BANK MANAGEMENT PROJECT

SWATHI XII A10
ANUSHKA KUSHWAHA XII A10

INDEX

- Problem definition
- Hardware and software requirement
- Future Enhancements
- Source Code
- Output
- Explanation of source code
- Bibliography

PROBLEM DEFINITION

Creating a bank management system involves designing a software application that can handle various banking operations and functionalities. The system needs to be secure, reliable, and user-friendly. A bank management system involves various components to
Facilitate effective user management, account management, transaction management, and loan management.

Project Overview

A bank management project aims to automate and streamline banking tasks such as account management, transactions, and customer interactions. It often involves a backend for handling data and a front end for user interaction.

Core Components

1. **Account Management:**
    - **Features:** Create, update, and delete customer accounts.
    - **Details:** Store customer information (e.g., name, contact details) and account specifics (e.g., account number, balance).

2. **Transaction Handling:**
    - **Features:** Process deposits, withdrawals, and transfers.
    - **Details:** Maintain transaction records, ensure proper

balance updates, and handle transaction validation.

3. **Customer Service:**
    - **Features:** Provide account balance checks, generate account statements, and assist with other inquiries.
    - **Details:** Offer easy access to account information and transaction history.

Technologies Used

1. **Database:**
    - **Purpose:** Store and manage account and transaction data.
    - **Technologies:** SQL-based database – MySQL

2. **Programming Language:**
    - **Purpose:** Implement the business logic and database interactions.
    - **Technologies:** Python

Typical Features

1. **Account Operations:**
    - **Create Account:** Register new accounts with initial details.
    - **View Account:** Display account information and balance.
    - **Update Account:** Modify account details as needed.
    - **Delete Account:** Remove accounts from the system.

2. **Transaction Operations:**
    - **Deposit:** Add funds to an account.
    - **Withdraw:** Remove funds from an account.
    - **Transfer:** Move funds between accounts.
3. **Reporting and Statements:**
    - **Generate Statements:** Provide account statements for

specified periods.
    - **Transaction History:** Show a log of all transactions for an account.

Security Considerations

1. **Data Protection:**
    - **Access Control:** Implement user authentication and authorization to protect account data.

2. **Validation:**
    - **Input Validation:** Ensure all inputs are checked to prevent errors and security issues.
    - **Transaction Validation:** Confirm that all transactions are legitimate and feasible.

Example of Workflow:

1. **User Registration:**
    - A user creates an account by providing the required details.
    - System verifies the information and creates the account.
2. **Transaction Handling:**
    - The user performs various transactions, and the system updates account balances accordingly.
3. **Account Management:**
    - The user can view, update, or delete their account details.
4. **Reporting:**
    - The user can request account statements and transaction histories.

Need for bank management:

Creating or managing a bank management system can be a significant endeavor with a variety of potential benefits and motivations.
Banks and financial institutions are constantly seeking skilled professionals to develop and manage their systems. Mastery of bank management systems can open doors to high-demand roles in technology and finance.
Working on bank management systems often involves using advanced technologies and solving complex problems, which can be intellectually rewarding and offer a chance to innovate.
Bank management systems are critical for daily financial operations, so developing or improving such systems can have a significant impact on a wide range of users.
For banks, it provides a way to streamline operations, ensure compliance, and improve security. For customers, it offers convenience, control, and enhanced financial management tools.

Conclusion:

A bank management project automates essential banking functions, improving efficiency and accuracy while ensuring data security. It typically involves backend development for handling data and frontend development for user interaction, and it can be implemented using various technologies depending on the specific requirements and scale of the project.

HARDWARE AND SOFTWARE REQUIREMENTS

OS	- Microsoft Windows 10 System type X86-64bit based
Processor	- Intel(R) core(TM) i5-8250U
CPU	- 2.4 GHz
Memory (RAM)	- 8.00 GB
Front End	-Python v3.8.2:7b3ab59, MSC v.1916 64 bit (AMD64)]
Back End	-MySQL Version 8.0

FUTURE ENHANCEMENTS

- **Input Validation:** Ensure all inputs are checked to prevent errors and security issues.
- **Transaction Validation:** Confirm that all transactions are legitimate and feasible
- **Encryption:** Secure sensitive data such as account details and transaction information.
- **Access Control:** Implement user authentication and authorization to protect account data.
- **Loan Management:**
  - Loan Applications: Process applications for personal, auto, home, and other types of loans.
  - Loan Tracking: Track loan status, payments, and outstanding balances.
  - Repayment Schedule: Generate and manage repayment schedules.
  - Interest Calculation: Calculate interest for savings accounts, loans, and fixed deposits.

