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)

```
+------------+---------------+------+-----+---------+----------------+
| Field      | Type          | Null | Key | Default | Extra          |
+------------+---------------+------+-----+---------+----------------+
| account_id | int(11)       | NO   | PRI | NULL    | auto_increment |
| name       | varchar(100)  | NO   |     | NULL    |                |
| email      | varchar(100)  | NO   |     | NULL    |                |
| phone      | varchar(15)   | YES  |     | NULL    |                |
| balance    | decimal(10,2) | NO   |     | NULL    |                |
+------------+---------------+------+-----+---------+----------------+
```

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:
```

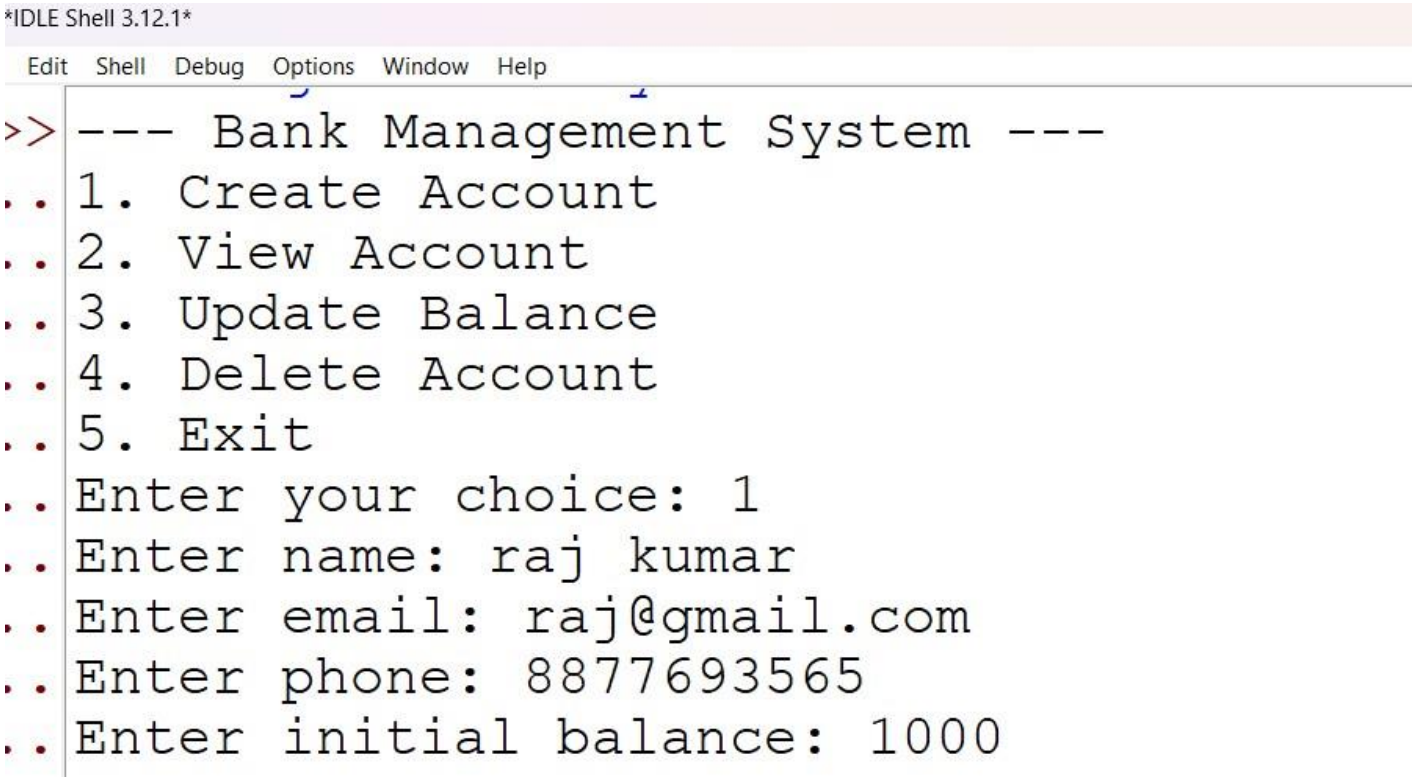
```
            print("Balance updated successfully!")
        else:
            print("Account not found.")
    except mysql.connector.Error as e:
        print(f"Error: {e}")
    finally:
        cursor.close()
        conn.close()

