COP4710 Final Project Report - Great Home Loans LOS

Vraj Patel, Daishak Patel, Harshit Suri

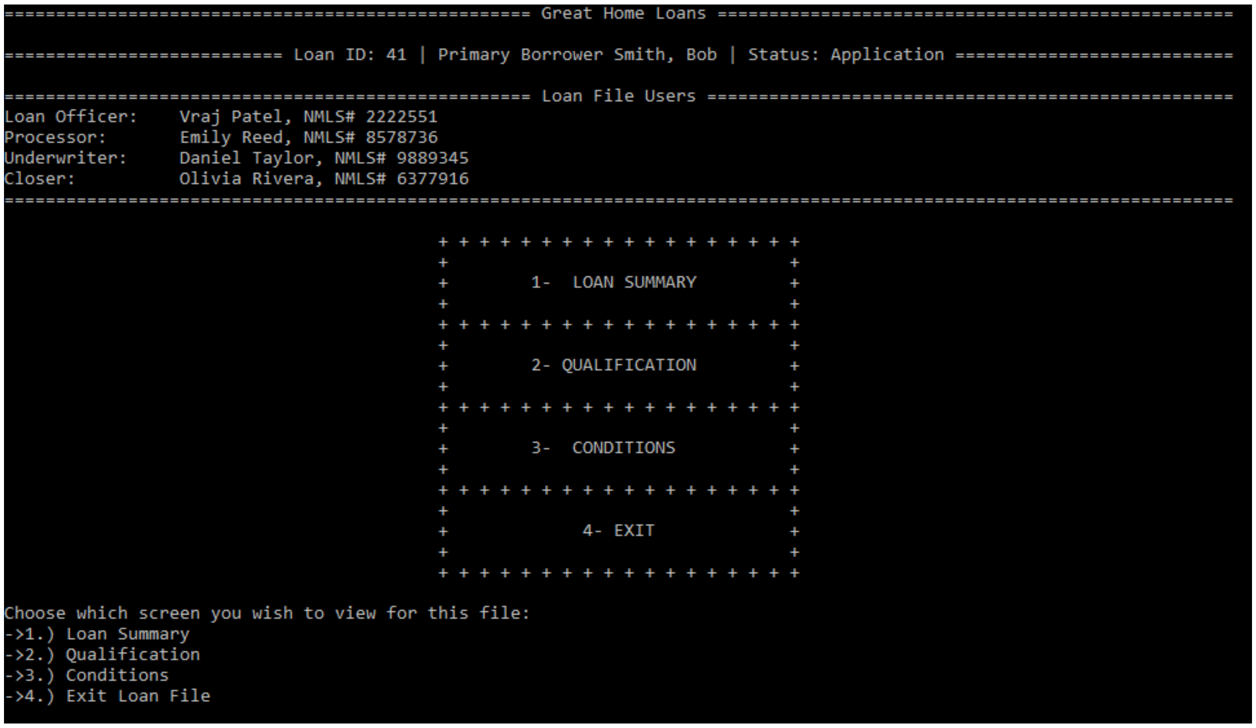
Project Description

This project presents the "Great Home Loans LOS," a dynamic and comprehensive console-based loan origination software designed to manage the mortgage process from application to closing efficiently. It automates data collection, tracking, and processing, offering features tailored to different user roles involved in loan origination.

Overview of Implemented Features

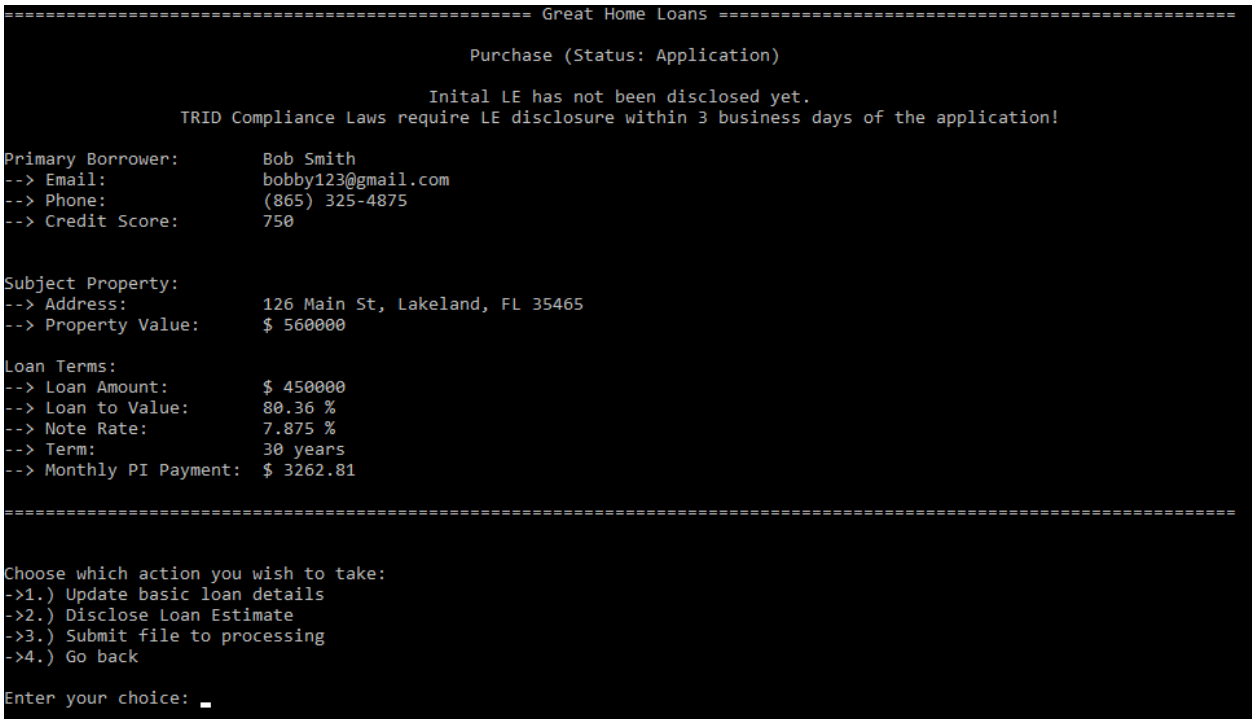
- **User Authentication:** Secure login and logout capabilities for different user roles ensure that only authorized personnel can operate within their designated areas of the system. In the LOS, there are four main types of users/roles – loan officers who take the applications and bring in the business, the processors, who work on conditions, the underwriters who approve the loans, and the closers who work on sending out the closing disclosure with final loan figures. All mortgage professionals are given a unique identifies by the Nationwide Mortgage Licensing System call their NMLS number, which we made the primary key for the users table.

- **Loan Management:** Comprehensive management capabilities allow users to create, search, view, and update loans through the system. Features are tailored to the role of the user, enhancing both security and efficiency.



