Stockwell Financial Database

VOLUME III: GUI in Python / Data Visualization in Grafana

Austin Stockwell
March 2020

Acknowledgments

Various resources were used to help create this database. The end result was a device that was suited to my specific needs. I would like to thank all of those who helped create the final product! [1] [2] [3] [4] [5] [6]

Contents

List of Figures

1 Introduction

This project started as a desire to create a personal database to archive financial assets and liabilities. The main goal of the project was to create a system that actively helped aid future financial decisions while maintaining accurate and easily accessible (yet secure) financial data.

The basic goal has been met and now the database continues to evolve into a more advanced tool with the addition of a Graphical User Interface created in Python and Business Intelligence (BI) Dashboards in Grafana.

2 Features

The following features have been integrated into the database and will be discussed in this report:

- Graphical user interface created in Python
- Data visualization and analytics within Grafana

3 Graphical User Interface in Python

3.1 Design Overview

Because entering data by hand inside of MySQL Workbench can often be cumbersome and not viable for many end users, a Graphical User Interface that allows the input of data into the seven tables of the 'Stockwell Financial Database' was created. This tool can be run without MySQL Workbench being opened and with no prior knowledge of MySQL Workbench (or SQL) needed by the end user. As a result, the data entry process for the database has been greatly streamlined, resulting in reduced potential for errors and increased speed of entry.

The GUI opens to a main window that consists of seven buttons – each button corresponds to a separate table within the 'Stockwell Financial' database. To begin, the user simply clicks on the button corresponding to the table they wish to enter data in and enters the corresponding table data as prompted. When the user is satisfied with the data entry, a button is pressed and the data is instantly populated into the corresponding table. If there are any errors in the data entry form/process (wrong data type, error in connecting to the database, etc), a 'WARNING' message will display via popup window which includes an error code so the user may try again to correctly input their data.

If data is entered correctly, the corresponding data will be displayed to the user and the 'WARNING' popup window will NOT be displayed. The user can then return to the 'Main Menu' page by clicking the 'Back to Main Menu' button on each corresponding entry page.

3.2 GUI Forms

This section includes photos of each of the seven windows of the GUI that are used to enter data into the corresponding tables as well as the 'Main Menu' page.

3.2.1 Main Menu

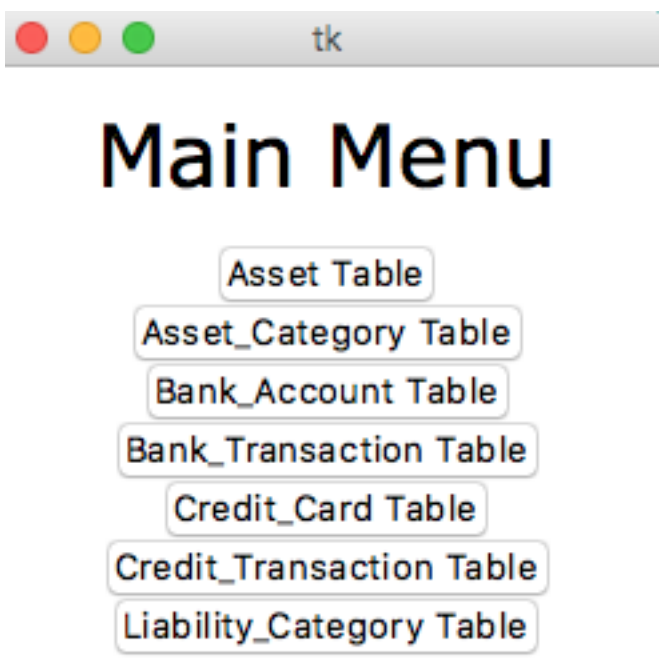


Figure 1: Main Menu

3.2.2 Asset Table Entry Page

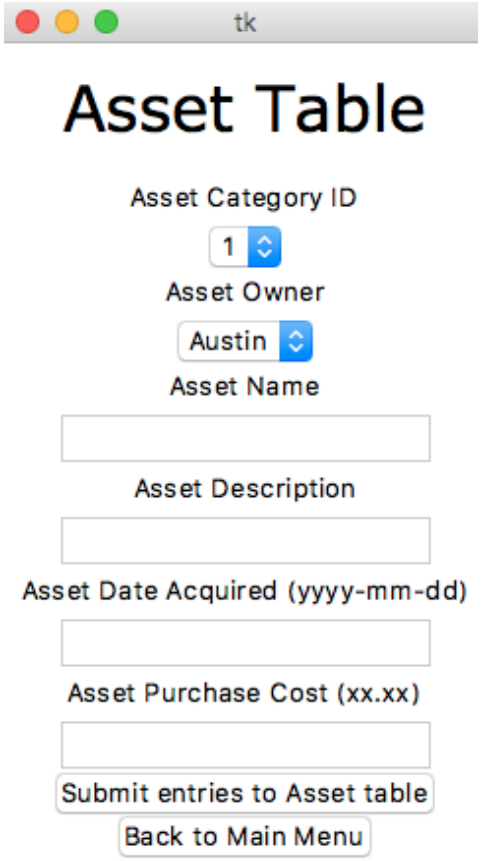


Figure 2: Asset Table Entry Page

NOTE: The 'Asset Table Entry Page' does NOT include entries for a 'Sell Price' and a 'Sell Date'. It was decided in the design stage that any updates to those fields will be coded within MySQL Workbench using SQL and will not be changed via GUI.

3.2.3 Asset Category Table Entry Page

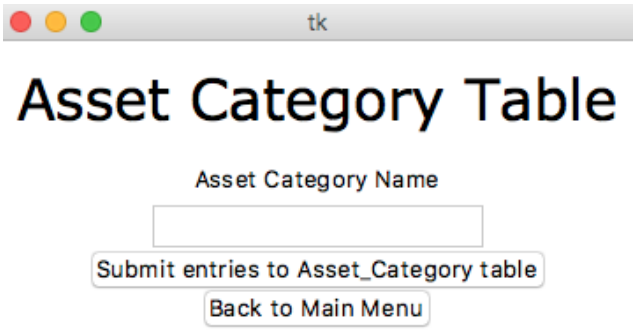


Figure 3: Asset Category Table Entry Page

3.2.4 Bank Account Table Entry Page

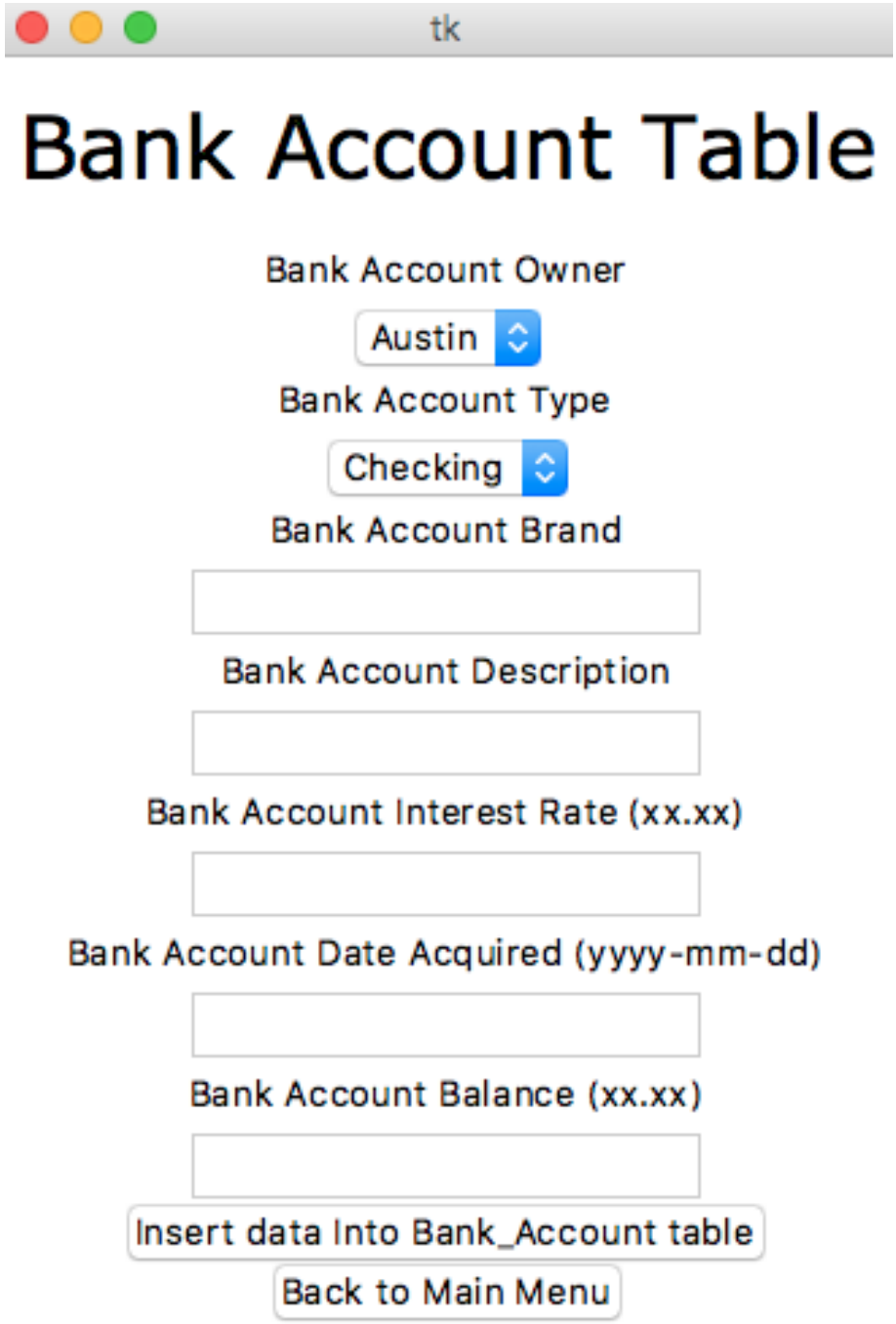


Figure 4: Bank Account Table Entry Page

3.2.5 Bank Transaction Table Entry Page

Because the 'Bank Transaction Table' can accept either a deposit OR a withdrawal, logic was implemented within the Python code to ensure a deposit was not made at the same time as a withdrawal. For example, if a WITHDRAWAL was being made, the 'Asset ID', 'Asset Category', and 'Deposit Amount' fields were set to NULL or 00.00. On the other hand, if a DEPOSIT was being made, the 'Withdrawal Amount', 'Liability Category' and 'Credit Card ID') were set to NULL or 00.00.