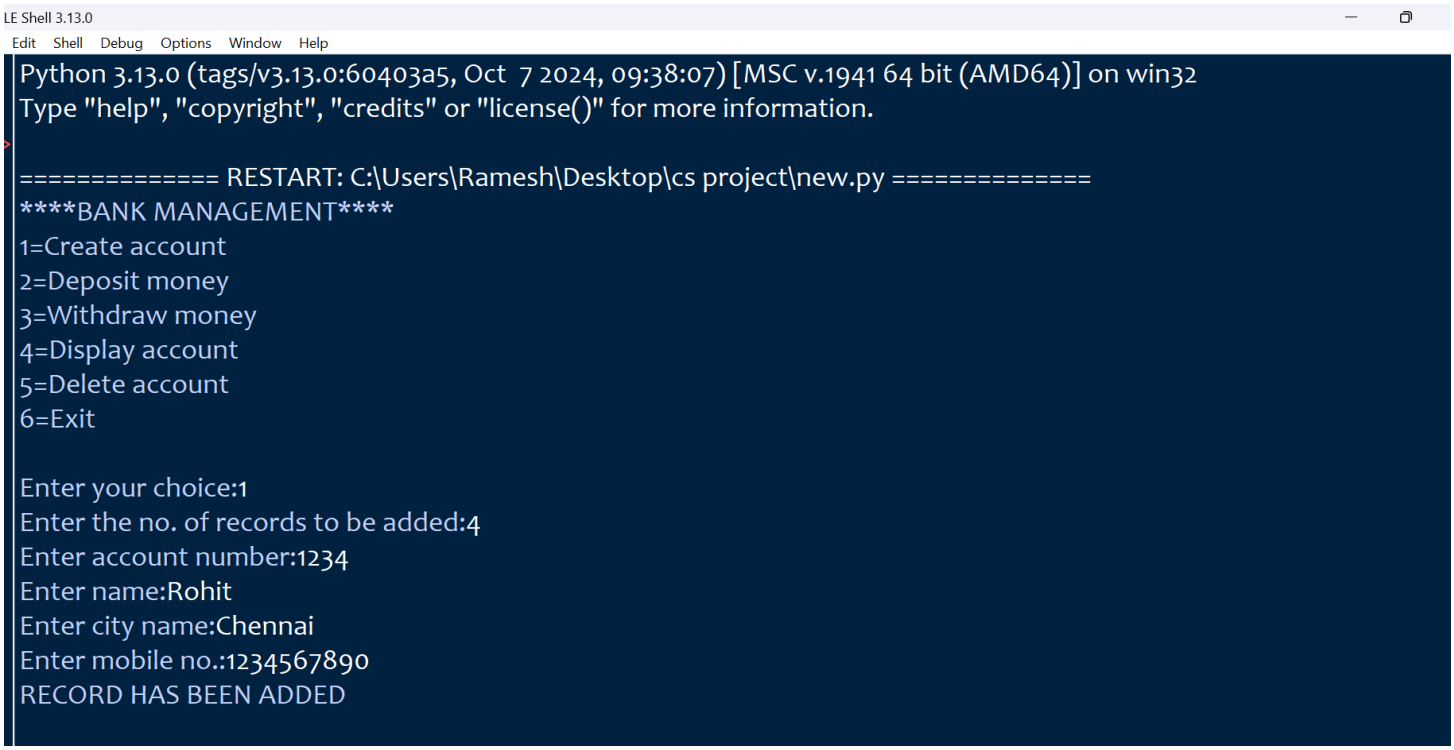
SOURCE CODE

```
print("****BANK MANAGEMENT****")

#creating database
import mysql.connector
mydb=mysql.connector.connect (host="localhost",user="root", passwd="Aspen@a43")
mycursor=mydb.cursor()
mycursor.execute("create database if not exists bank")
mycursor.execute("use bank")

def create():
#creating required tables
   mycursor.execute("create table if not exists records(acno char(4) primary key,name varchar(40),city
char(30),\
                mobileno char(10),balance int(10))")
   mycursor.execute("create table if not exists bank_trans(acno char (4),amount int(6),dot date,ttype
char(20),\
                foreign key (acno) references records(acno))")
   mydb.commit()

#ADDING RECORDS TO THE DATABASE
   n=int(input("Enter the no. of records to be added:"))
   for i in range(n):
      acno=str(input("Enter account number:"))
      mycursor.execute("select * from records where acno='{an}'".format(an=acno))
      myrecords=mycursor.fetchall()
      if len(myrecords)==0:
         name=input("Enter name:")
         city=str(input("Enter city name:"))
         mobile=str(input("Enter mobile no.:"))
         balance=0
         mycursor.execute("insert into records
values('"+acno+"','"+name+"','"+city+"','"+mobile+"','"+str(balance)+"')")
         mydb.commit()
         print("RECORD HAS BEEN ADDED\n")
      else:
         print("Account already exist")

#UPDATING DETAILS AFTER THE DEPOSITION OF MONEY BY THE APPLICANT
def deposit():
      print("DEPOSITING MONEY TO THE APPLICANTS ACCOUNT")
      acno=str(input("Enter account number:"))
      mycursor.execute("select * from records where acno='{an}'".format(an=acno))
      myrecords=mycursor.fetchall()
      if len(myrecords)!=0:
         dep=int(input("Enter amount to be deposited:"))
         dot=str(input("Enter date of Transaction: YYYY-MM-DD "))
         ttype="deposited"
         mycursor.execute("insert into bank_trans values('"+acno+"','"+str(dep)+"','"+dot+"','"+ttype+"')")
         mycursor.execute("update records set balance=balance+'"+str(dep)+"' where acno='"+acno+"'")
         mydb.commit()
         print("MONEY HAS BEEN DEPOSITED TO YOUR ACCOUNT\n")
      else:
```