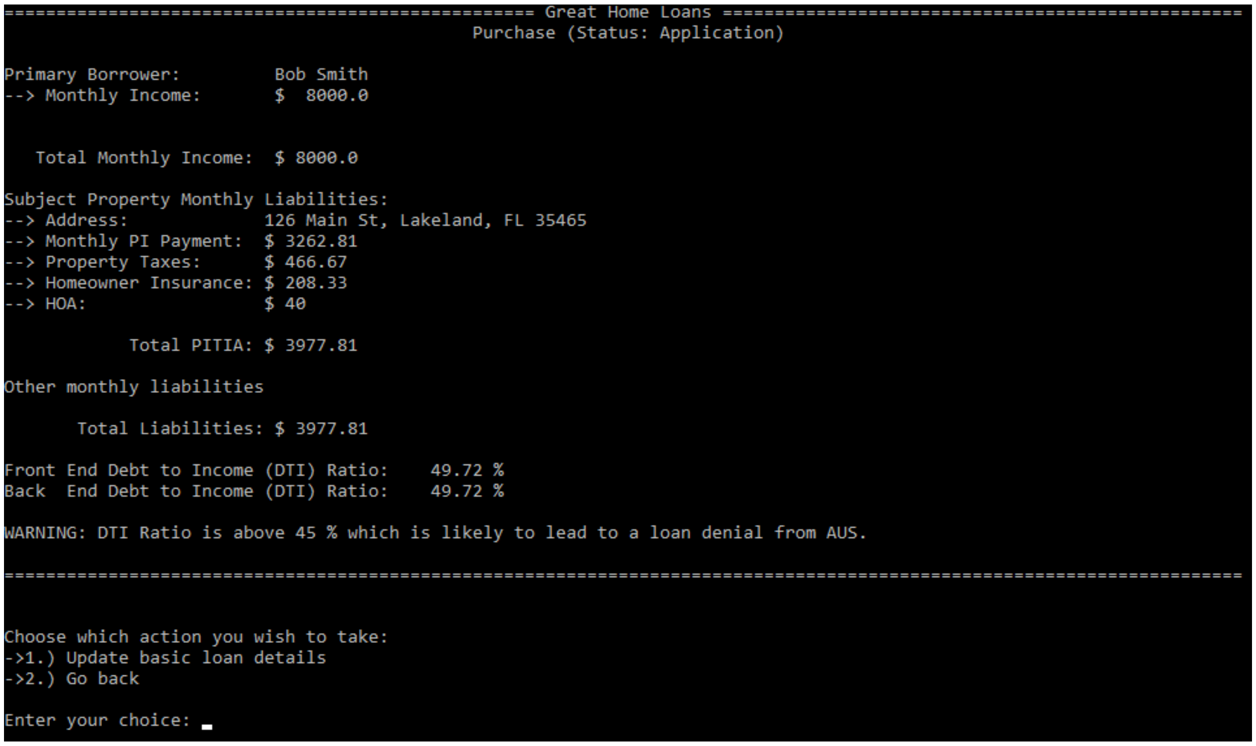
```
============================================= Great Home Loans ============================================

========================= Loan ID: 41 | Primary Borrower Smith, Bob | Status: Application =================

============================================= Loan File Users =============================================
Loan Officer:    Vraj Patel, NMLS# 2222551
Processor:       Emily Reed, NMLS# 8578736
Underwriter:     Daniel Taylor, NMLS# 9889345
Closer:          Olivia Rivera, NMLS# 6377916
===========================================================================================================

                                  + + + + + + + + + + + + + + + + + +
                                  +                                 +
                                  +        1-  LOAN SUMMARY         +
                                  +                                 +
                                  + + + + + + + + + + + + + + + + + +
                                  +                                 +
                                  +        2- QUALIFICATION         +
                                  +                                 +
                                  + + + + + + + + + + + + + + + + + +
                                  +                                 +
                                  +        3-  CONDITIONS           +
                                  +                                 +
                                  + + + + + + + + + + + + + + + + + +
                                  +                                 +
                                  +            4- EXIT              +
                                  +                                 +
                                  + + + + + + + + + + + + + + + + + +

Choose which screen you wish to view for this file:
->1.) Loan Summary
->2.) Qualification
->3.) Conditions
->4.) Exit Loan File
```

Main screen for a loan file



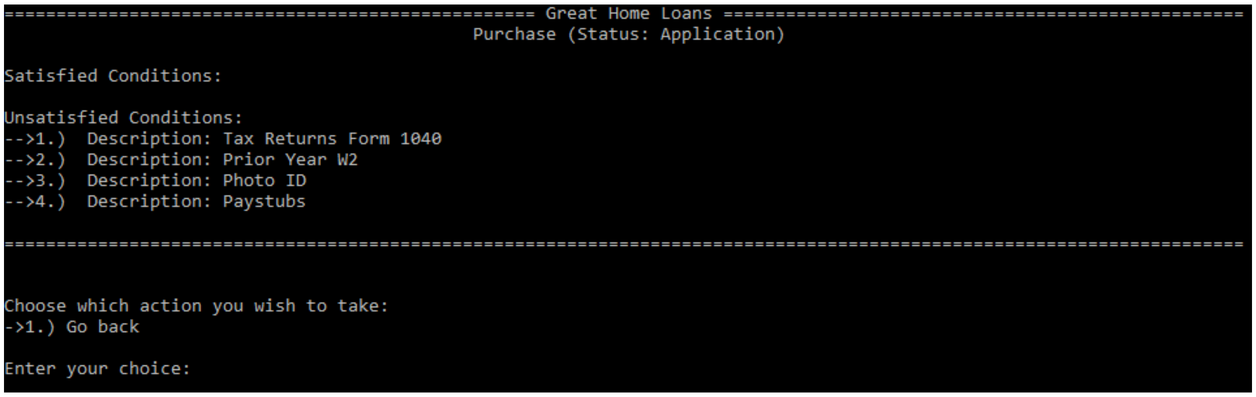
```
================================================== Great Home Loans ==================================================

                                              Purchase (Status: Application)

                                          Inital LE has not been disclosed yet.
                 TRID Compliance Laws require LE disclosure within 3 business days of the application!

Primary Borrower:        Bob Smith
--> Email:               bobby123@gmail.com
--> Phone:               (865) 325-4875
--> Credit Score:        750


Subject Property:
--> Address:             126 Main St, Lakeland, FL 35465
--> Property Value:      $ 560000

Loan Terms:
--> Loan Amount:         $ 450000
--> Loan to Value:       80.36 %
--> Note Rate:           7.875 %
--> Term:                30 years
--> Monthly PI Payment:  $ 3262.81

======================================================================================================================


Choose which action you wish to take:
->1.) Update basic loan details
->2.) Disclose Loan Estimate
->3.) Submit file to processing
->4.) Go back

Enter your choice:
```

Loan Summary Screen



```
================================================== Great Home Loans ==================================================
                                              Purchase (Status: Application)

Primary Borrower:         Bob Smith
--> Monthly Income:       $  8000.0


   Total Monthly Income:  $ 8000.0

Subject Property Monthly Liabilities:
--> Address:             126 Main St, Lakeland, FL 35465
--> Monthly PI Payment:  $ 3262.81
--> Property Taxes:      $ 466.67
--> Homeowner Insurance: $ 208.33
--> HOA:                 $ 40

            Total PITIA: $ 3977.81

Other monthly liabilities

       Total Liabilities: $ 3977.81

Front End Debt to Income (DTI) Ratio:    49.72 %
Back  End Debt to Income (DTI) Ratio:    49.72 %

WARNING: DTI Ratio is above 45 % which is likely to lead to a loan denial from AUS.

======================================================================================================================


Choose which action you wish to take:
->1.) Update basic loan details
->2.) Go back

Enter your choice:
```

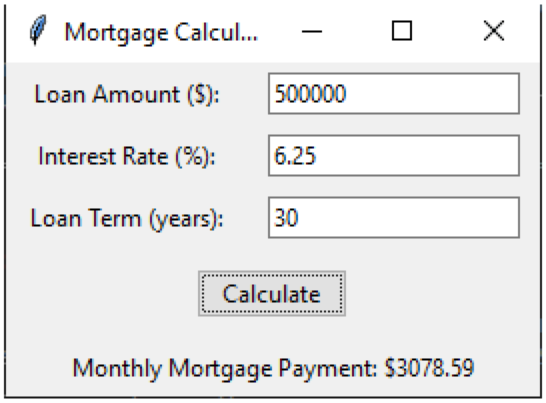
Qualification Screen



```
================================================== Great Home Loans ==================================================
                                          Purchase (Status: Application)

Satisfied Conditions:

Unsatisfied Conditions:
-->1.)  Description: Tax Returns Form 1040
-->2.)  Description: Prior Year W2
-->3.)  Description: Photo ID
-->4.)  Description: Paystubs

======================================================================================================================


Choose which action you wish to take:
->1.) Go back

Enter your choice:
```

Conditions Screen

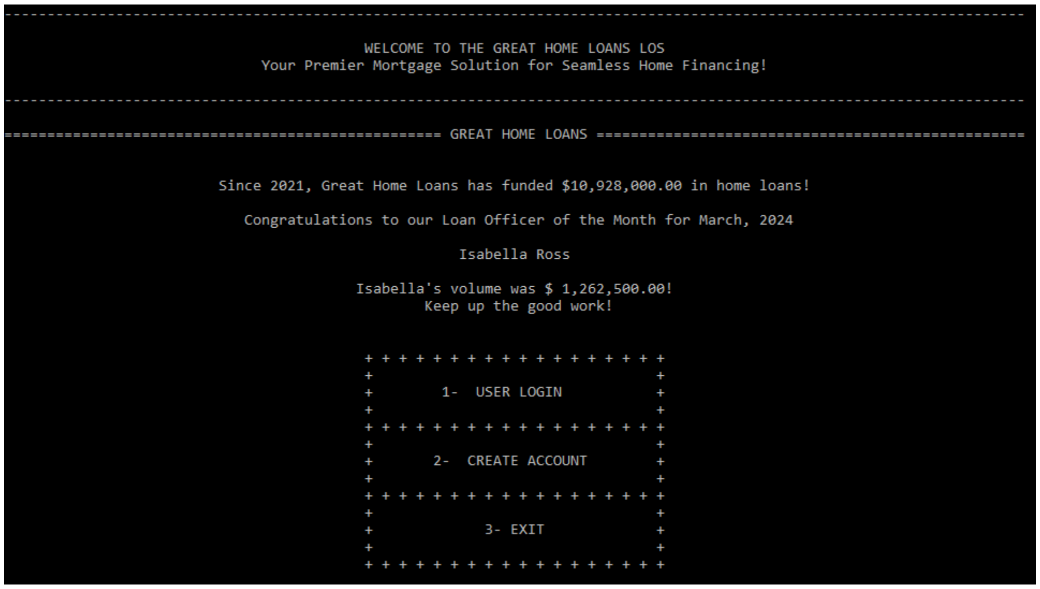
- **Mortgage Calculation:** A built in mortgage calculator helps users quickly calculate loan parameters and payment plans.



```
Mortgage Calcul...

Loan Amount ($):     500000
Interest Rate (%):   6.25
Loan Term (years):   30

          Calculate

Monthly Mortgage Payment: $3078.59
```