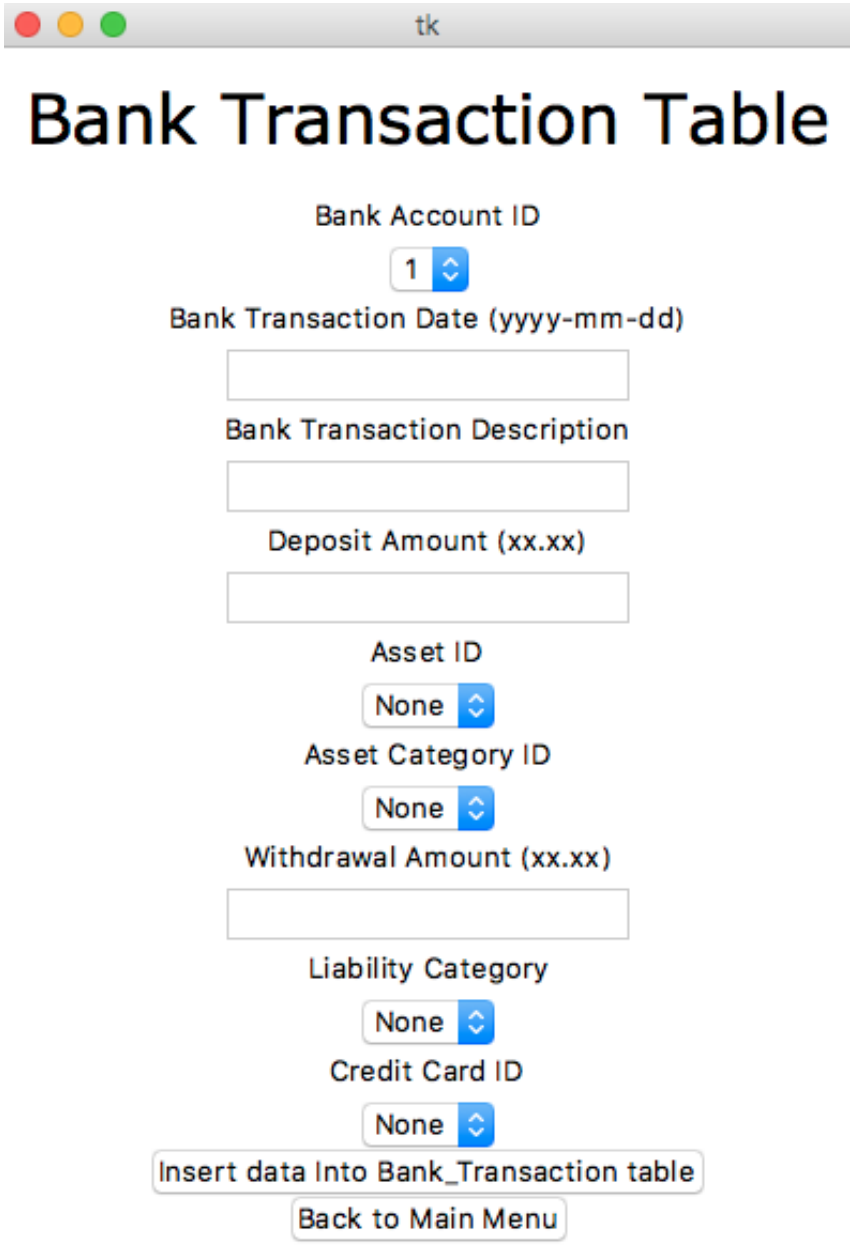


Figure 5: Bank Transaction Table Entry Page

3.2.6 Credit Card Table Entry Page

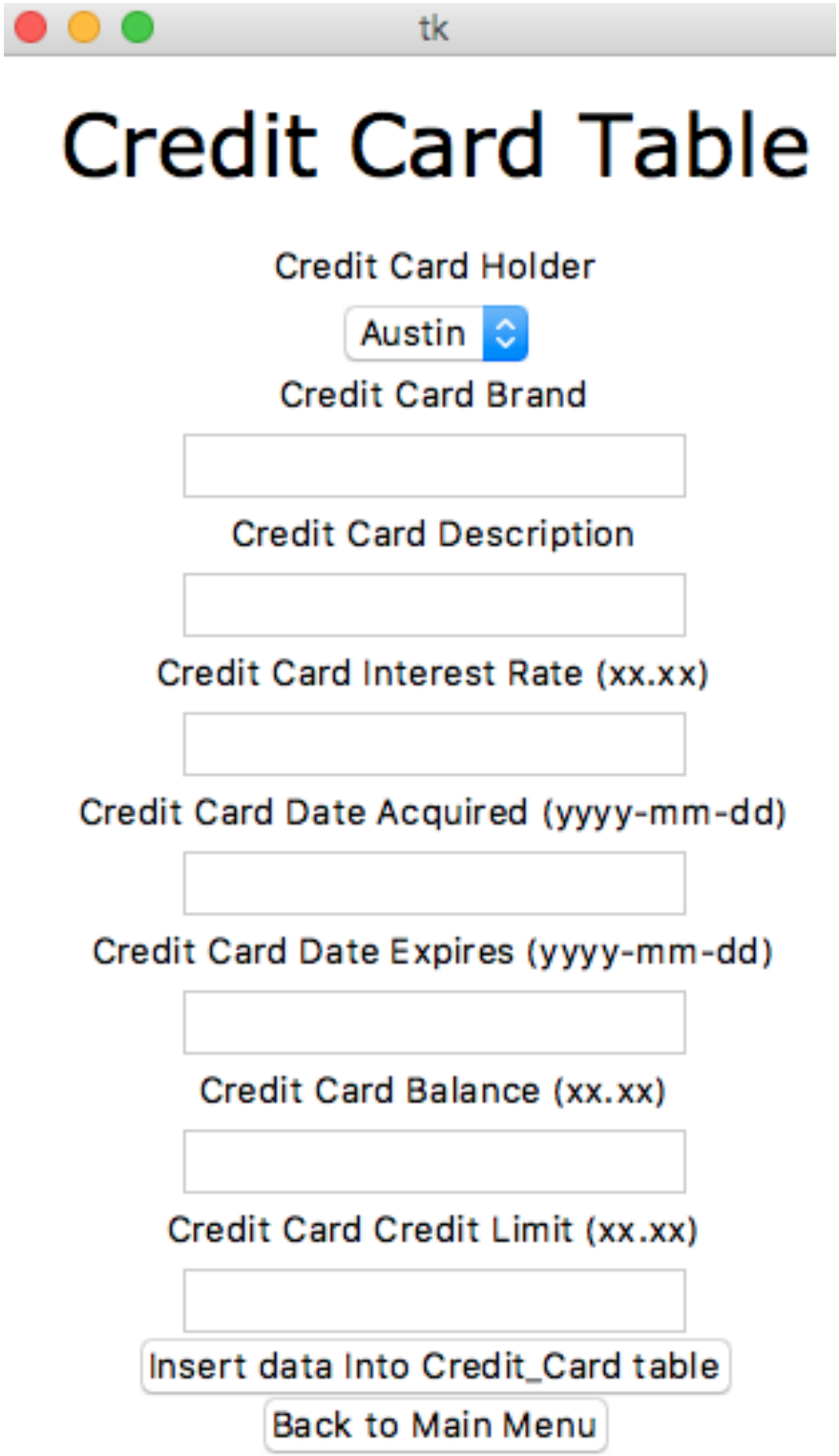


Figure 6: Credit Card Table Entry Page

3.2.7 Credit Transaction Table Entry Page

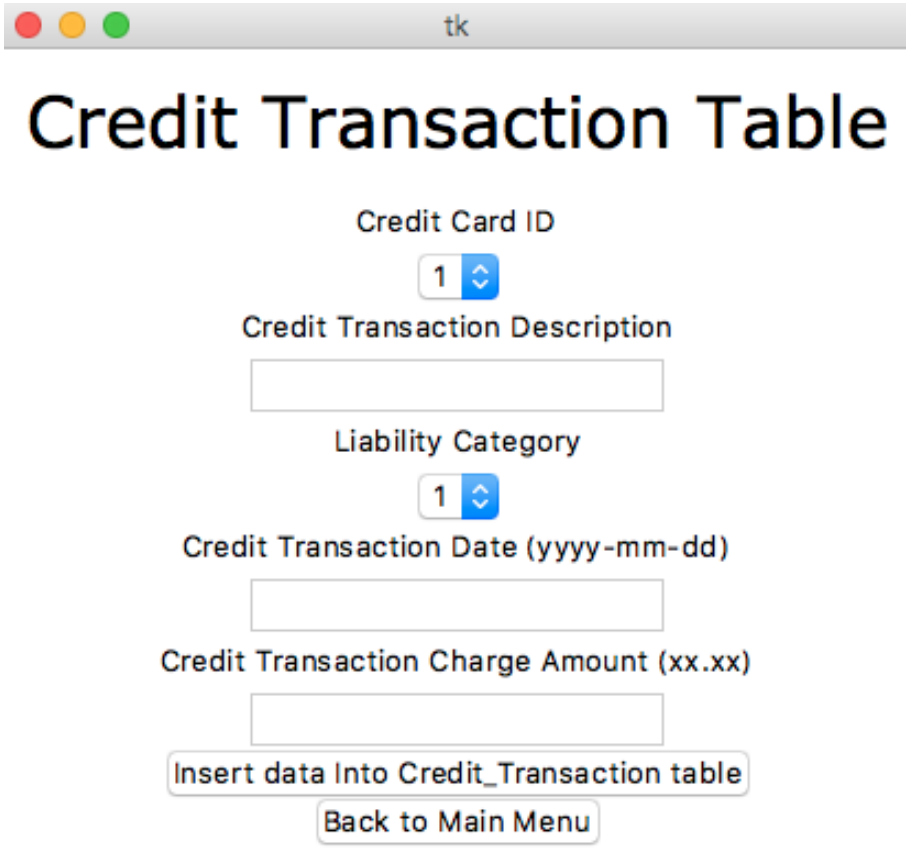


Figure 7: Credit Transaction Table Entry Page

3.2.8 Liability Category Table Entry Page

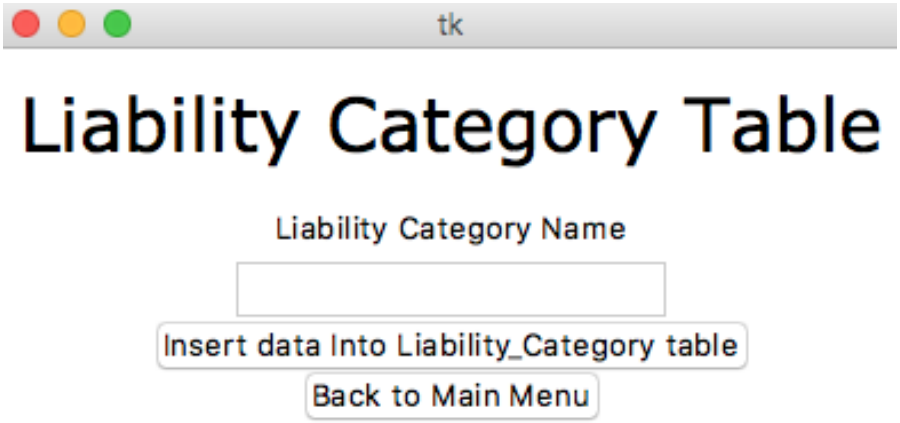


Figure 8: Liability Category Table Entry Page

3.2.9 Successful Data Entry Message

If data entered in the GUI satisfies the requirements of the fields within MySQL, a popup window will show all of the data that was just input into the database in a convenient format as shown below:

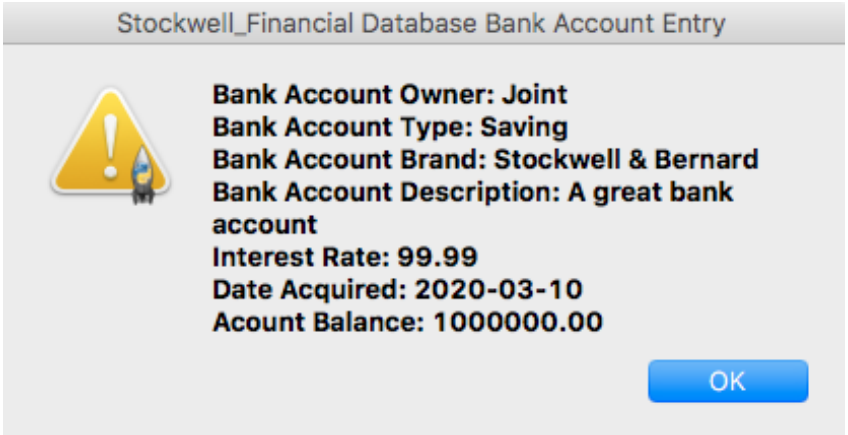


Figure 9: Data Entered Successfully

3.2.10 Error Message

If data entered in the GUI does NOT satisfy the requirements of the fields within MySQL (or other errors occur), a popup window will show a 'WARNING' message that also includes an error code for troubleshooting / help in reformatting the data for successful entry.