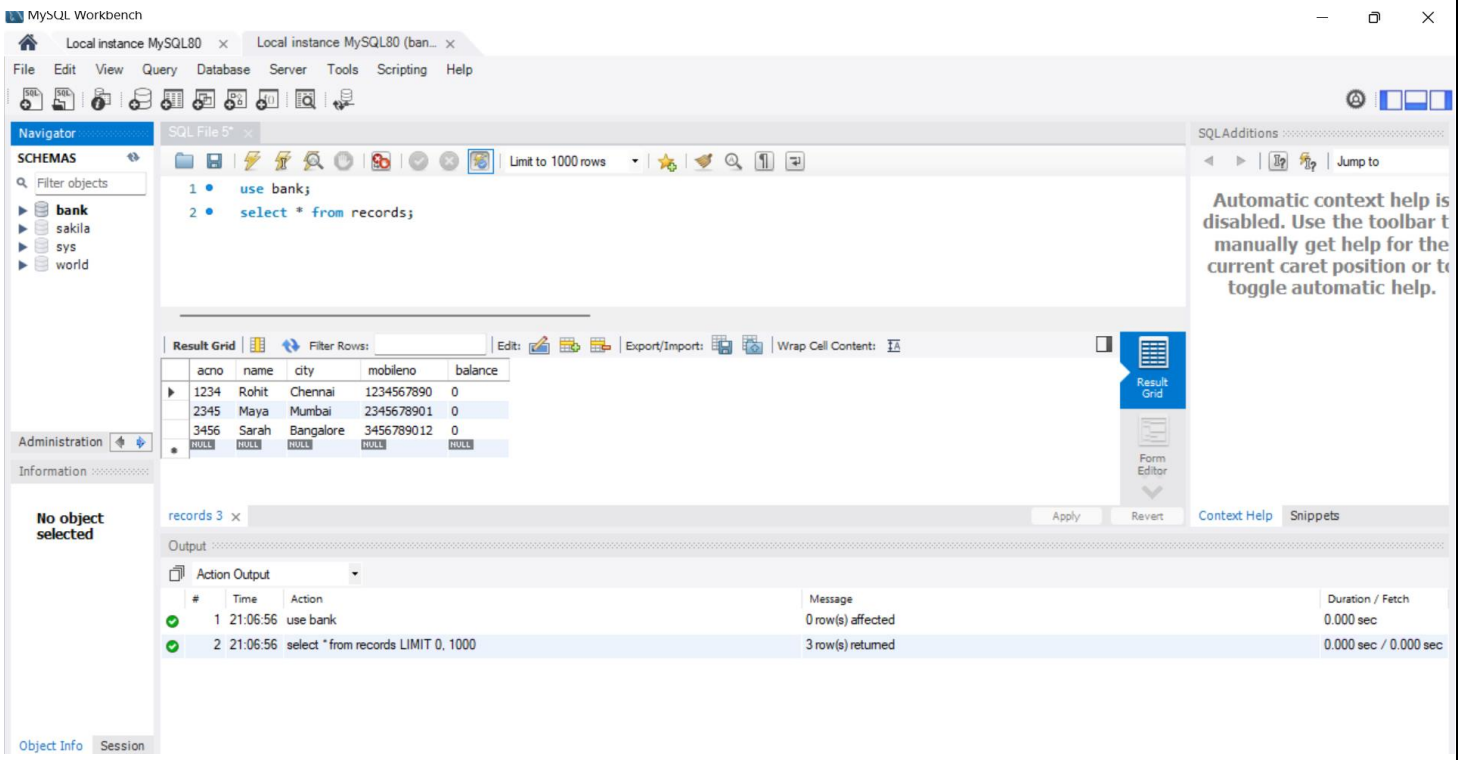
```
            print("Account doesn't exist")


#UPDATING THE DETAILS OF ACCOUNT AFTER THE WITHDRAW OF MONEY BY THE
APPLICANT
def withdraw():
        while True:
            print("WITHDRAWING MONEY FROM APPLICANT")
            acno=str(input("Enter account number:"))
            mycursor.execute("select * from records where acno='{an}'".format(an=acno))
            myrecords=mycursor.fetchall()
            if len(myrecords)!=0:
                wd=int(input("Enter amount to be withdrawn:"))
                dot=str(input("enter date of transaction: YYYY-MM-DD "))
                mycursor.execute("select balance from records where acno='"+acno+"'")
                myrecords=mycursor.fetchone()
                x=int(myrecords[0])
                if wd>x:
                    print("Not enough funds in account! Cannot withdraw money")
                    print("Current balance: ",x)
                    break
                ttype="withdrawn"
                mycursor.execute("insert into bank_trans values('"+acno+"','"+str(wd)+"','"+dot+"','"+ttype+"')")
                mycursor.execute("update records set balance=balance-'"+str(wd)+"' where
acno='"+acno+"'")
                mydb.commit()
                print("MONEY HAS BEEN WITHDRAWN FROM YOUR ACCOUNT\n")
                break
            else:
                print("Account doesn't exist")
                break

#DISPLAYING AN ALREADY EXISTING ACCOUNT
def details():
    try:
        print("DISPLAYING APPLICANT'S ACCOUNT")
        acno=str(input("Enter your account number:"))
        mycursor.execute("select * from records where acno='"+acno+"'")
        myrecords=mycursor.fetchall()
        print('acno: '+myrecords[0][0]+"\n"+'name: '+myrecords[0][1]+'\n'+'city:
'+myrecords[0][2]+'\n'+'mobile no.: '\
              +myrecords[0][3]+'\n'+'balance: '+str(myrecords[0][4]))
    except:
        print("Account doesn't exist")


#DELETING AN ALREADY EXISTING ACCOUNT
def delete():
        print("DELETING APPLICANT'S ACCOUNT")
        acno=str(input("Enter your account number:"))
        mycursor.execute("select * from records where acno='{an}'".format(an=acno))
        myrecords=mycursor.fetchall()
        if len(myrecords)!=0:
            mycursor.execute("delete from bank_trans where acno='"+acno+"'")
            mycursor.execute("delete from records where acno='"+acno+"'")
            mydb.commit()
            print("Account has been deleted")
        else:
```

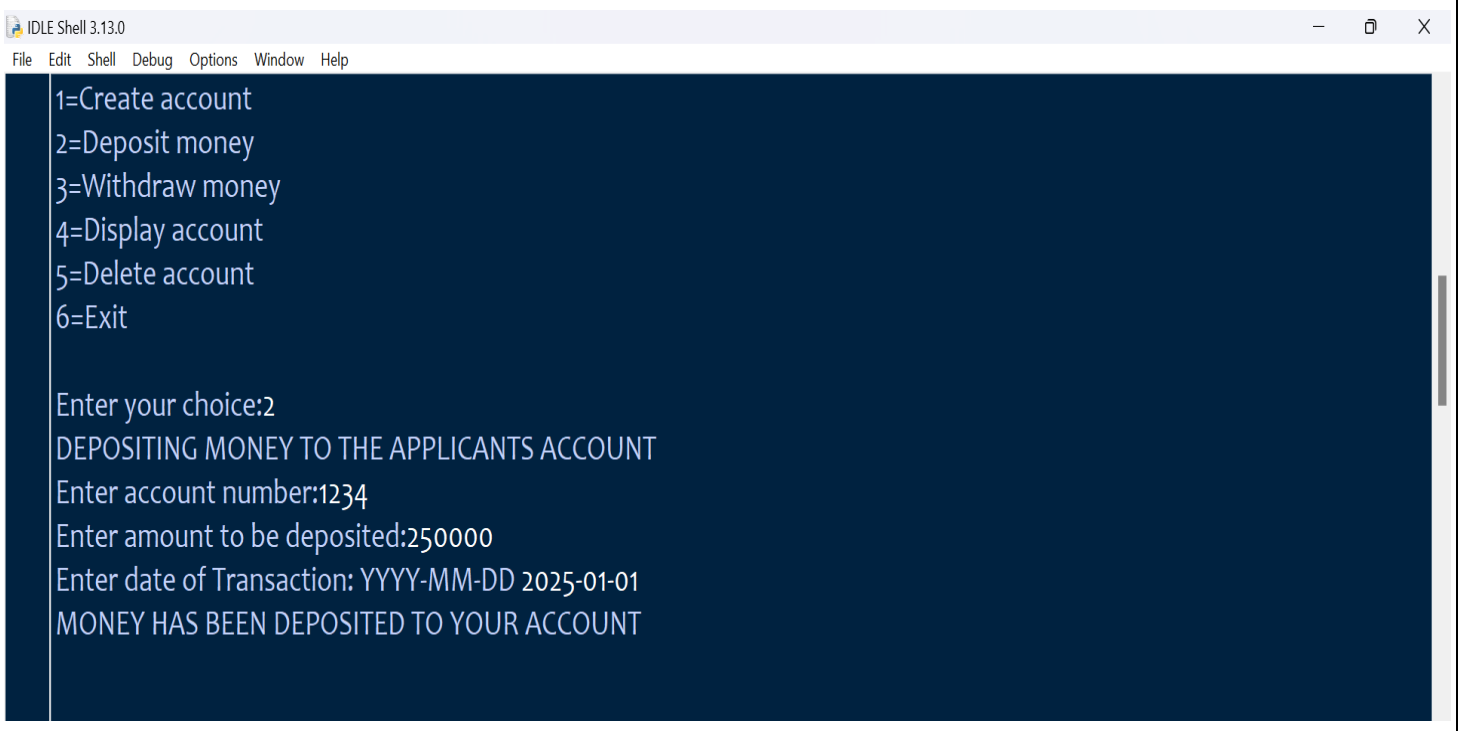
```
            print("Account doesn't exist")


ch=' '
while ch!=6:
    print("1=Create account")
    print("2=Deposit money")
    print("3=Withdraw money")
    print("4=Display account")
    print("5=Delete account")
    print("6=Exit")
    print()
    ch=int(input("Enter your choice:"))
    if ch==1:
        create()
        print('\t')
    elif ch==2:
        deposit()
        print('\t')
    elif ch==3:
        withdraw()
        print('\t')
    elif ch==4:
        details()
        print('\t')
    elif ch==5:
        delete()
        print('\t')
```