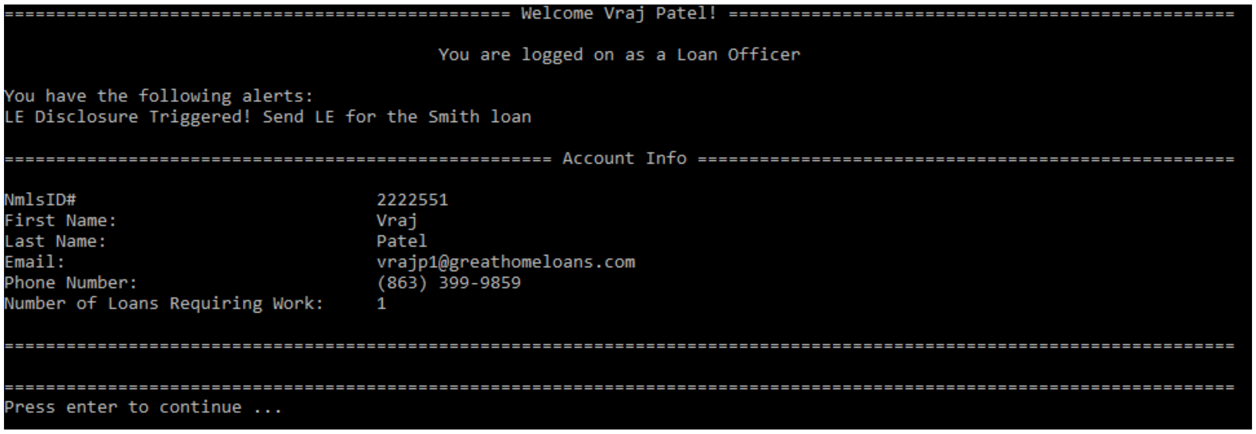
- **Document Management:** Users can simulate the management and tracking of the necessary documents attached to each loan application.

- **Dynamic Database Views for Home Screen Stats:** Utilization of a database view and assigning a user with limited privileges to dynamically generate real time statistics for the home screen such as total amount funded by the company and the Loan Officer of the month.



```
------------------------------------------------------------------------------------------------------------------------

                                            WELCOME TO THE GREAT HOME LOANS LOS
                               Your Premier Mortgage Solution for Seamless Home Financing!

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

================================================ GREAT HOME LOANS ================================================


                        Since 2021, Great Home Loans has funded $10,928,000.00 in home loans!

                           Congratulations to our Loan Officer of the Month for March, 2024

                                                     Isabella Ross

                                        Isabella's volume was $ 1,262,500.00!
                                                Keep up the good work!


                                          + + + + + + + + + + + + + + + + + +
                                          +                                 +
                                          +         1-  USER LOGIN          +
                                          +                                 +
                                          + + + + + + + + + + + + + + + + + +
                                          +                                 +
                                          +        2-  CREATE ACCOUNT       +
                                          +                                 +
                                          + + + + + + + + + + + + + + + + + +
                                          +                                 +
                                          +             3- EXIT             +
                                          +                                 +
                                          + + + + + + + + + + + + + + + + + +
```

- **Database Triggers:** Implementation of a database trigger to ensure compliance with disclosure laws. Specifically, the Real Estate Settlement and Procedures Act, RESPA, requires that a company send out the initial Loan Estimate within three business days of the application. The trigger implemented calls a stored procedure that checks if an application has been received (i.e. six key pieces of information have been received) and then alerts the loan officer to send out the initial Loan Estimate.



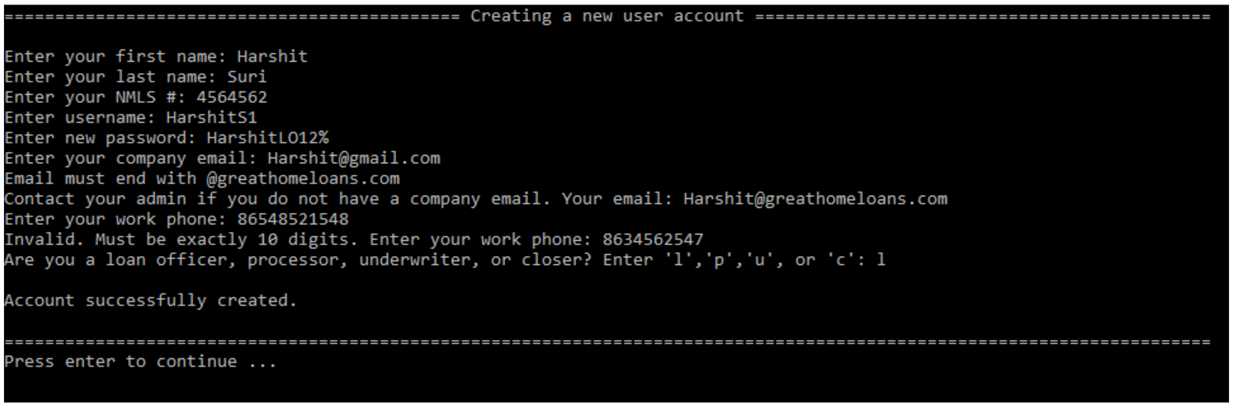
```
================================================= Welcome Vraj Patel! ================================================

                                              You are logged on as a Loan Officer

You have the following alerts:
LE Disclosure Triggered! Send LE for the Smith loan

===================================================== Account Info =====================================================

NmlsID#                             2222551
First Name:                         Vraj
Last Name:                          Patel
Email:                              vrajp1@greathomeloans.com
Phone Number:                       (863) 399-9859
Number of Loans Requiring Work:     1

========================================================================================================================

========================================================================================================================
Press enter to continue ...
```

- **Clustered Indexing for Query Optimization:** In the development of the Great Home Loans Loan Origination Software (LOS), an advanced feature was implemented to enhance database performance: a clustered index was added to the `borrowers` table on the attribute `lname`. This indexing strategy was chosen specifically because queries within the LOS frequently condition on the borrower's last name. By organizing the table data directly on disk based on last names, the clustered index significantly optimizes these types of queries, leading to faster search and retrieval times, and overall improved system efficiency for operations involving borrower information.

- **User Creation and RoleBased Access Control:**

    Systematic User Creation: Incorporating SQL commands to check the existence of user details (such as NMLS ID and email) and to insert new user data ensures that each user profile is unique and secure.
    RoleBased Privileges: The system defines specific actions that each user type can perform, which is enforced through database permissions and application logic. For example:

        Loan Officers can create loans and view loans in the 'Application' status.
        Processors handle loans in the 'Processing' status
        Underwriters can approve and clear conditions in the ‘Processing’ status
        Closers can send closing disclosures for loans that are ‘Clear to Close’



```
========================================== Creating a new user account ==========================================

Enter your first name: Harshit
Enter your last name: Suri
Enter your NMLS #: 4564562
Enter username: HarshitS1
Enter new password: HarshitLO12%
Enter your company email: Harshit@gmail.com
Email must end with @greathomeloans.com
Contact your admin if you do not have a company email. Your email: Harshit@greathomeloans.com
Enter your work phone: 86548521548
Invalid. Must be exactly 10 digits. Enter your work phone: 8634562547
Are you a loan officer, processor, underwriter, or closer? Enter 'l','p','u', or 'c': l

Account successfully created.

=================================================================================================================
Press enter to continue ...
```

SQL Tables:-

6 SQL Tables (Relations) and 2 Database Views have been implemented.

- **Loans Table:**
  The `Loans` table tracks all loan applications and their statuses. It includes comprehensive information like loan ID, associated borrower and co-borrower IDs, IDs for various roles involved in processing the loan (officer, processor, underwriter, closer), loan type, application and closure dates, associated property, loan amount, interest rate, term, and statuses of critical loan documents (loan estimate and closing disclosure).