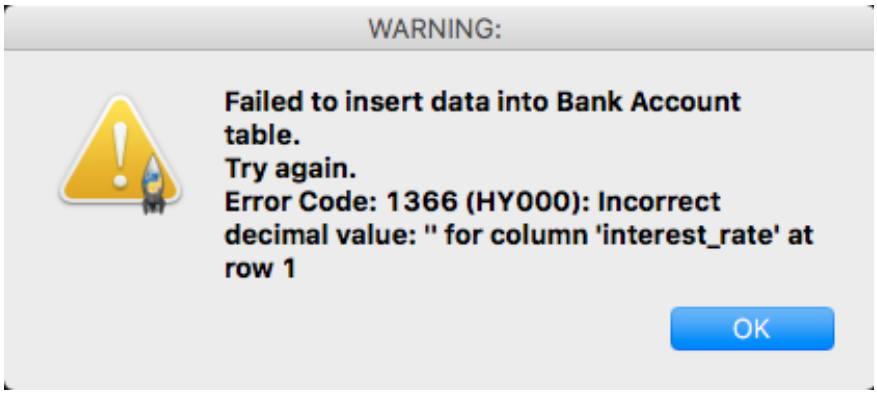


Figure 10: Error Entering Data

NOTE: Complete selection of PYTHON code is found in Appendix A.

4 BI Dashboards in Grafana

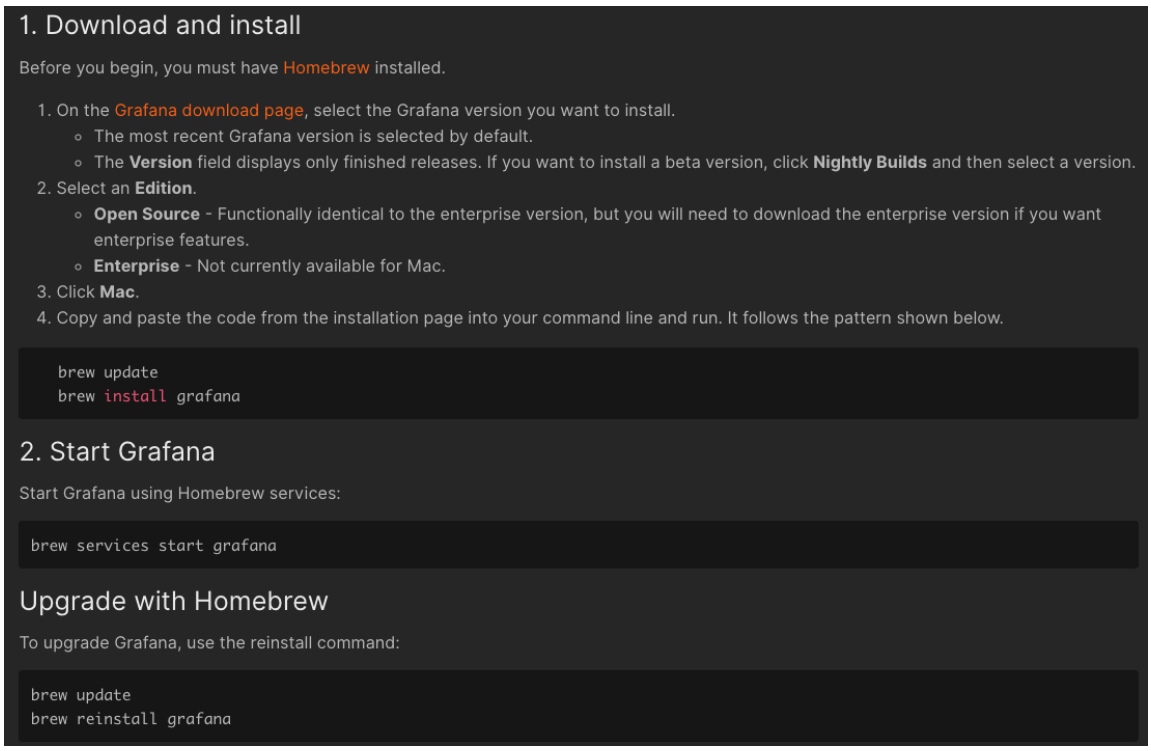
4.1 Introduction to BI Dashboards

It is very important to be able to make sense of data within databases. Data visualization tools are extremely important as they present data in a straightforward manner that would be impossible to do without such tools. Therefore, with the ability to visualize data comes the advantage of creating strategies and plans on how to act on data in valuable ways.

The 'Stockwell Financial' database was integrated with Grafana (an open source Data Visualization tool) to create custom Business Intelligence (BI) Dashboards that allow the data within the database to be conceptualized more intuitively than data in raw format.

4.2 Installation

To begin using Grafana for data visualization, Grafana was installed using 'Homebrew' package manager and the Terminal within OS X.



1. Download and install

Before you begin, you must have Homebrew installed.

1. On the Grafana download page, select the Grafana version you want to install.
   - The most recent Grafana version is selected by default.
   - The **Version** field displays only finished releases. If you want to install a beta version, click **Nightly Builds** and then select a version.
2. Select an **Edition**.
   - **Open Source** - Functionally identical to the enterprise version, but you will need to download the enterprise version if you want enterprise features.
   - **Enterprise** - Not currently available for Mac.
3. Click **Mac**.
4. Copy and paste the code from the installation page into your command line and run. It follows the pattern shown below.

```
brew update
brew install grafana
```

2. Start Grafana

Start Grafana using Homebrew services:

```
brew services start grafana
```

Upgrade with Homebrew

To upgrade Grafana, use the reinstall command:

```
brew update
brew reinstall grafana
```

Figure 11: Grafana Installation Instructions

4.3 Connection to Grafana

Once Grafana is successfully installed, the application is accessed via web-browser. The default connection to the data is: **http://localhost:3000/**

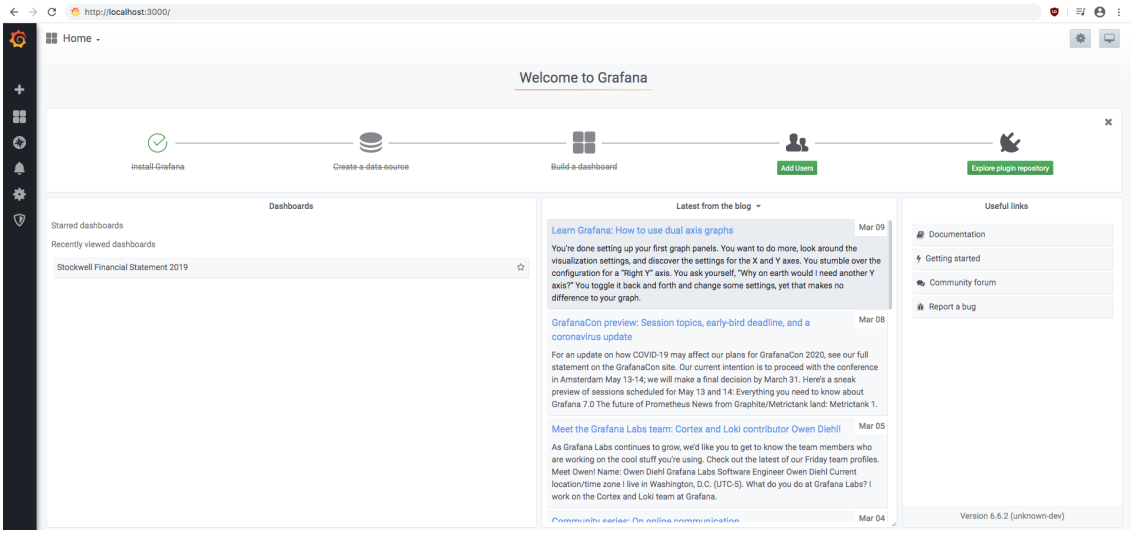


Figure 12: Grafana Home Page

The 'Welcome to Grafana' page takes the user through setup. The database connection was created by choosing 'Create a Datasource' and then selecting the 'MySQL' option. The required fields to connect to the database were then completed.

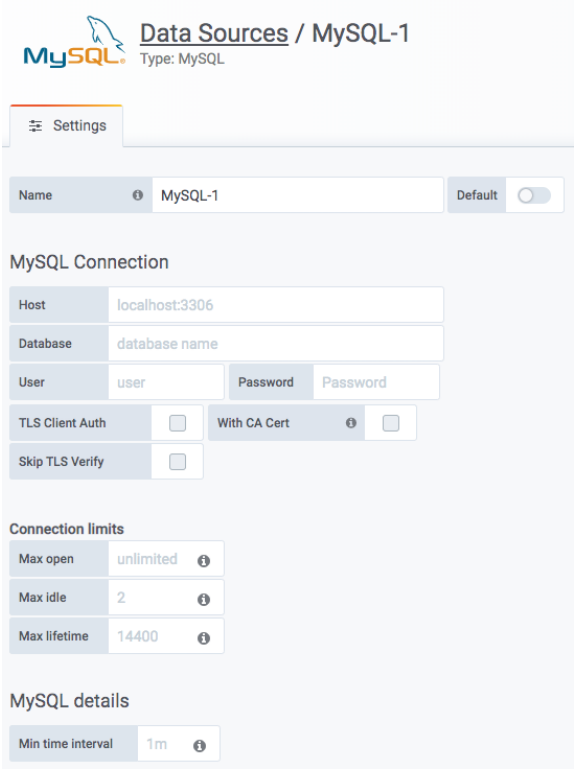


Figure 13: MySQL Data Connection in Grafana

4.4 Building BI Dashboard in Grafana

The BI dashboard in Grafana was created for the sole purpose of graphing total expenditures of Credit Cards in the year 2019 (as only Credit Card and Credit Transaction information was tracked with the 'Stockwell Financial' database in 2019).