OUTPUT


LE Shell 3.13.0
Edit Shell Debug Options Window Help
Python 3.13.0 (tags/v3.13.0:60403a5, Oct 7 2024, 09:38:07) [MSC v.1941 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
========== RESTART: C:\Users\Ramesh\Desktop\cs project\new.py ==========
****BANK MANAGEMENT****
1=Create account
2=Deposit money
3=Withdraw money
4=Display account
5=Delete account
6=Exit
Enter your choice:1
Enter the no. of records to be added:4
Enter account number:1234
Enter name:Rohit
Enter city name:Chennai
Enter mobile no.:1234567890
RECORD HAS BEEN ADDED


IDLE Shell 3.13.0
File Edit Shell Debug Options Window Help
Enter account number:2345
Enter name:Maya
Enter city name:Mumbai
Enter mobile no.:2345678901
RECORD HAS BEEN ADDED
Enter account number:3456
Enter name:Sarah
Enter city name:Bangalore
Enter mobile no.:3456789012
RECORD HAS BEEN ADDED
Enter account number:2345
Account already exist

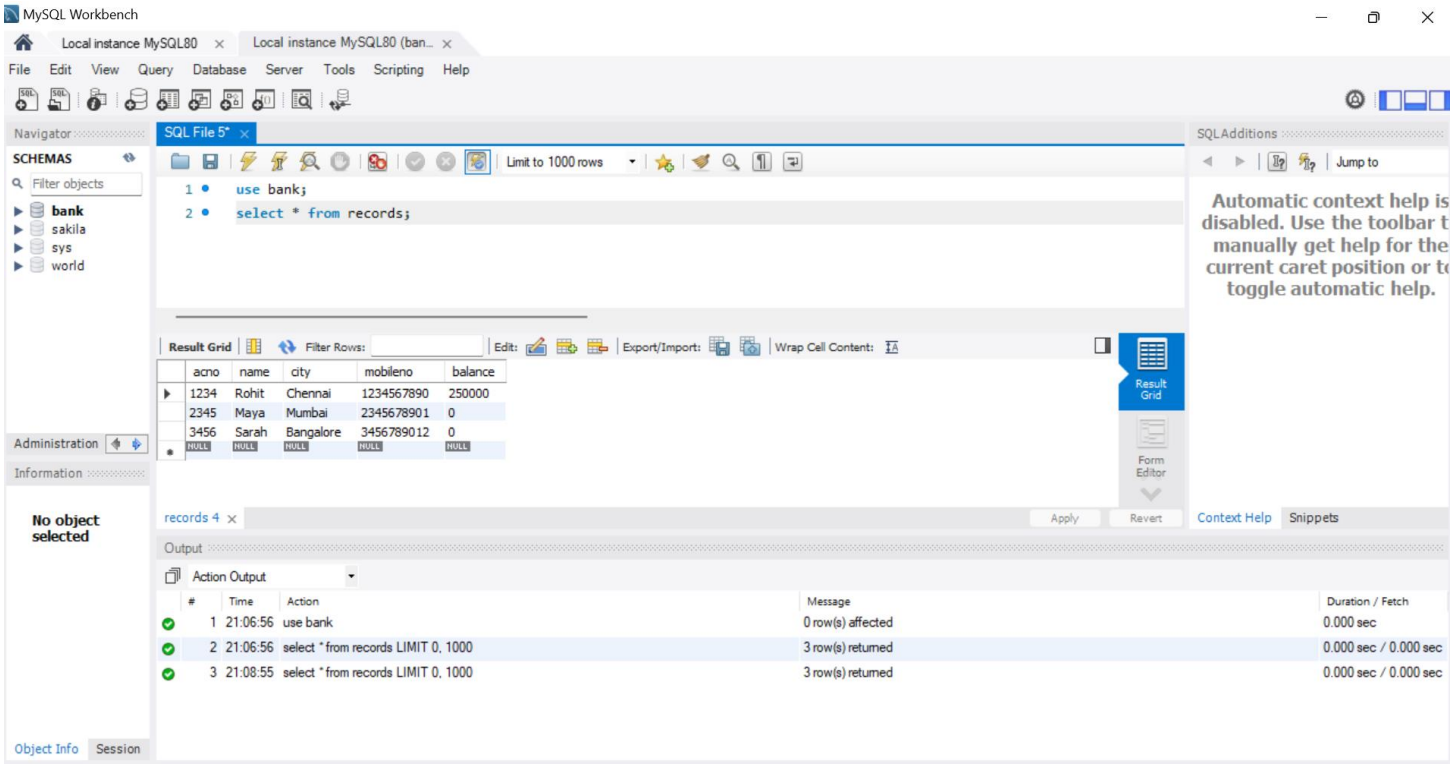

MySQL Workbench
Local instance MySQL80
Local instance MySQL80 (ban...
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
bank
sakila
sys
world
SQL File 5*
Limit to 1000 rows
1 use bank;
2 select * from records;
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
acno name city mobileno balance
1234 Rohit Chennai 1234567890 0
2345 Maya Mumbai 2345678901 0
3456 Sarah Bangalore 3456789012 0
Result Grid
Form Editor
Administration
Information
No object selected
records 3
Apply
Revert
Context Help
Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 21:06:56 use bank 0 row(s) affected 0.000 sec
2 21:06:56 select * from records LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
Object Info
Session


IDLE Shell 3.13.0
File Edit Shell Debug Options Window Help
1=Create account
2=Deposit money
3=Withdraw money
4=Display account
5=Delete account
6=Exit
Enter your choice:2
DEPOSITING MONEY TO THE APPLICANTS ACCOUNT
Enter account number:1234
Enter amount to be deposited:250000
Enter date of Transaction: YYYY-MM-DD 2025-01-01
MONEY HAS BEEN DEPOSITED TO YOUR ACCOUNT