  SQL Query: *CREATE TABLE loans (id INT, borrower INT, coborrower INT, officer INT, processor INT, underwriter INT, closer INT, status VARCHAR(255), type VARCHAR(255), appDate DATE, closeDate DATE, property INT, loanAmount INT, rate DECIMAL(5, 3), term INT, loanEstimateSent BOOLEAN, closingDisclosureSent BOOLEAN, PRIMARY KEY (id), FOREIGN KEY (borrower) REFERENCES borrowers(id), FOREIGN KEY (coborrower) REFERENCES borrowers(id), FOREIGN KEY (property) REFERENCES properties(id));*

- **Properties Table:**
  This table stores details about properties, including their unique identifier, valuation, location (street, city, state, zip code), and associated costs such as taxes, homeowners insurance (HOI), and homeowners association (HOA) fees.

  SQL Query: *CREATE TABLE properties (id INT, value INT, street VARCHAR(255), state CHAR(2), zip INT, taxes INT, hoi INT, hoa INT, city VARCHAR(255), PRIMARY KEY (id));*

- **Users Table:**
  The `Users` table contains information about users of a system, potentially loan officers or bank employees. It includes their national mortgage licensing system (NMLS) ID, personal details (first name, last name, email, phone), and account details (alerts, type of user, username, password).

SQL Query: *CREATE TABLE users (NMLS INT, fname VARCHAR(255), lname VARCHAR(255), email VARCHAR(255), phone BIGINT, alerts VARCHAR(255), type VARCHAR(255), username VARCHAR(255), password VARCHAR(255), PRIMARY KEY (NMLS));*

- **Borrowers Table:**
  This table keeps track of borrowers, including a unique ID, their first and last names, contact details (email and phone), and their credit score. It's used to manage customer information relevant to loan processing.

  SQL Query: *CREATE TABLE borrowers (id INT, fname VARCHAR(255), lname VARCHAR(255), email VARCHAR(255), credit INT, phone BIGINT, PRIMARY KEY (id));*

- **Conditions Table:**
  The `Conditions` table lists conditions associated with loans, such as required documents or criteria that need to be met. Each record includes an ID, the associated loan ID, a description of the condition, and whether the condition has been satisfied.

  SQL Query: *CREATE TABLE conditions (id INT, loan INT, descr VARCHAR(255), satisfied BOOLEAN, PRIMARY KEY (id), FOREIGN KEY (loan) REFERENCES loans(id));*

- **Liabilities Table:**
  This table records liabilities related to loans, detailing each liability's ID, the associated loan ID, the creditor, a description of the liability, the outstanding balance, the monthly payment amount, and whether the liability has been omitted from certain calculations or considerations.

  SQL Query: *CREATE TABLE liabilities (id INT, loan INT, creditor VARCHAR(255), descr VARCHAR(255), balance INT, amount DECIMAL(10, 2), omitted BOOLEAN, PRIMARY KEY (id), FOREIGN KEY (loan) REFERENCES loans(id));*

- **Stats View:**
  This view derived from the loans and users table above pulls the loan volumes funded and loan officer’s names and the home_screen user has limited privileges to SELECT from this database view.

  SQL Query: *CREATE VIEW Stats AS SELECT U.NMLS, U.fname, U.lname, SUM(L.loanamount) AS volume, DATE_PART('year',L.closedate) AS year, DATE_PART('month',L.closedate) AS month FROM Users U JOIN Loans L ON U.NMLS = L.officer WHERE L.status = 'Closed' GROUP BY U.NMLS, DATE_PART('year',L.closedate), DATE_PART('month',L.closedate);*

- **LE_Info View:**
  This view pulls the six key pieces of information from the borrower that constitutes a loan application – borrower's name, subject property address, borrower's income, borrowers' credit score, estimated property value, and loan amount sought. There is a trigger on this view that calls a stored procedure to alert the Loan Officer if a LE disclosure has been triggered.

  SQL Query: *CREATE VIEW LE_Info AS SELECT L.id, L.officer, B.lname, B.credit, B.income, P.value as property_value, P.street, L.loanamount as loan_amount, L.loanestimatesent FROM Loans L, Properties P, Borrowers B WHERE L.property = P.id AND L.borrower = B.id;*

Main SQL Queries:-

- **User Login Verification:**
  *SELECT * FROM users WHERE username = %s AND password = %s;*
  This query checks user credentials during the login process, ensuring secure access to the system.

- **Stored Procedure Definition for Loan Estimate Update Trigger:**
  ```
  CREATE OR REPLACE FUNCTION trigger_LE_function()
  RETURNS TRIGGER AS $$
  BEGIN
     RAISE NOTICE 'Trigger was activated for LE_info';
     -- Check if all required attributes are populated and LE not sent
     IF NEW.lname IS NOT NULL AND NEW.credit IS NOT NULL AND NEW.income IS
  NOT NULL AND NEW.property_value IS NOT NULL AND NEW.street IS NOT NULL
  AND NEW.loan_amount IS NOT NULL AND NEW.loanestimatesent = FALSE THEN
    -- Update alerts attribute in the Users table for the corresponding officer
    UPDATE Users
    SET alerts = 'LE Disclosure Triggered! Send LE for the ' || NEW.lname || ' loan'
  WHERE nmls = NEW.officer;
    END IF;
    RETURN NEW;
  END;
  $$ LANGUAGE plpgsql;
  ```
  This function (`trigger_LE_function`) is designed to act upon updates made in the `LE_Info` view. It checks if all necessary fields are filled and if the loan estimate has not been sent. If conditions are met, it updates the `alerts` field in the `Users` table to prompt the loan officer to send a loan estimate

- **Trigger Creation for Handling Updates on LE Information:**
  CREATE TRIGGER update_alerts_on_LE_trigger INSTEAD OF UPDATE ON LE_Info FOR EACH ROW EXECUTE FUNCTION trigger_LE_function();

This trigger (`update_alerts_on_LE_trigger`) fires on any update attempt on the `LE_Info` view. It uses the `trigger_LE_function` to perform actions defined in the function, ensuring that loan officers are alerted to send loan estimates when all prerequisites are satisfied.

- **Fetch Loans by Loan Officer:**
  SELECT L.id, B.lname, B.fname, L.status FROM Loans L JOIN Borrowers B ON L.borrower = B.id WHERE L.officer = %s AND B.lname LIKE %s AND L.status = 'Application';
  Loan officers use this query to retrieve loans they are managing, filtered by borrower's last name and loan status.

- **SQL View Creation for Loan Estimate (LE) Information:**
  CREATE VIEW LE_Info AS SELECT L.id, L.officer, B.lname, B.credit, B.income, P.value AS property_value, P.street, L.loanamount AS loan_amount, L.loanestimatesent FROM Loans L JOIN Properties P ON L.property = P.id JOIN Borrowers B ON L.borrower = B.id;
  This SQL view (`LE_Info`) aggregates important data related to loan estimates, joining information from the `Loans`, `Properties`, and `Borrowers` tables. It simplifies access to loan-related details necessary for processing loan estimates.

- **Update Loan Details:**
  UPDATE Loans SET status = %s WHERE id = %s;
  This query is used to update the status of a loan, an essential feature that tracks progress through the loan processing stages.

- **Retrieve Loan Summary:**
  *SELECT * FROM Loans WHERE id = %s;*
  This query fetches all details related to a specific loan, crucial for the loan summary view where users can inspect every aspect of the loan.

- **Retrieve Borrower Summary:**
  *SELECT * FROM Borrower WHERE id = %s;*
  This query fetches all details related to a specific Borrower.

- **Retrieve Property Info:**
  *SELECT * FROM Borrower Properties id = %s;*
  This query fetches all details related to a specific property.

- **Loan Officer Count of Loans in Application:**
  SELECT COUNT(*) FROM Users U JOIN Loans L ON U.NMLS = L.officer WHERE U.NMLS = %s AND L.status = 'Application'

This query counts the number of loans that are currently in the "Application" stage for a specific loan officer. It helps loan officers to quickly see how many loan applications they are managing that are yet to be processed further.

- **Processor Count of Loans in Processing:**
  SELECT COUNT(*) FROM Users U JOIN Loans L ON U.NMLS = L.processor WHERE U.NMLS = %s AND L.status = 'Processing'
  For loan processors, this query determines the total number of loans they are currently processing. It's critical for processors to monitor their active tasks and manage their time effectively to ensure timely processing of all loans.

- **Underwriter Count of Loans in Processing:**
  SELECT COUNT(*) FROM Users U JOIN Loans L ON U.NMLS = L.underwriter WHERE U.NMLS = %s AND L.status = 'Processing'
  This query is similar to the one for processors but tailored for underwriters. It counts how many loans are in the processing stage and are awaiting underwriting decisions. This helps underwriters prioritize their review tasks based on the volume of pending approvals.

- **Closer Count of Loans Ready for Closing:**
  SELECT COUNT(*) FROM Users U JOIN Loans L ON U.NMLS = L.closer WHERE U.NMLS = %s AND L.status = 'Clear to Close'
  For closers, this query counts the loans that have been processed and are now "Clear to Close." It allows closers to prepare for the final steps in the loan process, ensuring all necessary documentation and approvals are in place for a smooth closing.