The BI Dashboard shows two separate visualizations:

- Credit Card Holder Totals (2019)
- Credit Transaction Category Totals (2019)

The 'Credit Card Holder Totals 2019' visualization shows how much total spending occurred on the credit cards for each person over the course of the year. For example, if 'Person A' had three credit cards, all of the transactions that occurred on those cards in the year of 2019 were added together and graphed against the same measures for 'Person B'.

```
SELECT NOW() AS "time", card_holder AS metric, total as value FROM vcredit_card_holder_totals_2019 ORDER BY card_holder
```

Format as | Time series | Query Builder | Show Help

Figure 14: Credit Card Holder (2019) Grafana Code

NOTE: To create this view, it was necessary to create a new VIEW within MySQL that interfaces with the Grafana code above. The VIEW 'vCredit Card Holder Totals 2019' is shown below.

```
/*
Author: Austin Stockwell
  Type: VIEW
  Scope: To get totals for spending in each person's credit card history
  Directions:  FIND AND REPLACE ALL dates (using COMMAND +F) and run query.
 */

USE Stockwell_Financial;
CREATE VIEW vCredit_Card_Holder_Totals_2019
AS
SELECT
    card_holder,
    COUNT(*), SUM(charge) total
FROM
    vCredit_Transactions
WHERE
    charge_date BETWEEN '2019-01-01' AND '2019-12-31'
GROUP BY card_holder
ORDER BY total DESC;
```

Figure 15: vCreditCard HolderTotals2019 VIEW

The 'Credit Transaction Category Totals 2019' visualization shows the total amounts spent on the respective liability categories in the year of 2019.

```
SELECT NOW() AS "time", name AS metric, total as value FROM vCredit_Transactions_Category_Totals_2019  WHERE total > 0 order by total
```

Format as | Time series | Query Builder | Show Help

Figure 16: Credit Transaction Category Totals (2019) Grafana Code

NOTE: To create this view, it was necessary to create a new VIEW within MySQL that interfaces with the Grafana code above. The VIEW 'vCredit Transactions Category Totals 2019' is shown on the following pages.

```sql
/*
Author: Austin Stockwell
  Type: VIEW
  Scope: To get monthly totals for spending in each Credit Card Transaction Liability Category
  Directions:  FIND AND REPLACE ALL dates (using COMMAND +F) and run query.
  */

USE Stockwell_Financial;
CREATE VIEW vCredit_Transactions_Category_Totals_2019
AS
-- ----------------------------------------
-- HOW MUCH SPENT ON GASOLINE
-- ----------------------------------------
SELECT
   fk_liability_category_ID,
   Liability_Category.name,
   COUNT(*), SUM(charge) total
FROM
   Credit_Transaction
JOIN Liability_Category ON Credit_Transaction.fk_liability_category_ID = Liability_Category.idLiability_Category
WHERE fk_liability_category_ID = 10
and charge_date between '2019-01-01' and '2019-12-31'
-- ----------------------------------------
-- HOW MUCH SPENT ON CAR REPAIRS
-- ----------------------------------------
UNION
SELECT
   fk_liability_category_ID,
   Liability_Category.name,
   COUNT(*), SUM(charge) total
FROM
   Credit_Transaction
JOIN Liability_Category ON Credit_Transaction.fk_liability_category_ID = Liability_Category.idLiability_Category
WHERE fk_liability_category_ID = 11
and charge_date between '2019-01-01' and '2019-12-31'
-- ----------------------------------------
-- HOW MUCH SPENT ON GROCERIES
-- ----------------------------------------
UNION
SELECT
   fk_liability_category_ID,
   Liability_Category.name,
   COUNT(*), SUM(charge) total
FROM
   Credit_Transaction
JOIN Liability_Category ON Credit_Transaction.fk_liability_category_ID = Liability_Category.idLiability_Category
WHERE fk_liability_category_ID = 12
and charge_date between '2019-01-01' and '2019-12-31'
```

Figure 17: vCreditTransactionsCategoryTotals2019 VIEW (1/4)

```sql
-- ----------------------------------------
-- HOW MUCH SPENT ON BOOKS
-- ----------------------------------------
UNION
SELECT
   fk_liability_category_ID,
   Liability_Category.name,
   COUNT(*), SUM(charge) total
FROM
   Credit_Transaction
JOIN Liability_Category ON Credit_Transaction.fk_liability_category_ID = Liability_Category.idLiability_Category
WHERE fk_liability_category_ID = 17
and charge_date between '2019-01-01' and '2019-12-31'
-- ----------------------------------------
-- HOW MUCH SPENT ON HOBBIES
-- ----------------------------------------
UNION
SELECT
   fk_liability_category_ID,
   Liability_Category.name,
   COUNT(*), SUM(charge) total
FROM
   Credit_Transaction
JOIN Liability_Category ON Credit_Transaction.fk_liability_category_ID = Liability_Category.idLiability_Category
WHERE fk_liability_category_ID = 18
and charge_date between '2019-01-01' and '2019-12-31'
-- ----------------------------------------
-- HOW MUCH SPENT ON MUSIC
-- ----------------------------------------
UNION
SELECT
   fk_liability_category_ID,
   Liability_Category.name,
   COUNT(*), SUM(charge) total
FROM
   Credit_Transaction
JOIN Liability_Category ON Credit_Transaction.fk_liability_category_ID = Liability_Category.idLiability_Category
WHERE fk_liability_category_ID = 19
and charge_date between '2019-01-01' and '2019-12-31'
-- ----------------------------------------
-- HOW MUCH SPENT ON RESTAURANT
-- ----------------------------------------
UNION
SELECT
   fk_liability_category_ID,
   Liability_Category.name,
   COUNT(*), SUM(charge) total
FROM
   Credit_Transaction
JOIN Liability_Category ON Credit_Transaction.fk_liability_category_ID = Liability_Category.idLiability_Category
WHERE fk_liability_category_ID = 20
and charge_date between '2019-01-01' and '2019-12-31'
```

Figure 18: vCreditTransactionsCategoryTotals2019 VIEW(2/4)

```sql
-- ----------------------------------------
-- HOW MUCH SPENT ON ENTERTAINMENT
-- ----------------------------------------
UNION
SELECT
   fk_liability_category_ID,
   Liability_Category.name,
   COUNT(*), SUM(charge) total
FROM
   Credit_Transaction
JOIN Liability_Category ON Credit_Transaction.fk_liability_category_ID = Liability_Category.idLiability_Category
WHERE fk_liability_category_ID = 21
and charge_date between '2019-01-01' and '2019-12-31'
-- ----------------------------------------
-- HOW MUCH SPENT ON CLOTHING
-- ----------------------------------------
UNION
SELECT
   fk_liability_category_ID,
   Liability_Category.name,
   COUNT(*), SUM(charge) total
FROM
   Credit_Transaction
JOIN Liability_Category ON Credit_Transaction.fk_liability_category_ID = Liability_Category.idLiability_Category
WHERE fk_liability_category_ID = 22
and charge_date between '2019-01-01' and '2019-12-31'
-- ----------------------------------------
-- HOW MUCH SPENT ON TRAVEL
-- ----------------------------------------
UNION
SELECT
   fk_liability_category_ID,
   Liability_Category.name,
   COUNT(*), SUM(charge) total
FROM
   Credit_Transaction
JOIN Liability_Category ON Credit_Transaction.fk_liability_category_ID = Liability_Category.idLiability_Category
WHERE fk_liability_category_ID = 23
and charge_date between '2019-01-01' and '2019-12-31'
-- ----------------------------------------
-- HOW MUCH SPENT ON JEWELERY
-- ----------------------------------------
UNION
SELECT
   fk_liability_category_ID,
   Liability_Category.name,
   COUNT(*), SUM(charge) total
FROM
   Credit_Transaction
JOIN Liability_Category ON Credit_Transaction.fk_liability_category_ID = Liability_Category.idLiability_Category
WHERE fk_liability_category_ID = 24
and charge_date between '2019-01-01' and '2019-12-31'
```

Figure 19: vCreditTransactionsCategoryTotals2019 VIEW (3/4)

```sql
-- ----------------------------------------
-- HOW MUCH SPENT ON HOME IMPROVEMENT
-- ----------------------------------------
UNION
SELECT
   fk_liability_category_ID,
   Liability_Category.name,
   COUNT(*), SUM(charge) total
FROM
   Credit_Transaction
JOIN Liability_Category ON Credit_Transaction.fk_liability_category_ID = Liability_Category.idLiability_Category
WHERE fk_liability_category_ID = 25
and charge_date between '2019-01-01' and '2019-12-31'
-- ----------------------------------------
-- HOW MUCH SPENT ON MISCELLANEOUS
-- ----------------------------------------
UNION
SELECT
   fk_liability_category_ID,
   Liability_Category.name,
   COUNT(*), SUM(charge) total
FROM
   Credit_Transaction
JOIN Liability_Category ON Credit_Transaction.fk_liability_category_ID = Liability_Category.idLiability_Category
WHERE fk_liability_category_ID = 100
and charge_date between '2019-01-01' and '2019-12-31'
ORDER BY fk_liability_category_ID DESC;
```

Figure 20: vCreditTransactionsCategoryTotals 2019 VIEW(4/4)

4.5 Finished Grafana BI Dashboard

The figure below shows the two separate visualizations on one common 'BI Dashboard'. The top pie chart is the 'Credit Card Holder Totals (2019)' visualization. The bottom bar graph is the 'Credit Transaction Category Totals (2019)' visualization.

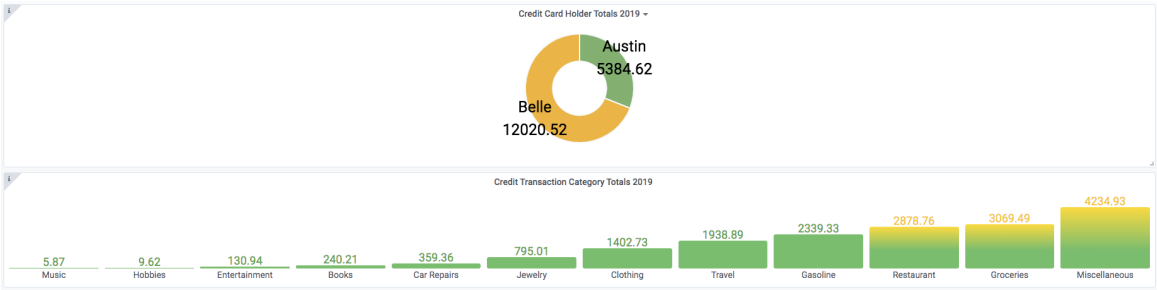


Figure 21: Completed Grafana BI Dashboard

5 Future Improvements

The current state of the database is completely functional and satisfies the original goal of the project. However, there are improvements that will be made and integrated into future iterations of the database. These features will be documented in the upcoming volumes.