IDLE Shell 3.13.0

File Edit Shell Debug Options Window Help

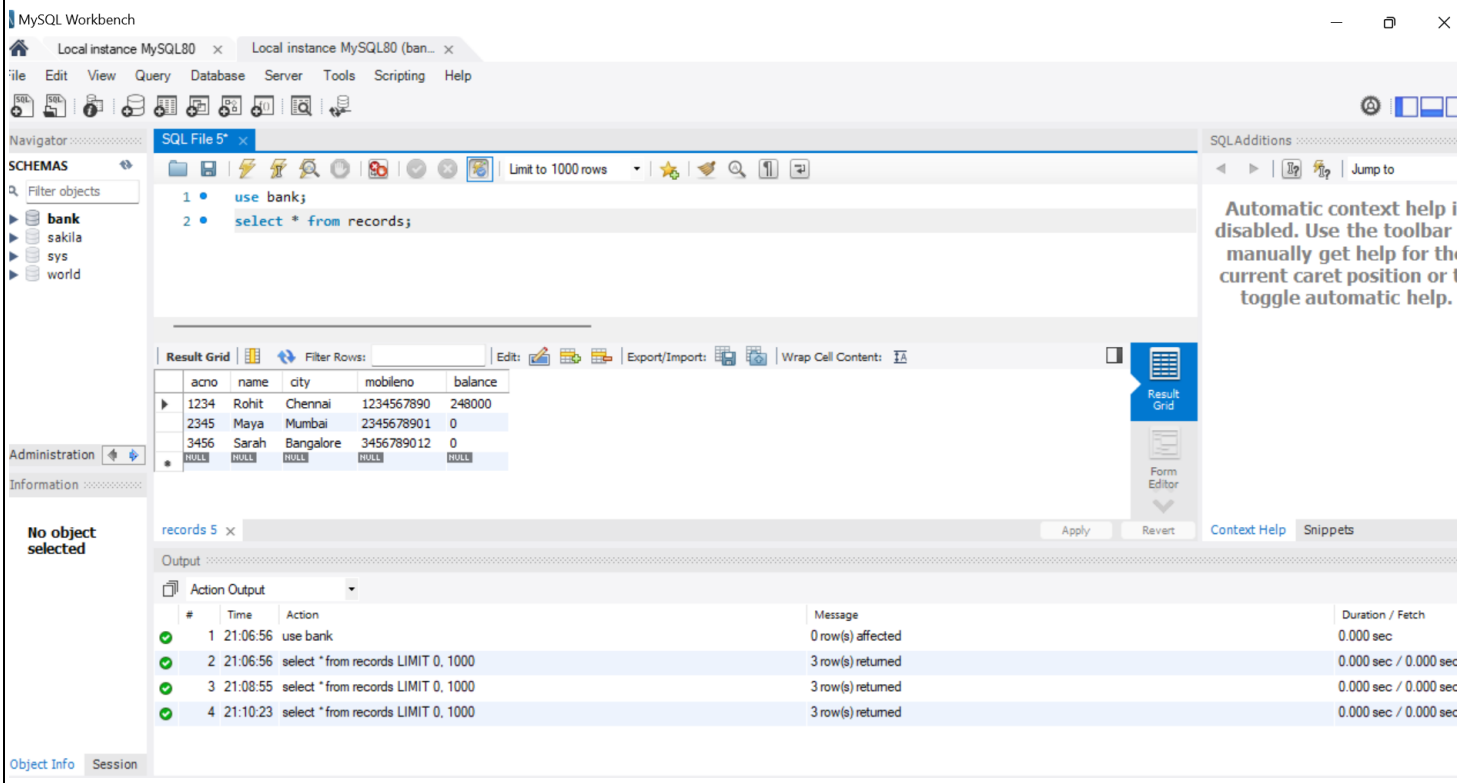
```
1=Create account
2=Deposit money
3=Withdraw money
4=Display account
5=Delete account
6=Exit

Enter your choice:2
DEPOSITING MONEY TO THE APPLICANTS ACCOUNT
Enter account number:5678
Account doesn't exist
```



```
1=Create account
2=Deposit money
3=Withdraw money
4=Display account
5=Delete account
6=Exit

Enter your choice:3
WITHDRAWING MONEY FROM APPLICANT
Enter account number:1234
Enter amount to be withdrawn:2000
enter date of transaction: YYYY-MM-DD 2025-01-03
MONEY HAS BEEN WITHDRAWN FROM YOUR ACCOUNT
```

```
1=Create account
2=Deposit money
3=Withdraw money
4=Display account
5=Delete account
6=Exit

Enter your choice:3
WITHDRAWING MONEY FROM APPLICANT
Enter account number:2345
Enter amount to be withdrawn:2500
enter date of transaction: YYYY-MM-DD 2025-01-02
Not enough funds in account! Cannot withdraw money
Current balance:  0
```

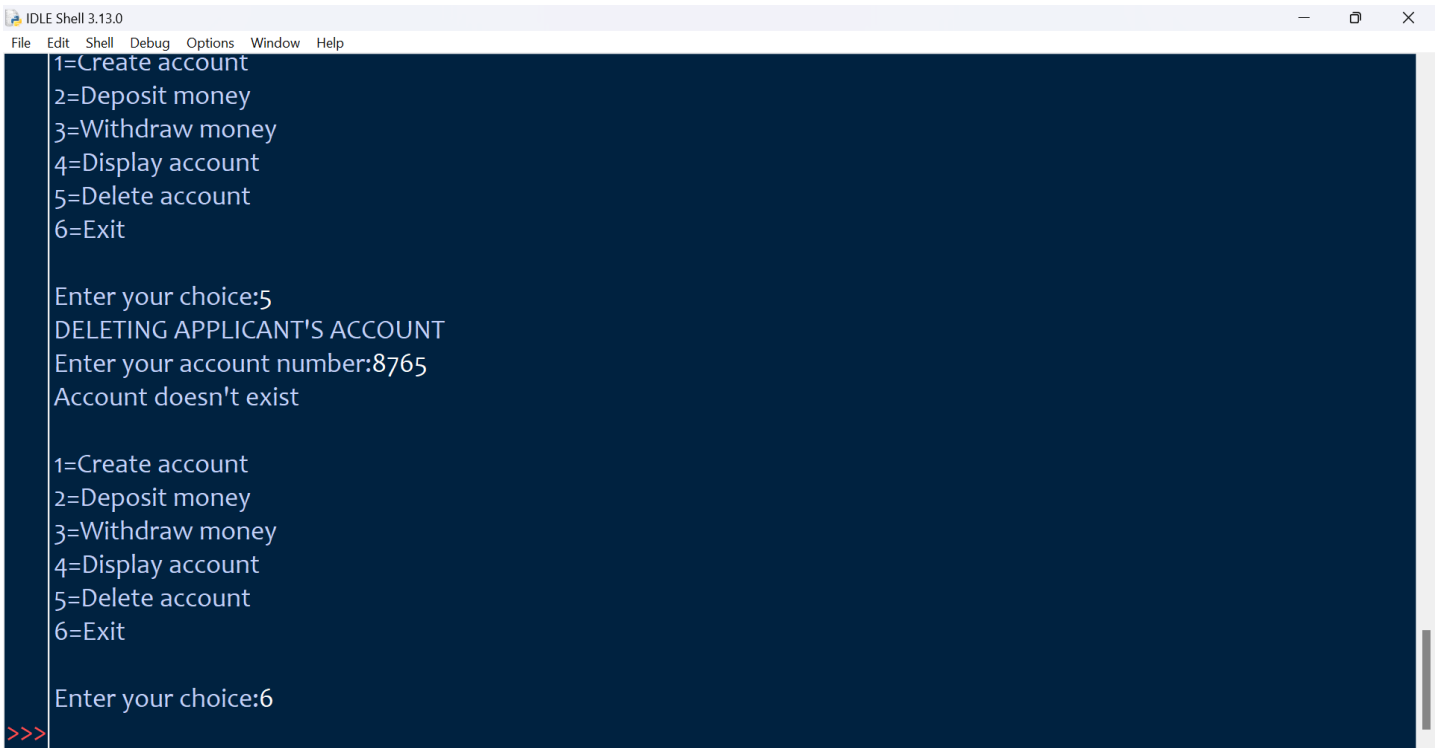
```
1=Create account
2=Deposit money
3=Withdraw money
4=Display account
5=Delete account
6=Exit

Enter your choice:3
WITHDRAWING MONEY FROM APPLICANT
Enter account number:6789
Account doesn't exist
```