- **Check for Existing NMLS ID:**
  *SELECT EXISTS(SELECT * FROM USERS WHERE nmls = %s);*
  This query is used to verify whether an NMLS ID already exists in the database. It returns a boolean value (`TRUE` or `FALSE`). If `TRUE`, it indicates that the NMLS ID is already registered, preventing the duplication of user records in the system. This check is crucial during the user registration process to ensure each NMLS ID is unique.

- **Check for Existing Email:**
  *SELECT EXISTS(SELECT * FROM USERS WHERE email = %s);*
  Similar to the NMLS ID check, this query determines if an email address is already associated with an existing user account. It helps maintain unique email addresses for each user, which is essential for account recovery processes and communications.

- **Insert New User Record:**
  INSERT INTO Users VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s);
  This SQL command is used to insert a new user record into the `Users` table. It's a critical function in the user registration process, allowing new users to be added to the system with their corresponding roles and credentials. This query ensures that all

necessary user information is stored correctly in the database at the time of account creation.

Reflection

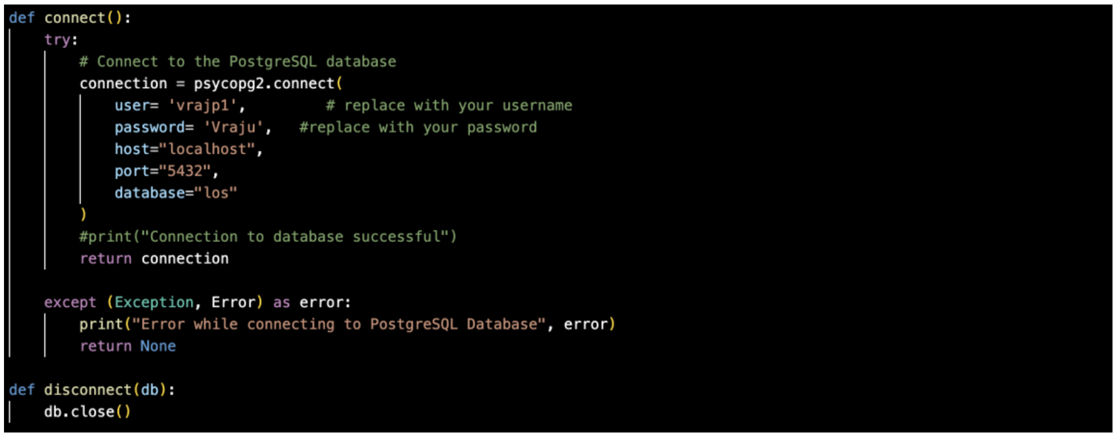
The design chosen for the final implementation is a subset of what was submitted for Milestone 1, primarily because the comprehensive initial design imagined at first with all the bells and whistles of a real LOS would have involved over 15 to 20 different entities with numerous relationships, clearly beyond the scope of this term project. Therefore, it was decided that the subset of the design chosen was sufficient to demonstrate the basic functionality and workflow that a loan file would typically go through at a mortgage company. In the end, we were able to create a sophisticated enterprise solution with six meaningful relations and numerous advanced features such as triggers and database views.

Conclusion

"Great Home Loans LOS" significantly enhances the efficiency of mortgage loan origination processes. Its robust database and application features streamline operations, reduce errors, and improve the overall user experience. The project showcases our ability to design and implement a complex system that meets real world business needs.

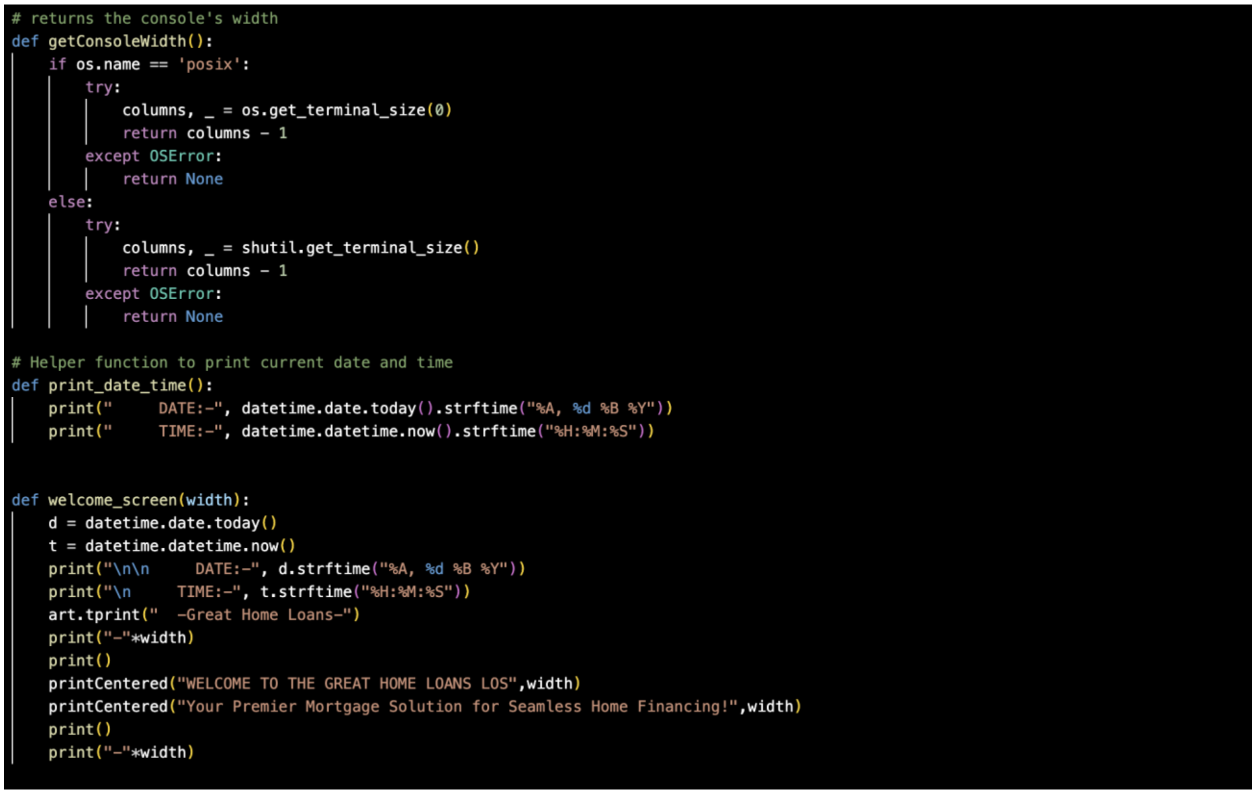
Appendix – Code Snippets

Selected Code Snippets:



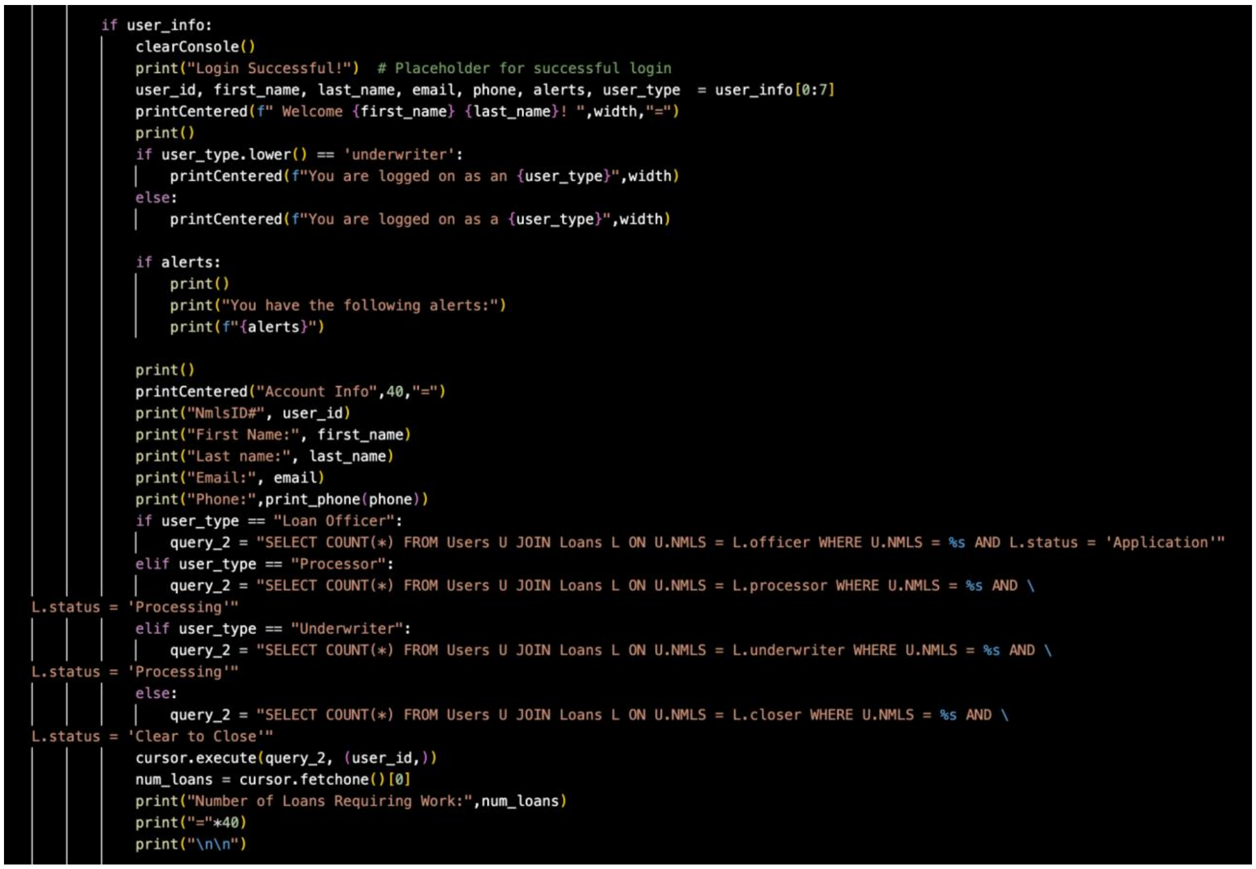
```python
def connect():
    try:
        # Connect to the PostgreSQL database
        connection = psycopg2.connect(
            user= 'vrajp1',         # replace with your username
            password= 'Vraju',   #replace with your password
            host="localhost",
            port="5432",
            database="los"
        )
        #print("Connection to database successful")
        return connection

    except (Exception, Error) as error:
        print("Error while connecting to PostgreSQL Database", error)
        return None

def disconnect(db):
    db.close()
```