# 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



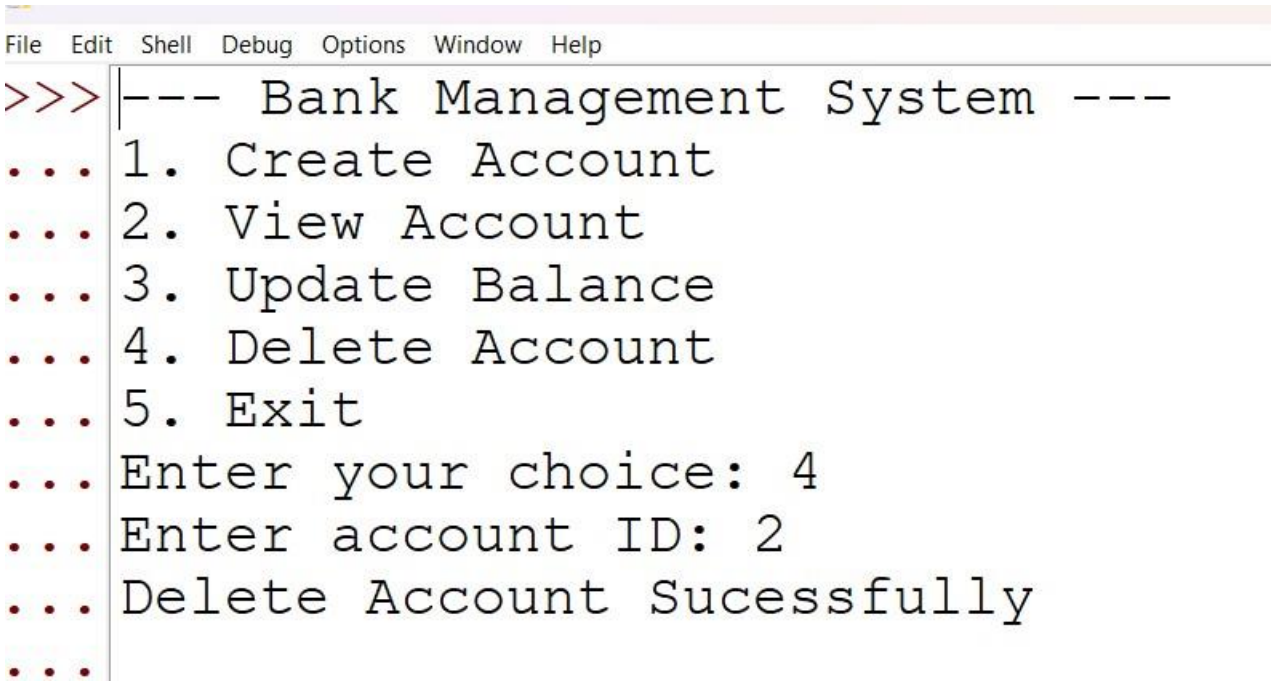
View Account Option 2.

```
File  Edit  Shell  Debug  Options  Window  Help
>>> --- Bank Management System ---
... 1. Create Account
... 2. View Account
... 3. Update Balance
... 4. Delete Account
... 5. Exit
... Enter your choice: 2
... Enter account ID: 1
... --- Bank Management System ---
... 1. Create Account
... 2. View Account
... 3. Update Balance
... 4. Delete Account
... 5. Exit
... Enter your choice: 3
... Enter account ID: 1
... Enter new balance: 1500
```