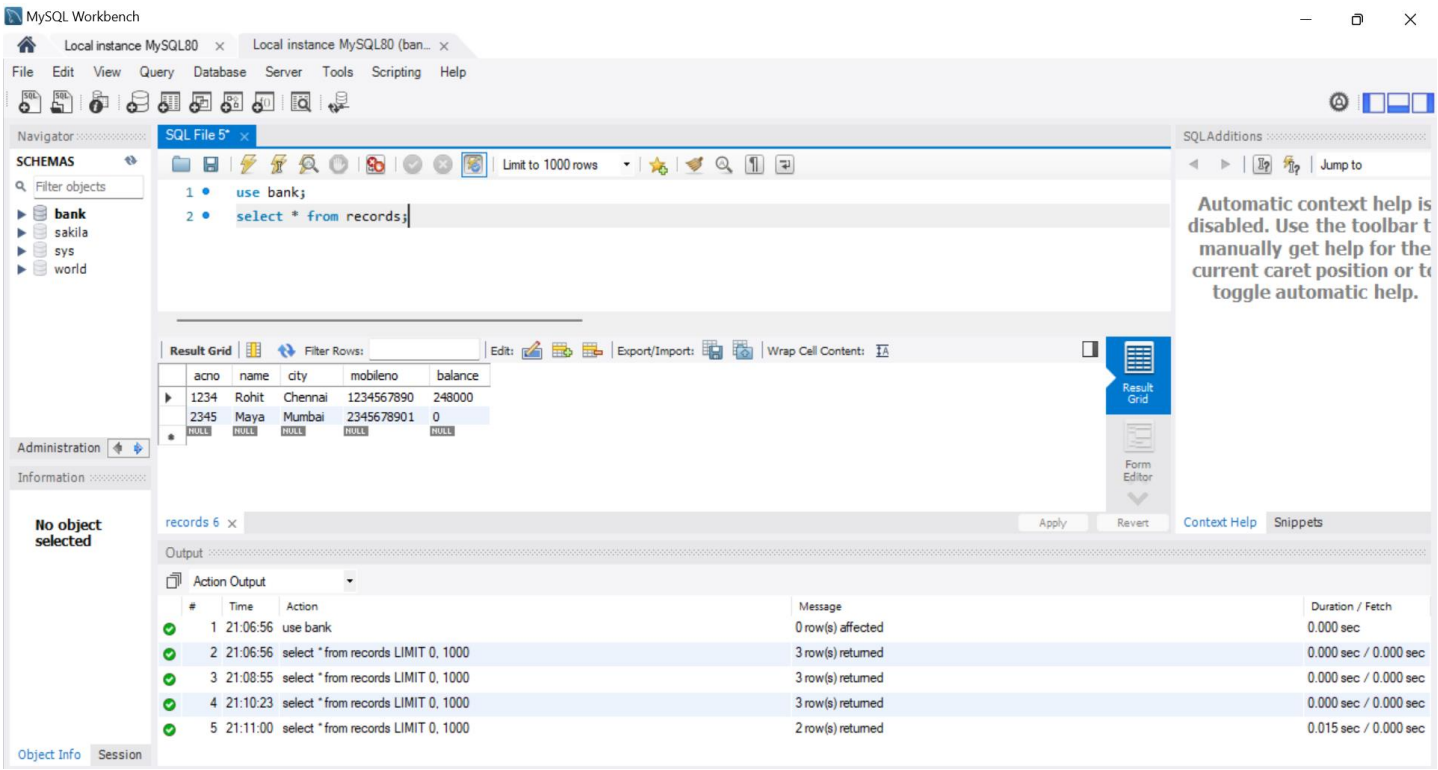


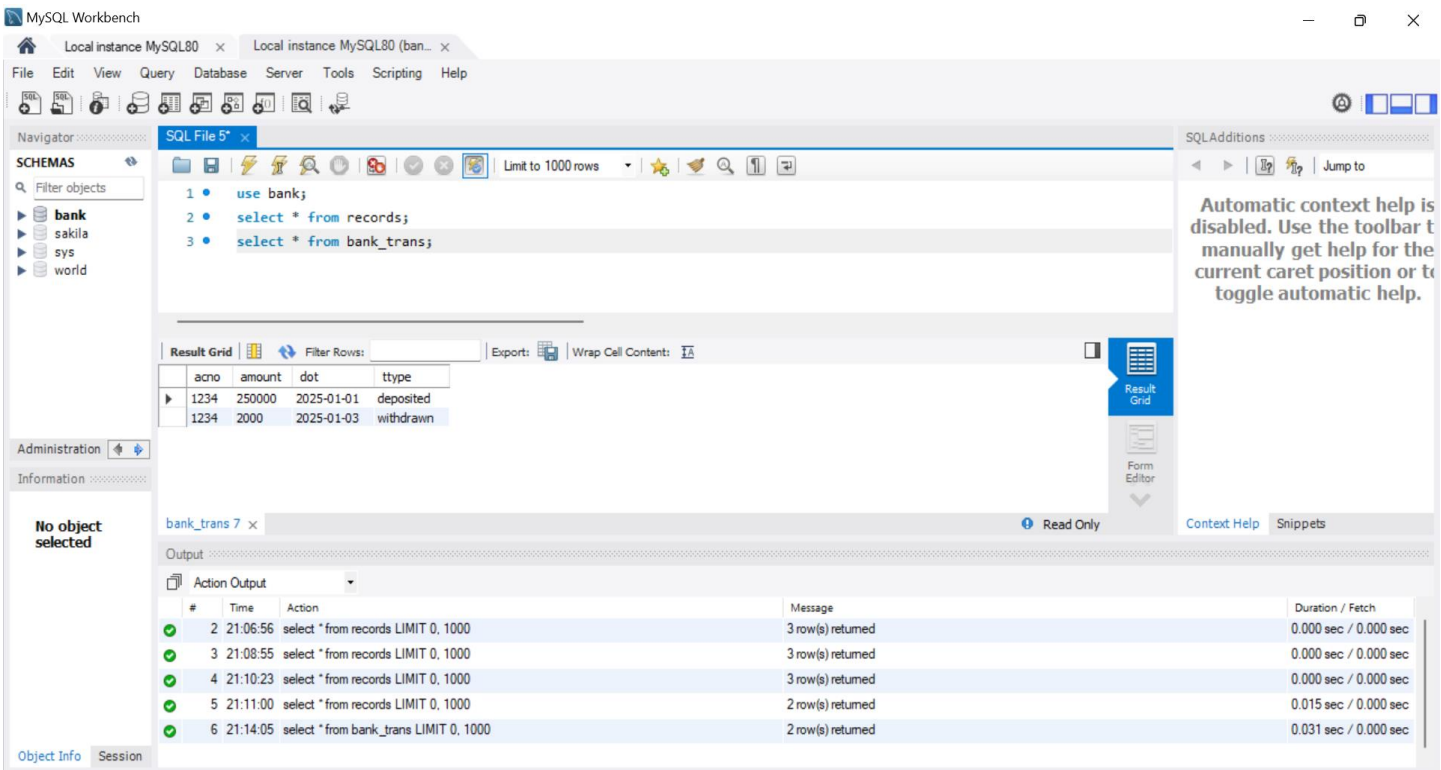

```
1=Create account
2=Deposit money
3=Withdraw money
4=Display account
5=Delete account
6=Exit

Enter your choice:4
DISPLAYING APPLICANT'S ACCOUNT
Enter your account number:1234
acno: 1234
name: Rohit
city: Chennai
mobile no.: 1234567890
balance: 248000
```

```
1=Create account
2=Deposit money
3=Withdraw money
4=Display account
5=Delete account
6=Exit

Enter your choice:4
DISPLAYING APPLICANT'S ACCOUNT
Enter your account number:5678
Account doesn't exist
```

```
1=Create account
2=Deposit money
3=Withdraw money
4=Display account
5=Delete account
6=Exit

Enter your choice:5
DELETING APPLICANT'S ACCOUNT
Enter your account number:3456
Account has been deleted
```




MySQL Workbench
Local instance MySQL80
Local instance MySQL80 (ban...
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
bank
sakila
sys
world
SQL File 5*
Limit to 1000 rows
1 use bank;
2 select * from records;
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
acno name city mobileno balance
1234 Rohit Chennai 1234567890 248000
2345 Maya Mumbai 2345678901 0
Result Grid
Form Editor
Administration
Information
No object selected
records 6
Apply
Revert
Context Help
Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 21:06:56 use bank 0 row(s) affected 0.000 sec
2 21:06:56 select * from records LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
3 21:08:55 select * from records LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
4 21:10:23 select * from records LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
5 21:11:00 select * from records LIMIT 0, 1000 2 row(s) returned 0.015 sec / 0.000 sec
Object Info
Session


MySQL Workbench
Local instance MySQL80
Local instance MySQL80 (ban...
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
bank
sakila
sys
world
SQL File 5*
Limit to 1000 rows
1 use bank;
2 select * from records;
3 select * from bank_trans;
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
acno amount dot ttype
1234 250000 2025-01-01 deposited
1234 2000 2025-01-03 withdrawn
Result Grid
Form Editor
Administration
Information
No object selected
bank_trans 7
Read Only
Context Help
Snippets
Output
Action Output
Time Action Message Duration / Fetch
2 21:06:56 select * from records LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