The following improvements will be integrated into this project:

- Commit() and rollback() functionality
- Remote access to database
- Refinements to GUI and BI Dashboards

References

[1] M. Z. L. Phillip J. Pratt, *A Guide to MySQL*. Course Technology, Cengage Learning, 2006.

[2] ——, *Concepts of Database Management*. Cengage Learning, 2012, vol. 8.

[3] (2020, March). [Online]. Available: www.grafana.com

[4] (2020, March). [Online]. Available: https://www.tcl.tk/

[5] (2020, March). [Online]. Available: https://www.mysqltutorial.org/getting-started-mysql-python-connector/

[6] (2020, March). [Online]. Available: https://pynative.com/python-mysql-database-connection/

6 Appendix A: Python Code

The Python code is shown in the following pages. The code consists of three files:

- V3GUIShell.py (Pg 19)
- pageinterface.py (Pg 20 - 30)
- entry.py (Pg 31 - 46)
- MenuFunctions.py (Pg 47 - 54)

```
##############################################################################
#    Author: Austin Stockwell
#    Date: 03-02-2020
#    Description: Data entry program for Stockwell_Financial MySQL database.
#    File: V3_GUI_Shell.py
##############################################################################
import mysql.connector
from tkinter import *
import tkinter as tk
from page_interface import *
LARGE_FONT = ("Verdana", 32)
##############################################################################
class Stockwell_FinancialApp(tk.Tk):
    def __init__(self, *args, **kwargs):

        tk.Tk.__init__(self, *args, **kwargs)

        container = tk.Frame(self)
        container.pack(side="top", fill="both", expand = True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames = {}

        for F in (StartPage, Asset, Asset_Category, Bank_Account,
                  Bank_Transaction,Credit_Card, Credit_Transaction,
                  Liability_Category):

            frame = F(container, self)

            self.frames[F] = frame

            frame.grid(row=0, column=0, sticky="nsew")

        self.show_frame(StartPage)

    def show_frame(self, cont):
        frame = self.frames[cont]
        frame.tkraise()

##############################################################################
## RUN APPLICATION
##############################################################################
app = Stockwell_FinancialApp()
app.mainloop()
```

```
################################################################################
#    Author: Austin Stockwell
#    Date: 03-02-2020
#    Description: This file contains all of the Tkinter frames (8 total) and
#                 each form's associated tkinter GUI objects (buttons, labels).
#                 Each form (class) also contains a call to its unique
#                 SUBMIT_x_ENTRY method that is used to retrieve the data
#                 entered on each form and pass it to the entry3.py file for
#                 further processing.
#    File: page_interface.py
################################################################################
import mysql.connector
from tkinter import *
import tkinter as tk
from entry import *
from MenuFunctions import *
LARGE_FONT= ("Verdana", 32)
################################################################################
class StartPage(tk.Frame):
    """Creates the Main Menu with 7 buttons corresponding to the MySQL tables"""
    def __init__(self, parent, controller):
        tk.Frame.__init__(self,parent)
        label = tk.Label(self, text="Main Menu", font=LARGE_FONT)
        label.pack(pady=10,padx=10)


        # GUI OBJECTS CREATION (LABELS, BUTTONS, MENUS, ETC)
        buttonAsset = tk.Button(self, text="Asset Table",
                        command=lambda: controller.show_frame(Asset))
        buttonAsset.pack()

        buttonAssetCategory = tk.Button(self, text="Asset_Category Table",
                        command=lambda: controller.show_frame(Asset_Category))
        buttonAssetCategory.pack()

        buttonBankAccount = tk.Button(self, text="Bank_Account Table",
                        command=lambda: controller.show_frame(Bank_Account))
        buttonBankAccount.pack()

        buttonBankTransaction = tk.Button(self, text="Bank_Transaction Table",
                        command=lambda: controller.show_frame(Bank_Transaction))
        buttonBankTransaction.pack()

        buttonBankTransaction = tk.Button(self, text="Credit_Card Table",
                        command=lambda: controller.show_frame(Credit_Card))
        buttonBankTransaction.pack()

        buttonBankTransaction = tk.Button(self, text="Credit_Transaction Table",
                        command=lambda: controller.show_frame(Credit_Transaction))
        buttonBankTransaction.pack()

        buttonBankTransaction = tk.Button(self, text="Liability_Category Table",
                        command=lambda: controller.show_frame(Liability_Category))
        buttonBankTransaction.pack()
```

```
################################################################################
## ENTRY FORM WINDOWS
################################################################################
class Asset(tk.Frame):
    """Creates the Asset form for data entry into Asset table"""
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Asset Table", font=LARGE_FONT)
        label.pack(pady=10,padx=10)


        # GUI OBJECTS CREATION (LABELS, BUTTONS, MENUS, ETC)
        tk.Label(self,text="Asset Category ID").pack()
        varAsset_fk_asset_category_ID = StringVar(self)
        varAsset_fk_asset_category_ID.set('(1) Stock') # default value
        optionAsset_fk_asset_category_ID = OptionMenu(self,
                                varAsset_fk_asset_category_ID,
                                '(1) Stock',
                                '(2) Bond',
                                '(3) Index Fund',
                                '(4) Mutual Fund',
                                '(5) IRA',
                                '(6) Roth IRA',
                                '(7) 401k',
                                '(8) Roth 401k',
                                '(9) CD',
                                '(10) Commodity',
                                '(11) Cryptocurrency',
                                '(12) Real Estate')
        optionAsset_fk_asset_category_ID.pack()

        tk.Label(self,text="Asset Owner").pack()
        varAsset_owner = StringVar(self)
        varAsset_owner.set("Austin") # default value
        optionAsset_owner = OptionMenu(self, varAsset_owner,
                                "Austin",
                                "Belle",
                                "Joint")
        optionAsset_owner.pack()

        tk.Label(self,text="Asset Name").pack()
        entryAsset_name = tk.Entry(self)
        entryAsset_name.pack()

        tk.Label(self,text="Asset Description").pack()
        entryAsset_description = tk.Entry(self)
        entryAsset_description.pack()

        tk.Label(self,text="Asset Date Acquired (yyyy-mm-dd)").pack()
        entryAsset_date_acquired = tk.Entry(self)
        entryAsset_date_acquired.pack()

        tk.Label(self,text="Asset Purchase Cost (xx.xx)").pack()
        entryAsset_purchase_cost = tk.Entry(self)
```

```
        entryAsset_purchase_cost.pack()

        buttonAssetCommit      = tk.Button(self, text = 'Submit entries to Asset table',
                        command = lambda: SUBMIT_ASSET_ENTRY())
        buttonAssetCommit.pack()

        button1 = tk.Button(self, text="Back to Main Menu",
                        command=lambda: controller.show_frame(StartPage))
        button1.pack()


        def SUBMIT_ASSET_ENTRY():
            """Retrieves values entered in Asset frame"""
            print("DOING ASSETS")
            #Creates an instance (object) of the Asset_ENTRY class.
            ASSET_INPUT = Asset_ENTRY()

ASSET_INPUT.set_asset_fk_asset_category_ID(GetAsset_AssetCategoryMenu(varAsset_fk_asset_category_ID))
            ASSET_INPUT.set_asset_owner(varAsset_owner.get())
            ASSET_INPUT.set_asset_name(entryAsset_name.get())
            ASSET_INPUT.set_asset_description(entryAsset_description.get())
            ASSET_INPUT.set_asset_date_acquired(entryAsset_date_acquired.get())
            ASSET_INPUT.set_asset_purchase_cost(entryAsset_purchase_cost.get())
            ASSET_INPUT.SUBMIT_ASSET()



######################################################################
class Asset_Category(tk.Frame):
    """Creates the Asset_Category form for data entry into Asset_Category table"""
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Asset Category Table", font=LARGE_FONT)
        label.pack(pady=10,padx=10)


        # GUI OBJECTS CREATION (LABELS, BUTTONS, MENUS, ETC)
        tk.Label(self,text="Asset Category Name").pack()
        entryAsset_Category_name = tk.Entry(self)
        entryAsset_Category_name.pack()

        buttonAsset_CategoryCommit = tk.Button(self, text="Submit entries to Asset_Category table",
                        command=lambda: SUBMIT_ASSET_CATEGORY_ENTRY())
        buttonAsset_CategoryCommit.pack()

        button1 = tk.Button(self, text="Back to Main Menu",
                        command=lambda: controller.show_frame(StartPage))
        button1.pack()


        def SUBMIT_ASSET_CATEGORY_ENTRY():
            """Retrieves values entered in Asset_Category frame"""
            print("DOING ASSET CATEGORIES")
            ASSET_INPUT = AssetCategory_ENTRY()
            ASSET_INPUT.set_asset_category_name(entryAsset_Category_name.get())
```

```
        ASSET_INPUT.SUBMIT_ASSET_CATEGORY()


#######################################################################
class Bank_Account(tk.Frame):
    """Creates the Bank_Account form for data entry into Bank_Account table"""
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Bank Account Table", font=LARGE_FONT)
        label.pack(pady=10,padx=10)


        # GUI OBJECTS CREATION (LABELS, BUTTONS, MENUS, ETC)
        tk.Label(self,text="Bank Account Owner").pack()
        varBank_Account_owner = StringVar(self)
        varBank_Account_owner.set("Austin") # default value
        optionBank_Account_owner = OptionMenu(self,
                                varBank_Account_owner,
                                "Austin",
                                "Belle",
                                "Joint")
        optionBank_Account_owner.pack()

        tk.Label(self,text="Bank Account Type").pack()
        varBank_Account_type = StringVar(self)
        varBank_Account_type.set("Checking") # default value
        optionBank_Account_type = OptionMenu(self,
                                varBank_Account_type,
                                "Checking",
                                "Saving")
        optionBank_Account_type.pack()

        tk.Label(self,text="Bank Account Brand").pack()
        entryBank_Account_brand = tk.Entry(self)
        entryBank_Account_brand.pack()

        tk.Label(self,text="Bank Account Description").pack()
        entryBank_Account_description = tk.Entry(self)
        entryBank_Account_description.pack()

        tk.Label(self,text="Bank Account Interest Rate (xx.xx)").pack()
        entryBank_Account_interest_rate = tk.Entry(self)
        entryBank_Account_interest_rate.pack()

        tk.Label(self,text="Bank Account Date Acquired (yyyy-mm-dd)").pack()
        entryBank_Account_date_acquired = tk.Entry(self)
        entryBank_Account_date_acquired.pack()

        tk.Label(self,text="Bank Account Balance (xx.xx)").pack()
        entryBank_Account_balance = tk.Entry(self)
        entryBank_Account_balance.pack()

        buttonBank_AccountCommit = tk.Button(self, text="Insert data Into Bank_Account table",
                        command=lambda: SUBMIT_BANK_ACCOUNT_ENTRY())
```

```
    buttonBank_AccountCommit.pack()

    button1 = tk.Button(self, text="Back to Main Menu",
                        command=lambda: controller.show_frame(StartPage))
    button1.pack()


    def SUBMIT_BANK_ACCOUNT_ENTRY():