Function to establish Database Connection



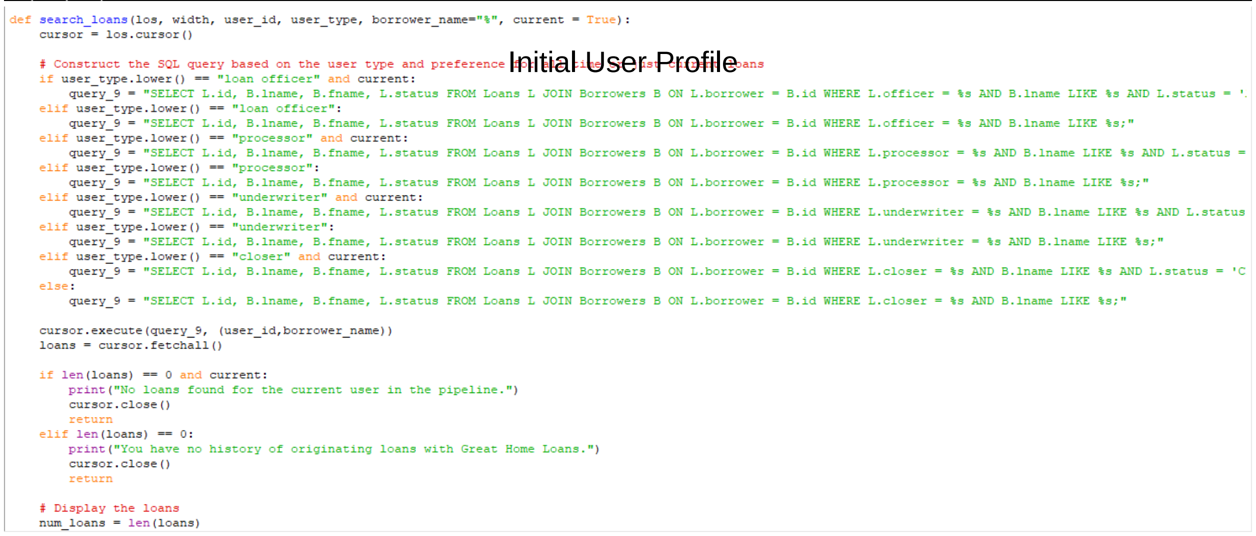
```python
# returns the console's width
def getConsoleWidth():
    if os.name == 'posix':
        try:
            columns, _ = os.get_terminal_size(0)
            return columns - 1
        except OSError:
            return None
    else:
        try:
            columns, _ = shutil.get_terminal_size()
            return columns - 1
        except OSError:
            return None

# Helper function to print current date and time
def print_date_time():
    print("     DATE:-", datetime.date.today().strftime("%A, %d %B %Y"))
    print("     TIME:-", datetime.datetime.now().strftime("%H:%M:%S"))


def welcome_screen(width):
    d = datetime.date.today()
    t = datetime.datetime.now()
    print("\n\n     DATE:-", d.strftime("%A, %d %B %Y"))
    print("\n     TIME:-", t.strftime("%H:%M:%S"))
    art.tprint("  -Great Home Loans-")
    print("-"*width)
    print()
    printCentered("WELCOME TO THE GREAT HOME LOANS LOS",width)
    printCentered("Your Premier Mortgage Solution for Seamless Home Financing!",width)
    print()
    print("-"*width)
```

Utility Functions for Printing



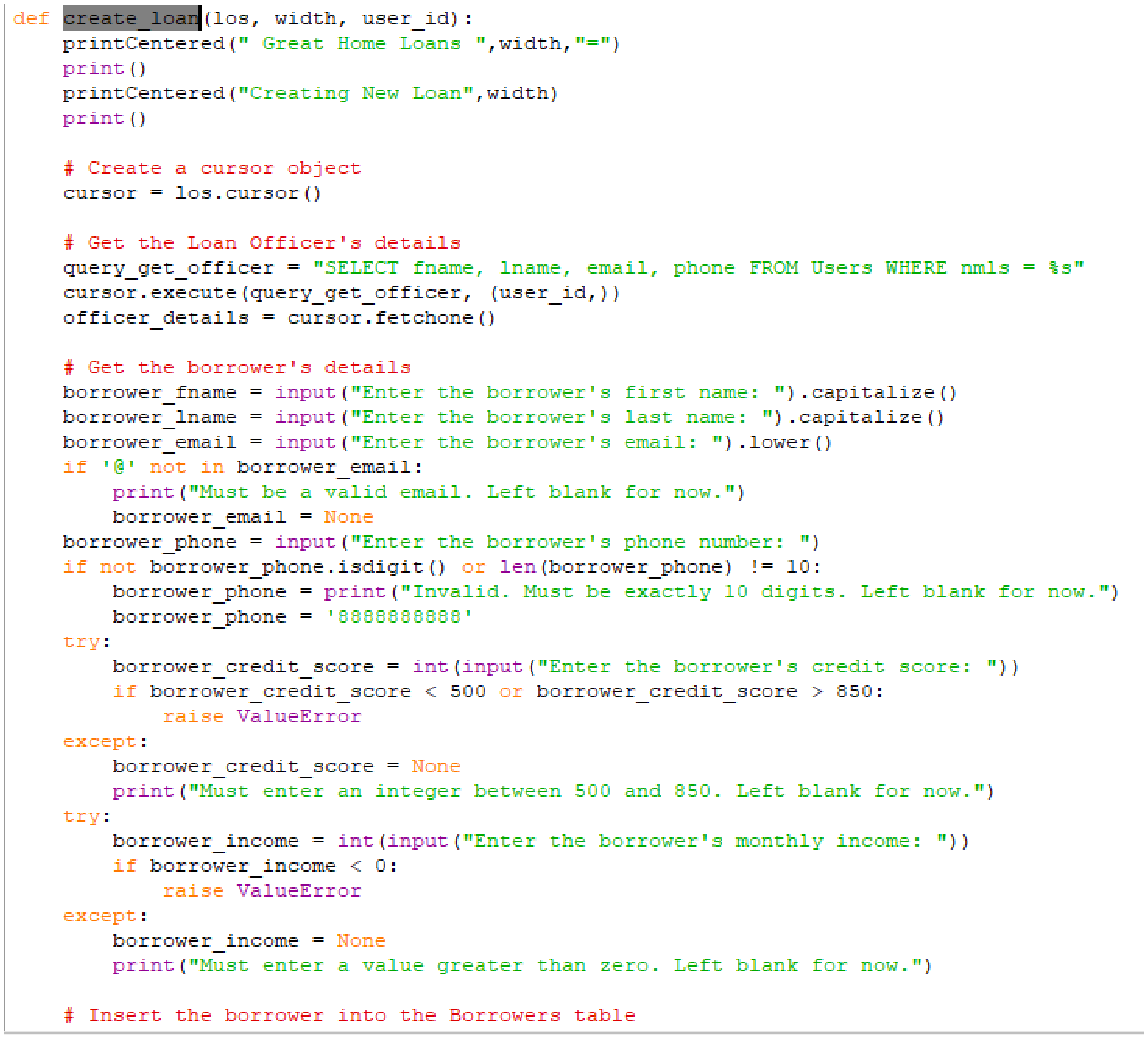
```
            if user_info:
                clearConsole()
                print("Login Successful!")  # Placeholder for successful login
                user_id, first_name, last_name, email, phone, alerts, user_type  = user_info[0:7]
                printCentered(f" Welcome {first_name} {last_name}! ",width,"=")
                print()
                if user_type.lower() == 'underwriter':
                    printCentered(f"You are logged on as an {user_type}",width)
                else:
                    printCentered(f"You are logged on as a {user_type}",width)

                if alerts:
                    print()
                    print("You have the following alerts:")
                    print(f"{alerts}")

                print()
                printCentered("Account Info",40,"=")
                print("NmlsID#", user_id)
                print("First Name:", first_name)
                print("Last name:", last_name)
                print("Email:", email)
                print("Phone:",print_phone(phone))
                if user_type == "Loan Officer":
                    query_2 = "SELECT COUNT(*) FROM Users U JOIN Loans L ON U.NMLS = L.officer WHERE U.NMLS = %s AND L.status = 'Application'"
                elif user_type == "Processor":
                    query_2 = "SELECT COUNT(*) FROM Users U JOIN Loans L ON U.NMLS = L.processor WHERE U.NMLS = %s AND \
L.status = 'Processing'"
                elif user_type == "Underwriter":
                    query_2 = "SELECT COUNT(*) FROM Users U JOIN Loans L ON U.NMLS = L.underwriter WHERE U.NMLS = %s AND \
L.status = 'Processing'"
                else:
                    query_2 = "SELECT COUNT(*) FROM Users U JOIN Loans L ON U.NMLS = L.closer WHERE U.NMLS = %s AND \
L.status = 'Clear to Close'"
                cursor.execute(query_2, (user_id,))
                num_loans = cursor.fetchone()[0]
                print("Number of Loans Requiring Work:",num_loans)
                print("="*40)
                print("\n\n")
```

