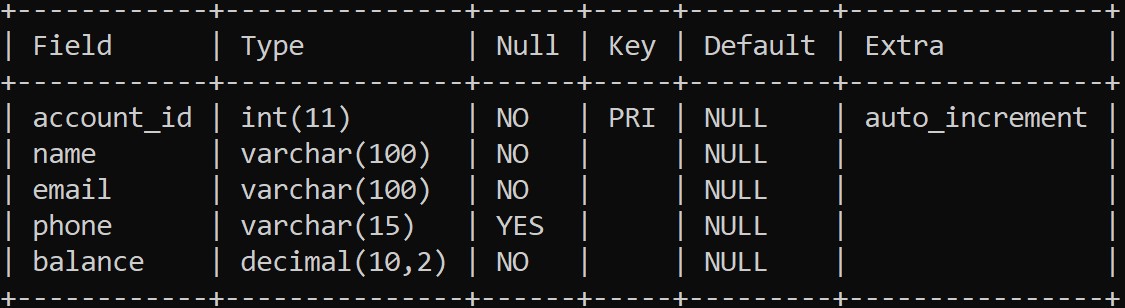
**Bank Management System**

**Project Requirements**

1. **Database Setup:**
   * **Database Name:** bank\_db
   * **Table Name:** accounts
   * **Columns in accounts table:**
     + account\_id (Primary Key, Auto Increment)
     + name (VARCHAR)
     + email (VARCHAR)
     + phone (VARCHAR)
     + balance (DECIMAL or FLOAT)
2. **Python & MySQL Integration:**
   * **MySQL Connector**: You are using mysql.connector for interacting with the MySQL database.
   * **Error Handling**: There is error handling for database connection and queries to ensure the application can recover gracefully from any issues.
   * **Account Management Operations**:
     + Create an account (insert data into the database).
     + View account details (fetch data from the database).
     + Update account balance (update the balance column).
     + Delete an account (delete an entry from the table).
3. **User Interface:**
   * **Menu System**: A simple menu with options to create, view, update, and delete accounts.
   * **User Input Validation**: The system asks for input from the user, validates input for numeric fields like balance and account IDs, and handles invalid input gracefully.
4. **Database Connection Logic**:
   * The connect\_to\_database function manages the connection to the MySQL database and handles any connection issues.

**Detailed Breakdown of Functions:**

1. **connect\_to\_database**:
   * Establishes a connection to the bank\_db MySQL database.
   * Returns the connection object or None if an error occurs.
2. **create\_account(name, email, phone, balance)**:
   * Connects to the database and inserts the new account details into the accounts table.
   * Requires name, email, phone, and balance as input.
   * Handles MySQL insertions and commits the changes.
3. **view\_account(account\_id)**:
   * Fetches account details from the accounts table based on the provided account\_id.
   * Displays the account information if found, otherwise displays an error message.
4. **update\_balance(account\_id, new\_balance)**:
   * Updates the balance of a given account using the account\_id.
   * Commits the new balance to the database and handles errors.
5. **delete\_account(account\_id)**:
   * Deletes an account from the accounts table using the provided account\_id.
   * Ensures that only existing accounts are deleted.
6. **bank\_menu()**:
   * A simple interactive menu for the user to select an option:
     + **1**: Create a new account.
     + **2**: View an account’s details.
     + **3**: Update an account’s balance.
     + **4**: Delete an account.
     + **5**: Exit the program.
   * The function loops, offering the menu until the user chooses to exit.

**Bank.py**

import mysql.connector

# Database connection

def connect\_to\_database():

try:

connection = mysql.connector.connect(

host="localhost",

user="root"

password=""

database="bank\_db"

)

print("Connected to the database successfully!")

return connection

except mysql.connector.Error as e:

print(f"Error connecting to MySQL: {e}")

return None

# Create a new account

def create\_account(name, email, phone, balance):

conn = connect\_to\_database()

if conn:

cursor = conn.cursor()

try:

query = "INSERT INTO accounts (name, email, phone, balance) VALUES (%s, %s, %s, %s)"

data = (name, email, phone, balance)

cursor.execute(query, data)

conn.commit()

print("Account created successfully!")