        """Retrieves values entered in Bank_Account frame"""
        print("DOING BANK ACCOUNTS")
        ASSET_INPUT = BankAccount_ENTRY()
        ASSET_INPUT.set_bank_account_owner(varBank_Account_owner.get())
        ASSET_INPUT.set_bank_account_type(varBank_Account_type.get())
        ASSET_INPUT.set_bank_account_brand(entryBank_Account_brand.get())
        ASSET_INPUT.set_bank_account_description(entryBank_Account_description.get())
        ASSET_INPUT.set_bank_account_interest_rate(entryBank_Account_interest_rate.get())
        ASSET_INPUT.set_bank_account_date_acquired(entryBank_Account_date_acquired.get())
        ASSET_INPUT.set_bank_account_balance(entryBank_Account_balance.get())
        ASSET_INPUT.SUBMIT_BANK_ACCOUNT()



#######################################################################
class Bank_Transaction(tk.Frame):
    """Creates the Bank_Transaction form for data entry into Bank_Transaction table"""
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Bank Transaction Table", font=LARGE_FONT)
        label.pack(pady=10,padx=10)


        # GUI OBJECTS CREATION (LABELS, BUTTONS, MENUS, ETC)
        tk.Label(self,text="Bank Account ID").pack()
        varBank_Transaction_fk_bank_account_ID = StringVar(self)
        varBank_Transaction_fk_bank_account_ID.set('(1) Austin Centier Saving')
        optionBank_Transaction_fk_bank_account_ID = tk.OptionMenu(self,
                                varBank_Transaction_fk_bank_account_ID,
                                '(1) Austin Centier Saving',
                                '(2) Austin Centier Checking',
                                '(3) Belle CHASE Saving',
                                '(4) Joint CHASE Checking',
                                '(5) Joint CHASE EMERGENCY SAVING')
        optionBank_Transaction_fk_bank_account_ID.pack()

        tk.Label(self,text="Bank Transaction Date (yyyy-mm-dd)").pack()
        entryBank_Transaction_date = tk.Entry(self)
        entryBank_Transaction_date.pack()

        tk.Label(self, text="Bank Transaction Description").pack()
        entryBank_Transaction_description = tk.Entry(self)
        entryBank_Transaction_description.pack()

        tk.Label(self, text="Deposit Amount (xx.xx)").pack()
        entryBank_Transaction_deposit= tk.Entry(self)
        entryBank_Transaction_deposit.pack()
```

```
tk.Label(self,text="Asset ID").pack()
varBank_Transaction_fk_asset_ID = StringVar(self)
varBank_Transaction_fk_asset_ID.set("None")
optionBank_Transaction_fk_asset_ID = OptionMenu(self,
                                varBank_Transaction_fk_asset_ID,
                                "None",
                                '(1) Ethereum',
                                '(2) Ripple')
optionBank_Transaction_fk_asset_ID.pack()

tk.Label(self,text="Asset Category ID").pack()
varBank_Transaction_fk_asset_category_ID = StringVar(self)
varBank_Transaction_fk_asset_category_ID.set("None") # default value
optionBank_Transaction_fk_asset_category_ID = OptionMenu(self,
                        varBank_Transaction_fk_asset_category_ID,
                        "None",
                        '(1) Stock',
                        '(2) Bond',
                        '(3) Index Fund',
                        '(4) Mutual Fund',
                        '(5) IRA',
                        '(6) Roth IRA',
                        '(7) 401k',
                        '(8) Roth 401k',
                        '(9) CD',
                        '(10) Commodity',
                        '(11) Cryptocurrency',
                        '(12) Real Estate',
                        '(100) Paycheck')
optionBank_Transaction_fk_asset_category_ID.pack()

tk.Label(self, text="Withdrawal Amount (xx.xx)").pack()
entryBank_Transaction_withdrawal = tk.Entry(self)
entryBank_Transaction_withdrawal.pack()

tk.Label(self, text="Liability Category").pack()
varBank_Transaction_fk_liability_category_ID = StringVar(self)
varBank_Transaction_fk_liability_category_ID.set("None") # default value
optionBank_Transaction_fk_liability_category_ID = OptionMenu(self,
                    varBank_Transaction_fk_liability_category_ID,
                    "None",
                    '(1) Rent',
                    '(2) Mortgage',
                    '(3) Water',
                    '(4) Electricity',
                    '(5) Home Insurance',
                    '(6) Trash',
                    '(7) Gas Bill',
                    '(8) Car Payment',
                    '(9) Car Insurance',
                    '(10) Gasoline',
                    '(11) Car Repairs',
                    '(12) Groceries',
                    '(13) Cellphone',
```

```
                    '(14) Wifi',
                    '(15) Bachelor Degree Loan',
                    '(16) Credit Card Payment',
                    '(17) Books',
                    '(18) Hobbies',
                    '(19) Music',
                    '(20) Restaurant',
                    '(21) Entertainment',
                    '(22) Clothing',
                    '(23) Travel',
                    '(24) Jewelery',
                    '(25) Home Improvement',
                    '(26) Self Improvement')
    optionBank_Transaction_fk_liability_category_ID.pack()

    tk.Label(self, text="Credit Card ID").pack()
    varBank_Transaction_fk_credit_card_ID = StringVar(self)
    varBank_Transaction_fk_credit_card_ID.set("None") # default value
    optionBank_Transaction_fk_credit_card_ID = OptionMenu(self,
                        varBank_Transaction_fk_credit_card_ID,
                        "None",
                        '(1) Austin: Sweetwater',
                        '(2) Austin: Discover',
                        '(3) Austin: CHASE Freedom',
                        '(4) Austin: Citi',
                        '(5) Belle: CHASE Freedom')
    optionBank_Transaction_fk_credit_card_ID.pack()

    buttonBank_TransactionCommit = tk.Button(self, text="Insert data Into Bank_Transaction table",
                command=lambda: SUBMIT_BANK_TRANSACTION_ENTRY())
    buttonBank_TransactionCommit.pack()

    button1 = tk.Button(self, text="Back to Main Menu",
                command=lambda: controller.show_frame(StartPage))
    button1.pack()


    def SUBMIT_BANK_TRANSACTION_ENTRY():
        """Retrieves values entered in Bank_Transaction frame"""
        print("DOING BANK TRANSACTIONS")
        ASSET_INPUT = BankTransaction_ENTRY()

ASSET_INPUT.set_bank_transaction_fk_bank_account_ID(GetBankTransaction_BankAccountMenu(varBank_Transac
tion_fk_bank_account_ID))
        ASSET_INPUT.set_bank_transaction_date(entryBank_Transaction_date.get())
        ASSET_INPUT.set_bank_transaction_description(entryBank_Transaction_description.get())
        ASSET_INPUT.set_bank_transaction_deposit(entryBank_Transaction_deposit.get())

ASSET_INPUT.set_bank_transaction_fk_asset_ID(GetBankTransaction_AssetMenu(varBank_Transaction_fk_asset_ID)
)

ASSET_INPUT.set_bank_transaction_fk_asset_category_ID(GetBankTransaction_AssetCategoryMenu(varBank_Trans
action_fk_asset_category_ID))
        ASSET_INPUT.set_bank_transaction_withdrawal(entryBank_Transaction_withdrawal.get())
```

```
ASSET_INPUT.set_bank_transaction_fk_liability_category_ID(GetBankTransaction_LiabilityCategoryMenu(varBank_Tr
ansaction_fk_liability_category_ID))

ASSET_INPUT.set_bank_transaction_fk_credit_card_ID(GetBankTransaction_CreditCardMenu(varBank_Transaction_f
k_credit_card_ID))
        ASSET_INPUT.SUBMIT_BANK_TRANSACTION()



#######################################################################
class Credit_Card(tk.Frame):
    """Creates the Credit_Card form for data entry into Credit_Card table"""
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Credit Card Table", font=LARGE_FONT)
        label.pack(pady=10,padx=10)


        # GUI OBJECTS CREATION (LABELS, BUTTONS, MENUS, ETC)
        tk.Label(self,text="Credit Card Holder").pack()
        varCredit_Card_owner = StringVar(self)
        varCredit_Card_owner.set("Austin") # default value
        optionCredit_Card_owner = OptionMenu(self,
                                varCredit_Card_owner,
                                "Austin",
                                "Belle",
                                "Joint")
        optionCredit_Card_owner.pack()

        tk.Label(self, text="Credit Card Brand").pack()
        entryCredit_Card_brand = tk.Entry(self)
        entryCredit_Card_brand.pack()

        tk.Label(self, text="Credit Card Description").pack()
        entryCredit_Card_description = tk.Entry(self)
        entryCredit_Card_description.pack()

        tk.Label(self, text="Credit Card Interest Rate (xx.xx)").pack()
        entryCredit_Card_interest_rate = tk.Entry(self)
        entryCredit_Card_interest_rate.pack()

        tk.Label(self, text="Credit Card Date Acquired (yyyy-mm-dd)").pack()
        entryCredit_Card_date_acquired = tk.Entry(self)
        entryCredit_Card_date_acquired.pack()

        tk.Label(self, text="Credit Card Date Expires (yyyy-mm-dd)").pack()
        entryCredit_Card_date_expires = tk.Entry(self)
        entryCredit_Card_date_expires.pack()

        tk.Label(self, text="Credit Card Balance (xx.xx)").pack()
        entryCredit_Card_balance = tk.Entry(self)
        entryCredit_Card_balance.pack()

        tk.Label(self, text="Credit Card Credit Limit (xx.xx)").pack()
```

```
        entryCredit_Card_credit_limit = tk.Entry(self)
        entryCredit_Card_credit_limit.pack()

        buttonCredit_CardCommit = tk.Button(self, text="Insert data Into Credit_Card table",
                        command=lambda: SUBMIT_CREDIT_CARD_ENTRY())
        buttonCredit_CardCommit.pack()

        button1 = tk.Button(self, text="Back to Main Menu",
                        command=lambda: controller.show_frame(StartPage))
        button1.pack()


        def SUBMIT_CREDIT_CARD_ENTRY():
            """Retrieves values entered in Credit_Card frame"""
            print("DOING CREDIT CARDS")
            ASSET_INPUT = CreditCard_ENTRY()
            ASSET_INPUT.set_credit_card_owner(varCredit_Card_owner.get())
            ASSET_INPUT.set_credit_card_brand(entryCredit_Card_brand.get())
            ASSET_INPUT.set_credit_card_description(entryCredit_Card_description.get())
            ASSET_INPUT.set_credit_card_interest_rate(entryCredit_Card_interest_rate.get())
            ASSET_INPUT.set_credit_card_date_acquired(entryCredit_Card_date_acquired.get())
            ASSET_INPUT.set_credit_card_date_expires(entryCredit_Card_date_expires.get())
            ASSET_INPUT.set_credit_card_balance(entryCredit_Card_balance.get())
            ASSET_INPUT.set_credit_card_limit(entryCredit_Card_credit_limit.get())
            ASSET_INPUT.SUBMIT_CREDIT_CARD()



#######################################################################
class Credit_Transaction(tk.Frame):
    """Creates the Credit_Transaction form for data entry into Credit_Transaction table"""
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Credit Transaction Table", font=LARGE_FONT)
        label.pack(pady=10,padx=10)


        # GUI OBJECTS CREATION (LABELS, BUTTONS, MENUS, ETC)
        tk.Label(self, text="Credit Card ID").pack()
        varCredit_Transaction_fk_credit_card_ID = StringVar(self)
        varCredit_Transaction_fk_credit_card_ID.set('(1) Austin: Sweetwater') # default value
        optionCredit_Transaction_fk_credit_card_ID = OptionMenu(self,
                                varCredit_Transaction_fk_credit_card_ID,
                                '(1) Austin: Sweetwater',
                                '(2) Austin: Discover',
                                '(3) Austin: CHASE Freedom',
                                '(4) Austin: Citi',
                                '(5) Belle: CHASE Freedom')
        optionCredit_Transaction_fk_credit_card_ID.pack()

        tk.Label(self, text="Credit Transaction Description").pack()
        entryCredit_Transaction_description = tk.Entry(self)
        entryCredit_Transaction_description.pack()