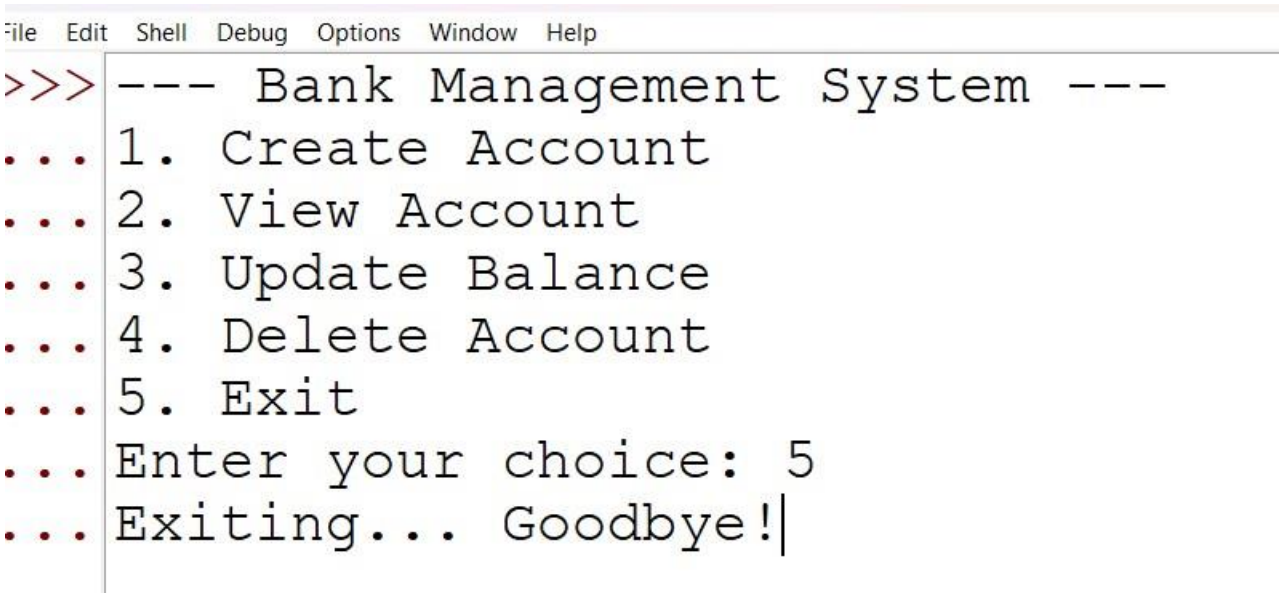
Update Balance Option 3

```
ile  Edit  Shell  Debug  Options  Window  Help
>>> --- Bank Management System ---
... 1. Create Account
... 2. View Account
... 3. Update Balance
... 4. Delete Account
... 5. Exit
... Enter your choice: 3
... Enter account ID: 1
... Enter new balance: 1500.50
```

Delete Account Option 4



Exit option 4



List Off All Data

```
MariaDB [(none)]> use bank_db;
Database changed
MariaDB [bank_db]> select * from accounts;
+------------+---------------+------------------+------------+---------+
| account_id | name          | email            | phone      | balance |
+------------+---------------+------------------+------------+---------+
|          1 | suman kumar   | suman@gmail.com  | 9570693564 | 1000.00 |
|          2 | raj kumar     | raj@gmail.com    | 8877693565 | 1000.00 |
|          3 | guddu kumar   | guddu@gmail.com  | 9876543210 | 1200.00 |
|          4 | gaurav kumar  | gaurav@gmail.com | 9876543211 | 1500.00 |
|          5 | vishal kumar  | vishal@gmail.com | 9876543212 | 1100.00 |
|          6 | aman kumar    | aman@gmail.com   | 9876543213 | 1000.00 |
|          7 | keshav kumar  | keshav@gmail.com | 9876543214 | 2000.00 |
|          8 | rishav kumar  | rishav@gmail.com | 9876543215 | 1800.00 |
|          9 | sunita kumari | sunita@gmail.com | 9876543216 | 2500.00 |
|         10 | kajal kumari  | kajal@gmail.com  | 8877693565 | 1000.00 |
+------------+---------------+------------------+------------+---------+
10 rows in set (0.001 sec)
```