except mysql.connector.Error as e:

print(f"Error: {e}")

finally:

cursor.close()

conn.close()

# Read account details

def view\_account(account\_id):

conn = connect\_to\_database()

if conn:

cursor = conn.cursor()

try:

query = "SELECT \* FROM accounts WHERE account\_id = %s"

cursor.execute(query, (account\_id,))

account = cursor.fetchone()

if account:

print(f"Account Details: {account}")

else:

print("Account not found.")

except mysql.connector.Error as e:

print(f"Error: {e}")

finally:

cursor.close()

conn.close()

# Update account balance

def update\_balance(account\_id, new\_balance):

conn = connect\_to\_database()

if conn:

cursor = conn.cursor()

try:

query = "UPDATE accounts SET balance = %s WHERE account\_id = %s"

cursor.execute(query, (new\_balance, account\_id))

conn.commit()

if cursor.rowcount:

print("Balance updated successfully!")

else:

print("Account not found.")

except mysql.connector.Error as e:

print(f"Error: {e}")

finally:

cursor.close()

conn.close()

# Delete an account

def delete\_account(account\_id):

conn = connect\_to\_database()

if conn:

cursor = conn.cursor()

try:

query = "DELETE FROM accounts WHERE account\_id = %s"

cursor.execute(query, (account\_id,))

conn.commit()

if cursor.rowcount:

print("Account deleted successfully!")

else:

print("Account not found.")

except mysql.connector.Error as e:

print(f"Error: {e}")

finally:

cursor.close()

conn.close()

# Menu for the Bank Management System

def bank\_menu():

while True:

print("\n--- Bank Management System ---")

print("1. Create Account")

print("2. View Account")

print("3. Update Balance")

print("4. Delete Account")

print("5. Exit")

choice = input("Enter your choice: ").strip()

if choice == "1":

name = input("Enter name: ").strip()

email = input("Enter email: ").strip()

phone = input("Enter phone: ").strip()

try:

balance = float(input("Enter initial balance: ").strip())

create\_account(name, email, phone, balance)

except ValueError:

print("Invalid balance. Please enter a number.")

elif choice == "2":

try:

account\_id = int(input("Enter account ID: ").strip())

view\_account(account\_id)

except ValueError:

print("Invalid account ID. Please enter a number.")

elif choice == "3":

try:

account\_id = int(input("Enter account ID: ").strip())

new\_balance = float(input("Enter new balance: ").strip())

update\_balance(account\_id, new\_balance)

except ValueError:

print("Invalid input. Please enter valid numbers.")

elif choice == "4":

try:

account\_id = int(input("Enter account ID: ").strip())

delete\_account(account\_id)

except ValueError:

print("Invalid account ID. Please enter a number.")

elif choice == "5":

print("Exiting... Goodbye!")

break

else:

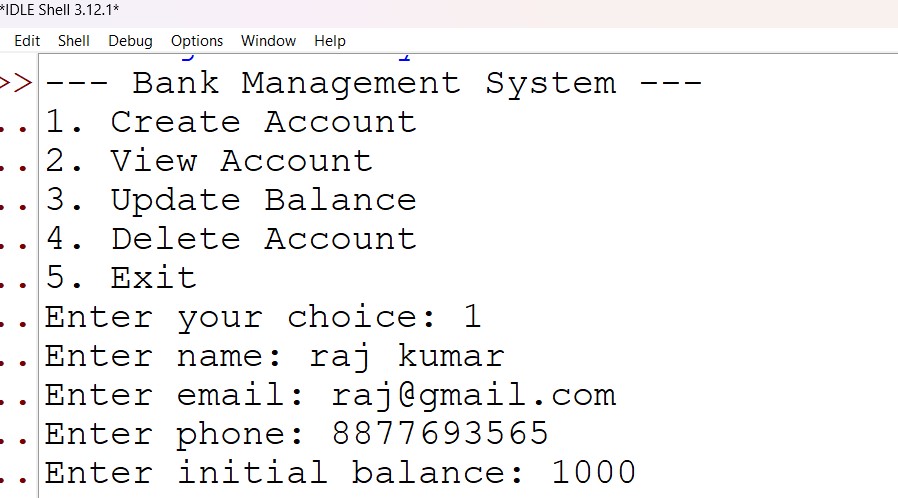
print("Invalid choice. Please try again.")