        tk.Label(self, text="Liability Category").pack()
```

```
varCredit_Transaction_fk_liability_category_ID = StringVar(self)
varCredit_Transaction_fk_liability_category_ID.set('(1) Rent') # default value
optionCredit_Transaction_fk_liability_category_ID = OptionMenu(self,
                    varCredit_Transaction_fk_liability_category_ID,
                    '(1) Rent',
                    '(2) Mortgage',
                    '(3) Water',
                    '(4) Electricity',
                    '(5) Home Insurance',
                    '(6) Trash',
                    '(7) Gas Bill',
                    '(8) Car Payment',
                    '(9) Car Insurance',
                    '(10) Gasoline',
                    '(11) Car Repairs',
                    '(12) Groceries',
                    '(13) Cellphone',
                    '(14) Wifi',
                    '(15) Bachelor Degree Loan',
                    '(16) Credit Card Payment',
                    '(17) Books',
                    '(18) Hobbies',
                    '(19) Music',
                    '(20) Restaurant',
                    '(21) Entertainment',
                    '(22) Clothing',
                    '(23) Travel',
                    '(24) Jewelery',
                    '(25) Home Improvement',
                    '(26) Self Improvement')
optionCredit_Transaction_fk_liability_category_ID.pack()

tk.Label(self, text="Credit Transaction Date (yyyy-mm-dd)").pack()
entryCredit_Transaction_charge_date = tk.Entry(self)
entryCredit_Transaction_charge_date.pack()

tk.Label(self, text="Credit Transaction Charge Amount (xx.xx)").pack()
entryCredit_Transaction_charge = tk.Entry(self)
entryCredit_Transaction_charge.pack()

buttonCredit_TransactionCommit = tk.Button(self, text="Insert data Into Credit_Transaction table",
                command=lambda: SUBMIT_CREDIT_TRANSACTION_ENTRY())
buttonCredit_TransactionCommit.pack()

button1 = tk.Button(self, text="Back to Main Menu",
                command=lambda: controller.show_frame(StartPage))
button1.pack()


def SUBMIT_CREDIT_TRANSACTION_ENTRY():
    """Retrieves values entered in Credit_Transaction frame"""
    print("DOING CREDIT CARDS")
    ASSET_INPUT = CreditTransaction_ENTRY()
```

```
ASSET_INPUT.set_credit_transaction_fk_credit_card_ID(GetCreditTransaction_CreditCardMenu(varCredit_Transactio
n_fk_credit_card_ID))
        ASSET_INPUT.set_credit_transaction_description(entryCredit_Transaction_description.get())

ASSET_INPUT.set_credit_transaction_fk_liability_category_ID(GetCreditTransaction_LiabilityCategoryMenu(varCredit
_Transaction_fk_liability_category_ID))
        ASSET_INPUT.set_credit_transaction_charge_date(entryCredit_Transaction_charge_date.get())
        ASSET_INPUT.set_credit_transaction_charge(entryCredit_Transaction_charge.get())
        ASSET_INPUT.SUBMIT_CREDIT_TRANSACTION()



#######################################################################
class Liability_Category(tk.Frame):
    """Creates the Liability_Category form for data entry into Liability_Category table"""
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        label = tk.Label(self, text="Liability Category Table", font=LARGE_FONT)
        label.pack(pady=10,padx=10)


        # GUI OBJECTS CREATION (LABELS, BUTTONS, MENUS, ETC)
        tk.Label(self, text="Liability Category Name").pack()
        entryLiability_Category_name = tk.Entry(self)
        entryLiability_Category_name.pack()

        buttonLiability_CategoryCommit = tk.Button(self, text="Insert data Into Liability_Category table",
                        command=lambda: SUBMIT_LIABILITY_CATEGORY_ENTRY())
        buttonLiability_CategoryCommit.pack()

        button1 = tk.Button(self, text="Back to Main Menu",
                        command=lambda: controller.show_frame(StartPage))
        button1.pack()


        def SUBMIT_LIABILITY_CATEGORY_ENTRY():
            """Retrieves values entered in Liability_Category frame"""
            print("DOING LIABILITY CATEGORIES")
            ASSET_INPUT = LiabilityCategory_ENTRY()
            ASSET_INPUT.set_liability_category_name(entryLiability_Category_name.get())
            ASSET_INPUT.SUBMIT_LIABILITY_CATEGORY()
```

```
################################################################################
#    Author: Austin Stockwell
#    Date: 03-02-2020
#    Description: This file contains classes that are used to enter the data
#                 into the MySQL database.  Each class corresponds to a seperate
#                 table in the database (7 total).
#                 Each class contains:
#                      0) Declaration of all GUI menu variables in each table
#                         as x_ENTRY class attributes.
#                      1) Accessors and Mutators to get and set the values of
#                         the associated GUI menu variables in page_interface
#                         file
#                      2) A message box that declares the entered values and
#                         confirms entry success / failure (with error code)
#                      3) Connection to MySQL database
#                      4) INSERT statement to enter data to associated table.
#    File: entry.py
################################################################################
import numpy as np
import mysql.connector
from tkinter import *
from tkinter import messagebox
################################################################################
class Asset_ENTRY():
    def __init__(self):
        self.entryAsset_fk_asset_category_ID = np.nan
        self.entryAsset_owner                = np.nan
        self.entryAsset_name                 = np.nan
        self.entryAsset_description          = np.nan
        self.entryAsset_date_acquired        = np.nan
        self.entryAsset_purchase_cost        = np.nan
        self.entryAsset_sell_date            = np.nan
        self.entryAsset_sell_price           = np.nan
        return

    #### Accessors and Mutators
    def set_asset_fk_asset_category_ID(self, input):
        self.entryAsset_fk_asset_category_ID = input
        return

    def set_asset_owner(self, input):
        self.entryAsset_owner = input
        return

    def set_asset_name(self, input):
        self.entryAsset_name = input
        return

    def set_asset_description(self, input):
        self.entryAsset_description = input
        return

    def set_asset_date_acquired(self, input):
        self.entryAsset_date_acquired = input
        return
```

```
def set_asset_purchase_cost(self, input):
    self.entryAsset_purchase_cost = input
    return

def set_asset_sell_date(self, input):
    self.entryAsset_sell_date = input
    return

def set_asset_sell_price(self, input):
    self.entryAsset_sell_price = input
    return

#######################################################################

def get_asset_fk_asset_cateogry_ID(self):

    return self.entryAsset_fk_asset_category_ID

def get_asset_owner(self):

    return self.entryAsset_owner

def get_asset_name(self):

    return self.entryAsset_name

def get_asset_description(self):

    return self.entryAsset_description

def get_asset_date_acquired(self):

    return self.entryAsset_date_acquired

def get_asset_purchase_cost(self):

    return self.entryAsset_purchase_cost

def get_asset_sell_date(self):

    return self.entryAsset_sell_date

def get_asset_sell_price(self):

    return self.entryAsset_sell_price

#######################################################################

def SUBMIT_ASSET(self):
    #Print user input from form to console
    print(self.get_asset_fk_asset_cateogry_ID(),
          self.get_asset_owner(),
          self.get_asset_name(),
          self.get_asset_description(),
```

```
        self.get_asset_date_acquired(),
        self.get_asset_purchase_cost())

    #POPUP CONFIRMATION WINDOW
    warningMessage = "Asset Category: " + self.entryAsset_fk_asset_category_ID
    warningMessage += "\nAsset Owner: " + self.entryAsset_owner
    warningMessage += "\nAsset Name: " + self.entryAsset_name
    warningMessage += "\nAsset Description: " + self.entryAsset_description
    warningMessage += "\nDate Acquired: " + self.entryAsset_date_acquired
    warningMessage += "\nPurchase Cost: " + self.entryAsset_purchase_cost
    window = Tk()
    window.eval('tk::PlaceWindow %s center' % window.winfo_toplevel())
    window.withdraw()
    messagebox.showwarning('Stockwell_Financial Database Asset Entry', warningMessage)
    window.deiconify()
    window.destroy()

    #INPUTS ASSET DATA FROM GUI INTO MYSQL
    try:
      connection = mysql.connector.connect(host='localhost',
                                           database='Test',
                                           user='root',
                                           password='Supra777')

      mySql_insert_query = """INSERT INTO Asset (idAsset,
                                                  fk_asset_category_ID,
                                                  owner,
                                                  name,
                                                  description,
                                                  date_acquired,
                                                  purchase_cost,
                                                  sell_date,
                                                  sell_price)
                              VALUES
                              (NULL, '%s', '%s', '%s', '%s', '%s', '%s', NULL, NULL)""" %
(str(self.get_asset_fk_asset_cateogry_ID()),
                                                                  str(self.get_asset_owner()),
                                                                  str(self.get_asset_name()),
                                                                  str(self.get_asset_description()),
                                                                  str(self.get_asset_date_acquired()),
                                                                  str(self.get_asset_purchase_cost()))
      cursor = connection.cursor()
      cursor.execute(mySql_insert_query)
      connection.commit()
      print(cursor.rowcount, "Record inserted successfully into Asset table")
      cursor.close()

    except mysql.connector.Error as error:
      messagebox.showwarning('WARNING: ', "Failed to insert data into Asset table.\nTry again.\nError Code:
{}".format(error))
      print("Failed to insert data into Asset table {}".format(error))

    finally:
      if (connection.is_connected()):
        connection.close()
```

```
            print("MySQL connection is closed")
        return


##############################################################################
class AssetCategory_ENTRY():
    def __init__(self):
        self.entryAsset_Category_name = np.nan
        return

    #### Accessors and Mutators are legit
    def set_asset_category_name(self, input):
        self.entryAsset_Category_name = input
        return

    ######################################################################

    def get_asset_category_name(self):

        return self.entryAsset_Category_name

    ######################################################################

    def SUBMIT_ASSET_CATEGORY(self):
        ### THIS IS THE FUNCTION THAT WILL WRRITE THE MEMBER DATA OUT TO SQL
        print(self.get_asset_category_name())

        #POPUP CONFIRMATION WINDOW
        window = Tk()
        window.eval('tk::PlaceWindow %s center' % window.winfo_toplevel())
        window.withdraw()
        messagebox.showwarning('Stockwell_Financial Database Asset_Category Entry',
        'Category Name: ' + self.entryAsset_Category_name)
        window.deiconify()
        window.destroy()

        #INPUTS ASSET DATA FROM GUI INTO MYSQL
        try:
            connection = mysql.connector.connect(host='localhost',
                                                 database='Test',
                                                 user='root',