3 21:08:55 select * from records LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
4 21:10:23 select * from records LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
5 21:11:00 select * from records LIMIT 0, 1000 2 row(s) returned 0.015 sec / 0.000 sec
6 21:14:05 select * from bank_trans LIMIT 0, 1000 2 row(s) returned 0.031 sec / 0.000 sec
Object Info
Session

EXPLANATION OF SOURCE CODE

1. Database Connection and Cursor Setup

python
Copy code

```
import mysql.connector
mydb = mysql.connector.connect(host="localhost", user="root", passwd="Aspen@a43")
mycursor = mydb.cursor()
```

- **`mysql.connector.connect`**: This function connects to the MySQL database. The parameters passed include:
  - `host`: The database server address (in this case, it's `localhost`).
  - `user`: The username for MySQL (here it's `root`).
  - `passwd`: The password for the MySQL user.
- **`mycursor`**: The cursor object (`mydb.cursor()`) is used to interact with the MySQL database. You can use the cursor to execute SQL queries.

2. Create Database and Tables

python
Copy code

```
mycursor.execute("create database if not exists bank")
mycursor.execute("use bank")
```

- **`create database if not exists bank`**: This creates a database named `bank` if it doesn't already exist.
- **`use bank`**: This switches to the `bank` database for executing further SQL queries.

Creating Tables

python
Copy code

```
def create():
    mycursor.execute("create table if not exists records(acno char(4) primary key, name
varchar(40), city char(30), mobileno char(10), balance int(10))")
    mycursor.execute("create table if not exists bank_trans(acno char(4), amount int(6),
dot date, ttype char(20), foreign key (acno) references records(acno))")
    mydb.commit()
```

- **`records` table**:
  - `acno`: Account number (4 characters), primary key.
  - `name`: Account holder's name (up to 40 characters).
  - `city`: City of the account holder (up to 30 characters).
  - `mobileno`: Mobile number (10 characters).
  - `balance`: Account balance (integer).
- **`bank_trans` table**:
  - `acno`: Account number (foreign key referring to `acno` in the `records` table).
  - `amount`: The transaction amount.
  - `dot`: Date of the transaction.
  - `ttype`: Type of transaction (either `deposited` or `withdrawn`).