# Run the menu

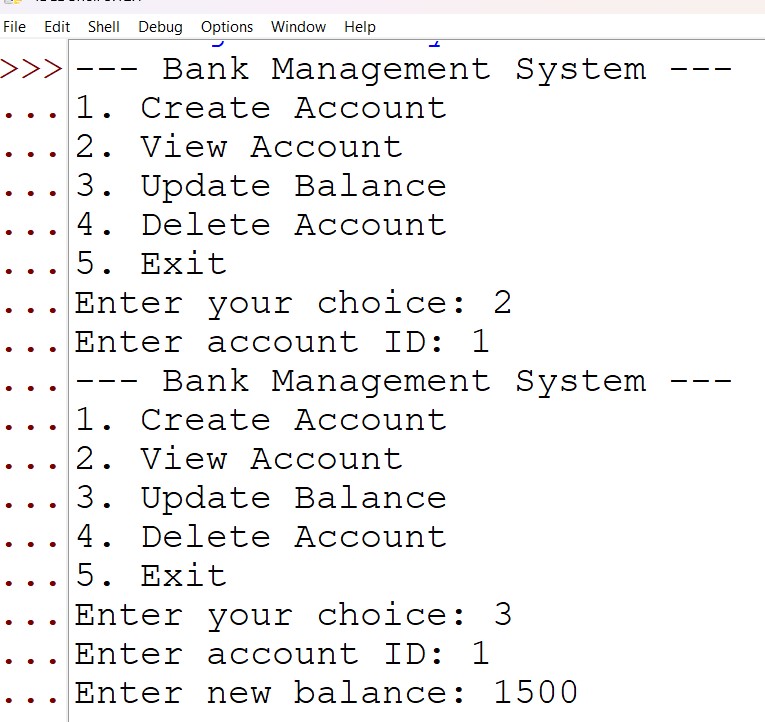
if \_\_name\_\_ == "\_\_main\_\_":

bank\_menu()

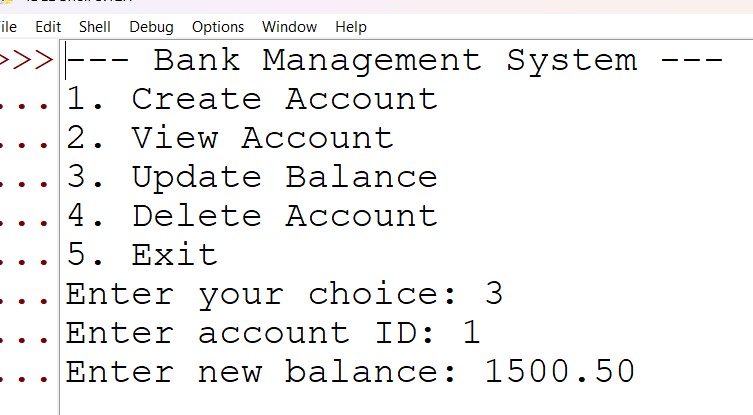
**OutPut**

**Create Accouunt Option 1**

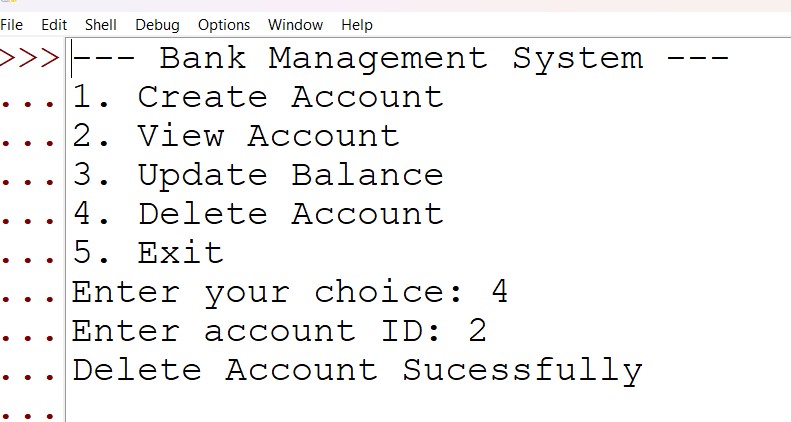
**View Account** Option 2.