Initial User Profile



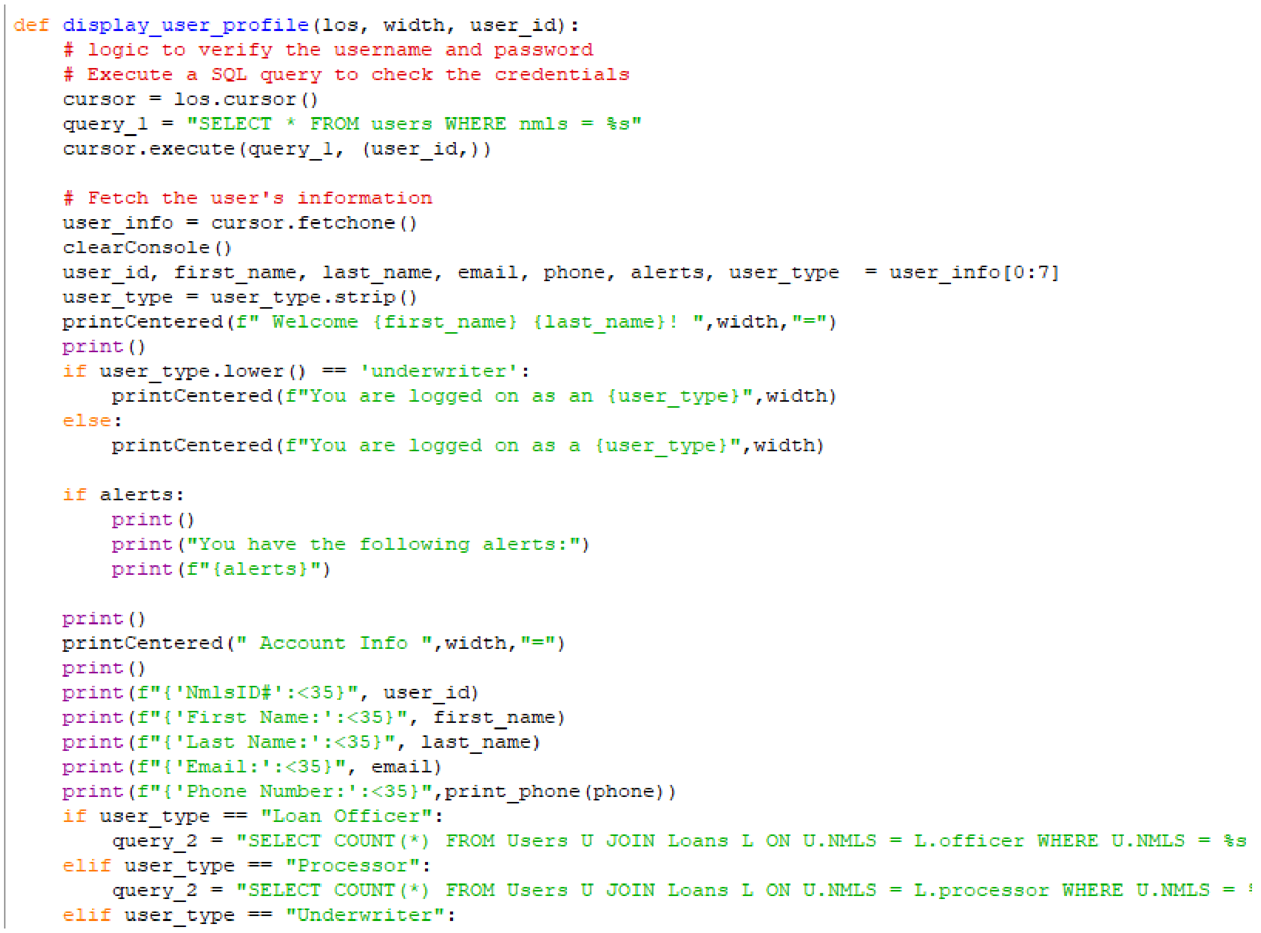
```
def search_loans(los, width, user_id, user_type, borrower_name="%", current = True):
    cursor = los.cursor()

    # Construct the SQL query based on the user type and preference for all time or just current loans
    if user_type.lower() == "loan officer" and current:
        query_9 = "SELECT L.id, B.lname, B.fname, L.status FROM Loans L JOIN Borrowers B ON L.borrower = B.id WHERE L.officer = %s AND B.lname LIKE %s AND L.status = '.
    elif user_type.lower() == "loan officer":
        query_9 = "SELECT L.id, B.lname, B.fname, L.status FROM Loans L JOIN Borrowers B ON L.borrower = B.id WHERE L.officer = %s AND B.lname LIKE %s;"
    elif user_type.lower() == "processor" and current:
        query_9 = "SELECT L.id, B.lname, B.fname, L.status FROM Loans L JOIN Borrowers B ON L.borrower = B.id WHERE L.processor = %s AND B.lname LIKE %s AND L.status =
    elif user_type.lower() == "processor":
        query_9 = "SELECT L.id, B.lname, B.fname, L.status FROM Loans L JOIN Borrowers B ON L.borrower = B.id WHERE L.processor = %s AND B.lname LIKE %s;"
    elif user_type.lower() == "underwriter" and current:
        query_9 = "SELECT L.id, B.lname, B.fname, L.status FROM Loans L JOIN Borrowers B ON L.borrower = B.id WHERE L.underwriter = %s AND B.lname LIKE %s AND L.status
    elif user_type.lower() == "underwriter":
        query_9 = "SELECT L.id, B.lname, B.fname, L.status FROM Loans L JOIN Borrowers B ON L.borrower = B.id WHERE L.underwriter = %s AND B.lname LIKE %s;"
    elif user_type.lower() == "closer" and current:
        query_9 = "SELECT L.id, B.lname, B.fname, L.status FROM Loans L JOIN Borrowers B ON L.borrower = B.id WHERE L.closer = %s AND B.lname LIKE %s AND L.status = 'C
    else:
        query_9 = "SELECT L.id, B.lname, B.fname, L.status FROM Loans L JOIN Borrowers B ON L.borrower = B.id WHERE L.closer = %s AND B.lname LIKE %s;"

    cursor.execute(query_9, (user_id,borrower_name))
    loans = cursor.fetchall()

    if len(loans) == 0 and current:
        print("No loans found for the current user in the pipeline.")
        cursor.close()
        return
    elif len(loans) == 0:
        print("You have no history of originating loans with Great Home Loans.")
        cursor.close()
        return

    # Display the loans
    num_loans = len(loans)
```