3. Create Account Function

python
Copy code

```
def create():
    n = int(input("Enter the no. of records to be added:"))
    for i in range(n):
        acno = str(input("Enter account number:"))
        mycursor.execute("select * from records where acno='{an}'".format(an=acno))
        myrecords = mycursor.fetchall()
        if len(myrecords) == 0:
            name = input("Enter name:")
            city = str(input("Enter city name:"))
            mobile = str(input("Enter mobile no.:"))
            balance = 0
            mycursor.execute("insert into records
values('"+acno+"','"+name+"','"+city+"','"+mobile+"','"+str(balance)+"')")
            mydb.commit()
            print("RECORD HAS BEEN ADDED\n")
        else:
            print("Account already exists")
```

- **`n = int(input(...))`**: Prompts the user for how many accounts they want to create.
- **`for i in range(n)`**: Loop to create multiple accounts based on the user's input.
- **`acno = str(input(...))`**: User enters the account number.
- **`mycursor.execute(...).fetchall()`**: Checks if the account already exists in the database. If no records are found (`len(myrecords) == 0`), a new account is created.
- **`mycursor.execute(...)`**: The `insert` statement adds the new account to the `records` table. The account is initialized with a balance of `0`.
- **`mydb.commit()`**: This commits the transaction to the database, making the changes permanent.

4. Deposit Money Function

python
Copy code

```
def deposit():
    print("DEPOSITING MONEY TO THE APPLICANT'S ACCOUNT")
    acno = str(input("Enter account number:"))
    mycursor.execute("select * from records where acno='{an}'".format(an=acno))
    myrecords = mycursor.fetchall()
    if len(myrecords) != 0:
        dep = int(input("Enter amount to be deposited:"))
        dot = str(input("Enter date of Transaction: YYYY-MM-DD "))
        ttype = "deposited"
        mycursor.execute("insert into bank_trans
values('"+acno+"','"+str(dep)+"','"+dot+"','"+ttype+"')")
        mycursor.execute("update records set balance=balance+'"+str(dep)+"' where
acno='"+acno+"'")
        mydb.commit()
        print("MONEY HAS BEEN DEPOSITED TO YOUR ACCOUNT\n")
    else:
        print("Account doesn't exist")
```

- The user is prompted for the account number to which they want to deposit money.
- **`mycursor.execute(...)`** checks if the account exists.
- **`dep = int(input(...))`**: User enters the amount to be deposited.
- **`ttype = "deposited"`**: The transaction type is marked as `deposited`.
- **`mycursor.execute(...)`**: The `insert` statement adds the transaction to the `bank_trans` table, and the `update` statement increases the account balance by the deposited amount.
- **`mydb.commit()`**: Saves the changes to the database.

5. Withdraw Money Function

python
Copy code

```
def withdraw():
    while True:
        print("WITHDRAWING MONEY FROM APPLICANT'S ACCOUNT")
        acno = str(input("Enter account number:"))
        mycursor.execute("select * from records where acno='{an}'".format(an=acno))
        myrecords = mycursor.fetchall()
        if len(myrecords) != 0:
            wd = int(input("Enter amount to be withdrawn:"))
            dot = str(input("Enter date of transaction: YYYY-MM-DD "))
            mycursor.execute("select balance from records where acno='"+acno+"'")
            myrecords = mycursor.fetchone()
            x = int(myrecords[0])
            if wd > x:
                print("Not enough funds in account! Cannot withdraw money")
                print("Current balance: ", x)
                break
            ttype = "withdrawn"
            mycursor.execute("insert into bank_trans
values('"+acno+"','"+str(wd)+"','"+dot+"','"+ttype+"')")
            mycursor.execute("update records set balance=balance-'"+str(wd)+"' where
acno='"+acno+"'")
            mydb.commit()
            print("MONEY HAS BEEN WITHDRAWN FROM YOUR ACCOUNT\n")
            break
        else:
            print("Account doesn't exist")
            break
```

- The user enters the account number from which they want to withdraw money.
- **`mycursor.execute(...)`** checks if the account exists.
- **`wd = int(input(...))`**: User enters the withdrawal amount.
- **`mycursor.execute("select balance from records where acno='"+acno+"'")`**: Retrieves the current balance to check if there are enough funds for withdrawal.
- If the withdrawal amount exceeds the balance, an error message is displayed.
- The transaction is recorded in the `bank_trans` table, and the balance is updated in the `records` table.

6. Display Account Details Function

python
Copy code

```
def details():
    try:
        print("DISPLAYING APPLICANT'S ACCOUNT")
        acno = str(input("Enter your account number:"))
        mycursor.execute("select * from records where acno='"+acno+"'")
        myrecords = mycursor.fetchall()
        print('acno: ' + myrecords[0][0] + "\n" + 'name: ' + myrecords[0][1] + '\n' +
'city: ' + myrecords[0][2] + '\n' + 'mobile no.: ' + myrecords[0][3] + '\n' + 'balance: ' +
str(myrecords[0][4]))
    except:
        print("Account doesn't exist")
```

- This function displays the details of an account.
- **`mycursor.execute("select * from records where acno='"+acno+"'")`** retrieves the details of the account.
- The details are printed to the console.
- If an exception occurs (e.g., the account doesn't exist), an error message is shown.

7. Delete Account Function

python
Copy code

```
def delete():
    print("DELETING APPLICANT'S ACCOUNT")
    acno = str(input("Enter your account number:"))
    mycursor.execute("select * from records where acno='{an}'".format(an=acno))
    myrecords = mycursor.fetchall()
    if len(myrecords) != 0:
        mycursor.execute("delete from bank_trans where acno='"+acno+"'")
        mycursor.execute("delete from records where acno='"+acno+"'")
        mydb.commit()
        print("Account has been deleted")
    else:
        print("Account doesn't exist")
```

- The user enters the account number they wish to delete.
- **`mycursor.execute(...)`** checks if the account exists.
- If the account exists, both the `bank_trans` and `records` tables are updated to remove the account.
- **`mydb.commit()`** ensures the changes are saved to the database.

8. Main Menu and Loop

python
Copy code

```
ch = ' '
while ch != 6:
    print("1=Create account")
    print("2=Deposit money")
    print("3=Withdraw money")
    print("4=Display account")
    print("5=Delete account")
    print("6=Exit")
    ch = int(input("Enter your choice:"))
    if ch == 1:
        create()
    elif ch == 2:
        deposit()
    elif ch == 3:
        withdraw()
    elif ch == 4:
        details()
    elif ch == 5:
        delete()
```

- This is a simple menu-driven loop that allows the user to choose between different actions:
    - `1` to create an account
    - `2` to deposit money
    - `3` to withdraw money
    - `4` to display account details
    - `5` to delete an account
    - `6` to exit the program

BIBLIOGRAPHY

- Computer Science with Python by Sumita Arora from Dhanpatrai &Co. publication
- www.mysql.com
- www.python.org