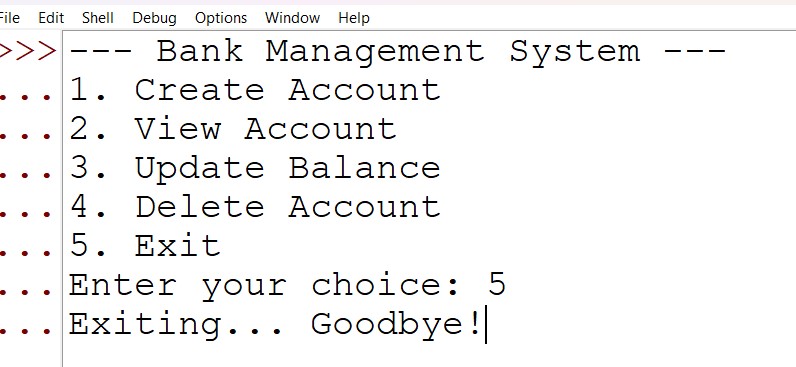
**Update Balance**  Option 3

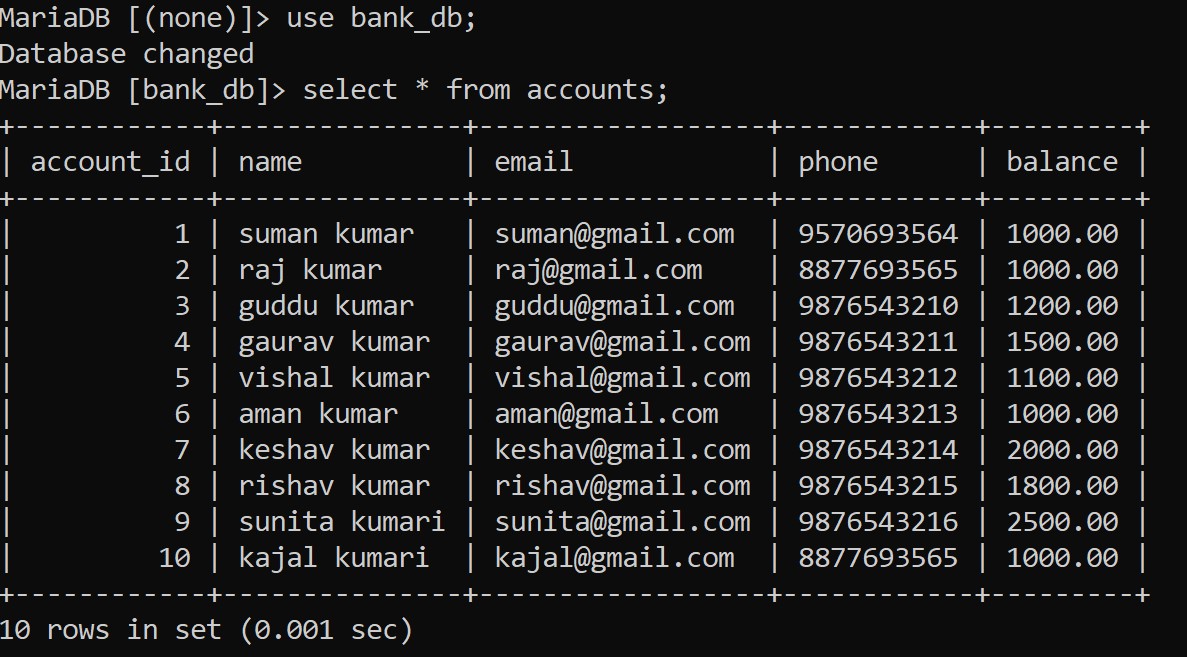


**Delete Account**  Option 4



**Exit**  option 4



**List Off All Data**