ACCOUNT DETAILS
[8]     DELETE ACCOUNT (Permanent)
------------------------------------------

[0]         EXIT

Choose a menu option: 3
Enter the name of account holder: Raj Sharma
Enter the phone number: 9056404505
Enter the email of account holder: rajsharma@hotmail.com
Enter the address of account holder: 12 st. Selina street. david gorrins road.
Enter the initial balance: 100
Whether loan taken (yes/no): no
Successfully new account was added with A/C No: 8969264846
```

New Admin Account

```
------------------ADMIN MENU--------------------
[1]     VIEW ALL ACCOUNT HOLDERS
[2]     VIEW FEEDBACK
[3]     ADD ACCOUNT HOLDER
[4]     NEW ADMIN ACCOUNT
[5]     APPROVE LOAN
[6]     LOAN DEFAULTERS
[7]     UPDATE ACCOUNT DETAILS
[8]     DELETE ACCOUNT (Permanent)
------------------------------------------

[0]         EXIT

Choose a menu option: 4
Enter new admin username: glazer
Enter new admin password: thefool

 New admin account created successfully with username: glazer
```

Approve Loan

```
------------------ADMIN MENU--------------------
[1]     VIEW ALL ACCOUNT HOLDERS
[2]     VIEW FEEDBACK
[3]     ADD ACCOUNT HOLDER
[4]     NEW ADMIN ACCOUNT
[5]     APPROVE LOAN
[6]     LOAN DEFAULTERS
[7]     UPDATE ACCOUNT DETAILS
[8]     DELETE ACCOUNT (Permanent)
------------------------------------------------

[0]         EXIT

Choose a menu option: 5
Enter Loan ID to approve: 16
Loan 16 approved successfully
```

Loan Defaulters

```
------------------ADMIN MENU--------------------
[1]     VIEW ALL ACCOUNT HOLDERS
[2]     VIEW FEEDBACK
[3]     ADD ACCOUNT HOLDER
[4]     NEW ADMIN ACCOUNT
[5]     APPROVE LOAN
[6]     LOAN DEFAULTERS
[7]     UPDATE ACCOUNT DETAILS
[8]     DELETE ACCOUNT (Permanent)
------------------------------------------------

[0]         EXIT

Choose a menu option: 6
(7, 'Michael Johnson', Decimal('120000.00'), 'DEFAULT')
(11, 'James Thomas', Decimal('175000.00'), 'DEFAULT')
```

Update Account Details

```
------------------ADMIN MENU--------------------
[1]     VIEW ALL ACCOUNT HOLDERS
[2]     VIEW FEEDBACK
[3]     ADD ACCOUNT HOLDER
[4]     NEW ADMIN ACCOUNT
[5]     APPROVE LOAN
[6]     LOAN DEFAULTERS
[7]     UPDATE ACCOUNT DETAILS
[8]     DELETE ACCOUNT (Permanent)
------------------------------------------

[0]         EXIT

Choose a menu option: 7

------------------UPDATE MENU--------------------
[1]         UPDATE NAME
[2]         UPDATE EMAIL
[3]         UPDATE PHONE NUMBER
[4]         UPDATE ADDRESS

Choose a menu option: 1
Enter the account number: 1234567896
Enter the updated name: Olivia Huntson
Name successfully updated!
```

Delete Account

```
------------------ADMIN MENU--------------------
[1]     VIEW ALL ACCOUNT HOLDERS
[2]     VIEW FEEDBACK
[3]     ADD ACCOUNT HOLDER
[4]     NEW ADMIN ACCOUNT
[5]     APPROVE LOAN
[6]     LOAN DEFAULTERS
[7]     UPDATE ACCOUNT DETAILS
[8]     DELETE ACCOUNT (Permanent)
------------------------------------------

[0]         EXIT

Choose a menu option: 8
Enter the account number to delete: 1234567896
Are you sure you want to permanently delete account 1234567896? (y/N): y
Account 1234567896 permanently deleted by admin.
```

EXITING THE BANK MANAGEMENT SYSTEM

```
============================================================
                BANK MANAGEMENT SYSTEM
============================================================


------------------MAIN MENU--------------------
[1]     USER MANAGMENT
[2]     ADMIN DASHBOARD
-----------------------------------------------

[3]         NEW FEEDBACK
[4]         CUSTOMER SUPPORT / HELP DESK ☎
[0]         EXIT

Choose a menu option: 0
Exited Successfully!
Thank you for using us.
```