Search Loans



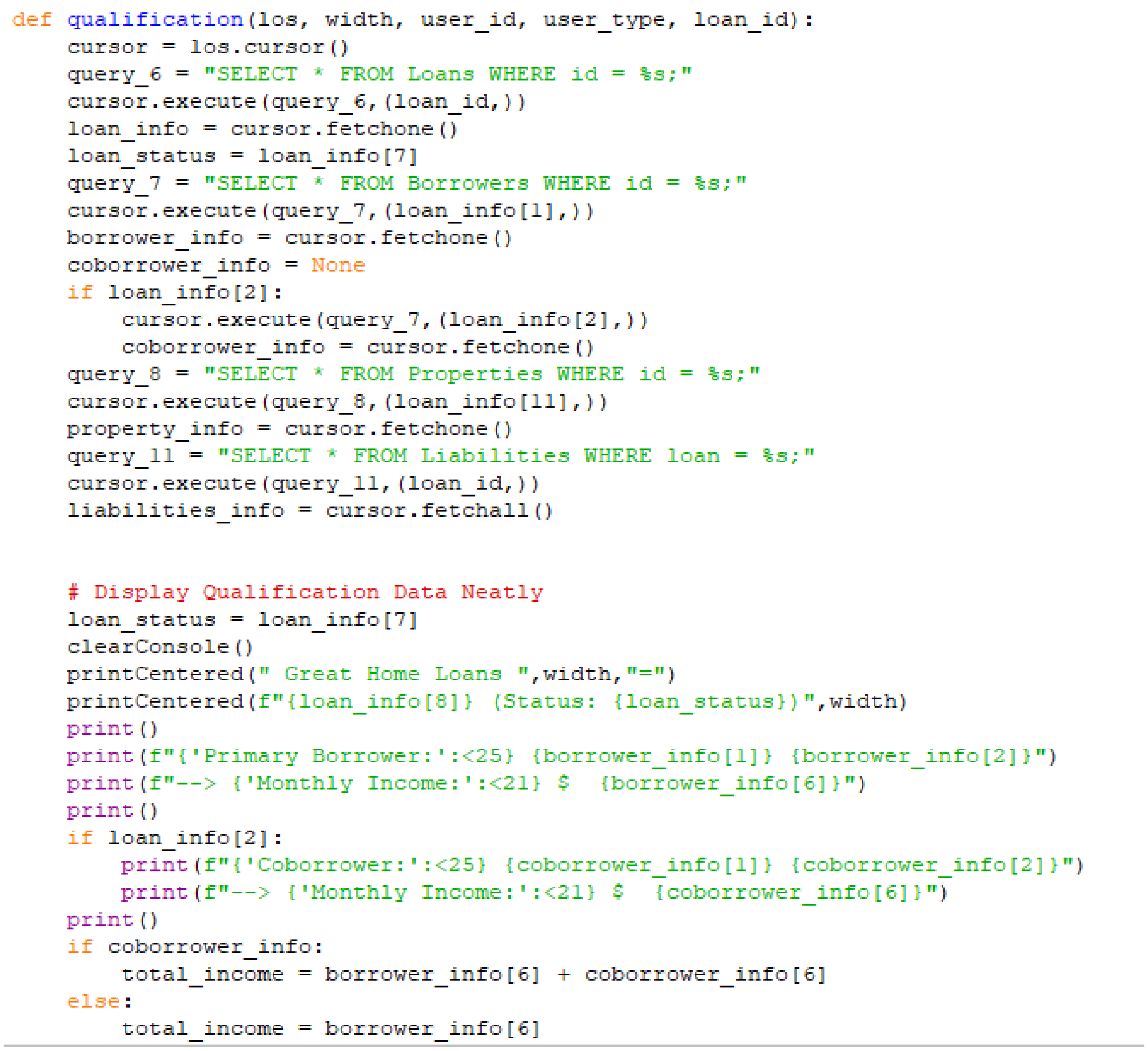
```
def create_loan(los, width, user_id):
    printCentered(" Great Home Loans ",width,"=")
    print()
    printCentered("Creating New Loan",width)
    print()

    # Create a cursor object
    cursor = los.cursor()

    # Get the Loan Officer's details
    query_get_officer = "SELECT fname, lname, email, phone FROM Users WHERE nmls = %s"
    cursor.execute(query_get_officer, (user_id,))
    officer_details = cursor.fetchone()

    # Get the borrower's details
    borrower_fname = input("Enter the borrower's first name: ").capitalize()
    borrower_lname = input("Enter the borrower's last name: ").capitalize()
    borrower_email = input("Enter the borrower's email: ").lower()
    if '@' not in borrower_email:
        print("Must be a valid email. Left blank for now.")
        borrower_email = None
    borrower_phone = input("Enter the borrower's phone number: ")
    if not borrower_phone.isdigit() or len(borrower_phone) != 10:
        borrower_phone = print("Invalid. Must be exactly 10 digits. Left blank for now.")
        borrower_phone = '8888888888'
    try:
        borrower_credit_score = int(input("Enter the borrower's credit score: "))
        if borrower_credit_score < 500 or borrower_credit_score > 850:
            raise ValueError
    except:
        borrower_credit_score = None
        print("Must enter an integer between 500 and 850. Left blank for now.")
    try:
        borrower_income = int(input("Enter the borrower's monthly income: "))
        if borrower_income < 0:
            raise ValueError
    except:
        borrower_income = None
        print("Must enter a value greater than zero. Left blank for now.")

    # Insert the borrower into the Borrowers table
```

User Profile



```python
def display_user_profile(los, width, user_id):
    # logic to verify the username and password
    # Execute a SQL query to check the credentials
    cursor = los.cursor()
    query_1 = "SELECT * FROM users WHERE nmls = %s"
    cursor.execute(query_1, (user_id,))

    # Fetch the user's information
    user_info = cursor.fetchone()
    clearConsole()
    user_id, first_name, last_name, email, phone, alerts, user_type  = user_info[0:7]
    user_type = user_type.strip()
    printCentered(f" Welcome {first_name} {last_name}! ",width,"=")
    print()
    if user_type.lower() == 'underwriter':
        printCentered(f"You are logged on as an {user_type}",width)
    else:
        printCentered(f"You are logged on as a {user_type}",width)

    if alerts:
        print()
        print("You have the following alerts:")
        print(f"{alerts}")

    print()
    printCentered(" Account Info ",width,"=")
    print()
    print(f"{'NmlsID#':<35}", user_id)
    print(f"{'First Name:':<35}", first_name)
    print(f"{'Last Name:':<35}", last_name)
    print(f"{'Email:':<35}", email)
    print(f"{'Phone Number:':<35}",print_phone(phone))
    if user_type == "Loan Officer":
        query_2 = "SELECT COUNT(*) FROM Users U JOIN Loans L ON U.NMLS = L.officer WHERE U.NMLS = %s
    elif user_type == "Processor":
        query_2 = "SELECT COUNT(*) FROM Users U JOIN Loans L ON U.NMLS = L.processor WHERE U.NMLS =
    elif user_type == "Underwriter":
```

User Profile



```python
def qualification(los, width, user_id, user_type, loan_id):
    cursor = los.cursor()
    query_6 = "SELECT * FROM Loans WHERE id = %s;"
    cursor.execute(query_6,(loan_id,))
    loan_info = cursor.fetchone()
    loan_status = loan_info[7]
    query_7 = "SELECT * FROM Borrowers WHERE id = %s;"
    cursor.execute(query_7,(loan_info[1],))
    borrower_info = cursor.fetchone()
    coborrower_info = None
    if loan_info[2]:
        cursor.execute(query_7,(loan_info[2],))
        coborrower_info = cursor.fetchone()
    query_8 = "SELECT * FROM Properties WHERE id = %s;"
    cursor.execute(query_8,(loan_info[11],))
    property_info = cursor.fetchone()
    query_11 = "SELECT * FROM Liabilities WHERE loan = %s;"
    cursor.execute(query_11,(loan_id,))
    liabilities_info = cursor.fetchall()


    # Display Qualification Data Neatly
    loan_status = loan_info[7]
    clearConsole()
    printCentered(" Great Home Loans ",width,"=")
    printCentered(f"{loan_info[8]} (Status: {loan_status})",width)
    print()
    print(f"{'Primary Borrower:':<25} {borrower_info[1]} {borrower_info[2]}")
    print(f"--> {'Monthly Income:':<21} $  {borrower_info[6]}")
    print()
    if loan_info[2]:
        print(f"{'Coborrower:':<25} {coborrower_info[1]} {coborrower_info[2]}")
        print(f"--> {'Monthly Income:':<21} $  {coborrower_info[6]}")
    print()
    if coborrower_info:
        total_income = borrower_info[6] + coborrower_info[6]
    else:
        total_income = borrower_info[6]
```

Qualification Screen