                                                 password='Supra777')

            mySql_insert_query = """INSERT INTO Asset_Category (idAsset_Category, name)
                                    VALUES
                                    (NULL, '%s')""" % (str(self.get_asset_category_name()))
            cursor = connection.cursor()
            cursor.execute(mySql_insert_query)
            connection.commit()
            print(cursor.rowcount, "Record inserted successfully into Asset_Category table")
            cursor.close()

        except mysql.connector.Error as error:
            messagebox.showwarning('WARNING: ', "Failed to insert data into Asset Category table.\nTry again.\nError
Code: {}".format(error))
```

```
        print("Failed to insert data into Asset_Category table {}".format(error))

    finally:
        if (connection.is_connected()):
            connection.close()
            print("MySQL connection is closed")
    return


##############################################################################
class BankAccount_ENTRY():
    def __init__(self):
        self.entryBank_Account_owner              = np.nan
        self.entryBank_Account_type               = np.nan
        self.entryBank_Account_brand              = np.nan
        self.entryBank_Account_description        = np.nan
        self.entryBank_Account_interest_rate      = np.nan
        self.entryBank_Account_date_acquired      = np.nan
        self.entryBank_Account_balance            = np.nan
        return

    #### Accessors and Mutators
    def set_bank_account_owner(self, input):
        self.entryBank_Account_owner = input
        return

    def set_bank_account_type(self, input):
        self.entryBank_Account_type  = input
        return

    def set_bank_account_brand(self, input):
        self.entryBank_Account_brand = input
        return

    def set_bank_account_description(self, input):
        self.entryBank_Account_description = input
        return

    def set_bank_account_interest_rate(self, input):
        self.entryBank_Account_interest_rate = input
        return

    def set_bank_account_date_acquired(self, input):
        self.entryBank_Account_date_acquired = input
        return

    def set_bank_account_balance(self, input):
        self.entryBank_Account_balance = input
        return

    ##########################################################################

    def get_bank_account_owner(self):

        return self.entryBank_Account_owner
```

```
def get_bank_account_type(self):

    return self.entryBank_Account_type

def get_bank_account_brand(self):

    return self.entryBank_Account_brand

def get_bank_account_description(self):

    return self.entryBank_Account_description

def get_bank_account_interest_rate(self):

    return self.entryBank_Account_interest_rate

def get_bank_account_date_acquired(self):

    return self.entryBank_Account_date_acquired

def get_bank_account_balance(self):

    return self.entryBank_Account_balance

#######################################################################

def SUBMIT_BANK_ACCOUNT(self):
    ### THIS IS THE FUNCTION THAT WILL WRRITE THE MEMBER DATA OUT TO SQL
    print(self.get_bank_account_owner(),
          self.get_bank_account_type(),
          self.get_bank_account_brand(),
          self.get_bank_account_description(),
          self.get_bank_account_interest_rate(),
          self.get_bank_account_date_acquired(),
          self.get_bank_account_balance())

    #POPUP CONFIRMATION WINDOW
    warningMessage = "Bank Account Owner: " + self.entryBank_Account_owner
    warningMessage += "\nBank Account Type: " + self.entryBank_Account_type
    warningMessage += "\nBank Account Brand: " + self.entryBank_Account_brand
    warningMessage += "\nBank Account Description: " + self.entryBank_Account_description
    warningMessage += "\nInterest Rate: " + self.entryBank_Account_interest_rate
    warningMessage += "\nDate Acquired: " + self.entryBank_Account_date_acquired
    warningMessage += "\nAcount Balance: " + self.entryBank_Account_balance
    window = Tk()
    window.eval('tk::PlaceWindow %s center' % window.winfo_toplevel())
    window.withdraw()
    messagebox.showwarning('Stockwell_Financial Database Bank Account Entry', warningMessage)
    window.deiconify()
    window.destroy()

    #INPUTS ASSET DATA FROM GUI INTO MYSQL
    try:
        connection = mysql.connector.connect(host='localhost',
```

```
                                    database='Test',
                                    user='root',
                                    password='Supra777')

        mySql_insert_query = """INSERT INTO Bank_Account (idBank_Account, owner,
                                        type, brand, description,
                                        interest_rate, date_acquired,
                                        balance)
                                VALUES
                                (NULL, '%s', '%s', '%s', '%s', '%s', '%s','%s') """ % (str(self.get_bank_account_owner()),
                                                        str(self.get_bank_account_type()),
                                                        str(self.get_bank_account_brand()),
                                                        str(self.get_bank_account_description()),
                                                        str(self.get_bank_account_interest_rate()),
                                                        str(self.get_bank_account_date_acquired()),
                                                        str(self.get_bank_account_balance()))
        cursor = connection.cursor()
        cursor.execute(mySql_insert_query)
        connection.commit()
        print(cursor.rowcount, "Record inserted successfully into Bank_Account table")
        cursor.close()

    except mysql.connector.Error as error:
        messagebox.showwarning('WARNING: ', "Failed to insert data into Bank Account table.\nTry again.\nError
Code: {}".format(error))
        print("Failed to insert data into Bank_Account table {}".format(error))

    finally:
        if (connection.is_connected()):
            connection.close()
            print("MySQL connection is closed")
    return


##############################################################################
class BankTransaction_ENTRY():
    def __init__(self):
        self.entryBank_Transaction_fk_bank_account_ID          = np.nan
        self.entryBank_Transaction_date                        = np.nan
        self.entryBank_Transaction_description                 = np.nan
        self.entryBank_Transaction_deposit                     = np.nan
        self.entryBank_Transaction_fk_asset_ID                 = np.nan
        self.entryBank_Transaction_fk_asset_category_ID        = np.nan
        self.entryBank_Transaction_withdrawal                  = np.nan
        self.entryBank_Transaction_fk_liability_category_ID    = np.nan
        self.entryBank_Transaction_fk_credit_card_ID           = np.nan
        return

    #### Accessors and Mutators
    def set_bank_transaction_fk_bank_account_ID(self, input):
        self.entryBank_Transaction_fk_bank_account_ID = input
        return

    def set_bank_transaction_date(self, input):
        self.entryBank_Transaction_date  = input
```

```
    return

def set_bank_transaction_description(self, input):
    self.entryBank_Transaction_description = input
    return

def set_bank_transaction_deposit(self, input):
    self.entryBank_Transaction_deposit = input
    return

def set_bank_transaction_fk_asset_ID(self, input):
    self.entryBank_Transaction_fk_asset_ID = input
    return

def set_bank_transaction_fk_asset_category_ID(self, input):
    self.entryBank_Transaction_fk_asset_category_ID = input
    return

def set_bank_transaction_withdrawal(self, input):
    self.entryBank_Transaction_withdrawal = input
    return

def set_bank_transaction_fk_liability_category_ID(self, input):
    self.entryBank_Transaction_fk_liability_category_ID = input
    return

def set_bank_transaction_fk_credit_card_ID(self, input):
    self.entryBank_Transaction_fk_credit_card_ID = input
    return

#######################################################################

def get_bank_transaction_fk_bank_account_ID(self):

    return self.entryBank_Transaction_fk_bank_account_ID

def get_bank_transaction_date(self):

    return self.entryBank_Transaction_date

def get_bank_transaction_description(self):

    return self.entryBank_Transaction_description

def get_bank_transaction_deposit(self):

    return self.entryBank_Transaction_deposit

def get_bank_transaction_fk_asset_ID(self):

    return self.entryBank_Transaction_fk_asset_ID

def get_bank_transaction_fk_asset_category_ID(self):

    return self.entryBank_Transaction_fk_asset_category_ID
```

```
    def get_bank_transaction_withdrawal(self):

        return self.entryBank_Transaction_withdrawal

    def get_bank_transaction_fk_liability_category_ID(self):

        return self.entryBank_Transaction_fk_liability_category_ID

    def get_bank_transaction_fk_credit_card_ID(self):

        return self.entryBank_Transaction_fk_credit_card_ID

    ######################################################################
    def SUBMIT_BANK_TRANSACTION(self):
        ### THIS IS THE FUNCTION THAT WILL WRRITE THE MEMBER DATA OUT TO SQL
        print(self.get_bank_transaction_fk_bank_account_ID(),
              self.get_bank_transaction_date(),
              self.get_bank_transaction_description(),
              self.get_bank_transaction_deposit(),
              self.get_bank_transaction_fk_asset_ID(),
              self.get_bank_transaction_fk_asset_category_ID(),
              self.get_bank_transaction_withdrawal(),
              self.get_bank_transaction_fk_liability_category_ID(),
              self.get_bank_transaction_fk_credit_card_ID())

        #POPUP CONFIRMATION WINDOW
        warningMessage = "Bank Account ID: " + self.entryBank_Transaction_fk_bank_account_ID
        warningMessage += "\nTransacction Date: " + self.entryBank_Transaction_date
        warningMessage += "\nTransaction Description: " + self.entryBank_Transaction_description
        warningMessage += "\nDeposit: " + self.entryBank_Transaction_deposit
        warningMessage += "\nAsset ID: " + self.entryBank_Transaction_fk_asset_ID
        warningMessage += "\nAsset Category ID: " + self.entryBank_Transaction_fk_asset_category_ID
        warningMessage += "\nWithdrawal: " + self.entryBank_Transaction_withdrawal
        warningMessage += "\nLiability Category ID: " + self.entryBank_Transaction_fk_liability_category_ID
        warningMessage += "\nCredit Card ID: " + self.entryBank_Transaction_fk_credit_card_ID
        window = Tk()
        window.eval('tk::PlaceWindow %s center' % window.winfo_toplevel())
        window.withdraw()
        messagebox.showwarning('Stockwell_Financial Database Bank Transaction Entry', warningMessage)
        window.deiconify()
        window.destroy()

        #INPUTS ASSET DATA FROM GUI INTO MYSQL
        try:
            connection = mysql.connector.connect(host='localhost',
                                                 database='Test',
                                                 user='root',
                                                 password='Supra777')

            #THIS IS ENTERING WITHDRAWALS
            if(self.entryBank_Transaction_fk_asset_ID == "None") & (self.entryBank_Transaction_fk_asset_category_ID ==
"None"):
                mySql_insert_query = """INSERT INTO Bank_Transaction (idBank_Transaction,
                                        fk_bank_account_ID, date, description, deposit,
```

```
                                    fk_asset_ID, fk_asset_category_ID, withdrawal,
                                    fk_liability_category_ID, fk_credit_card_ID)
                                   VALUES
                                   (NULL, '%s', '%s', '%s', 00.00, NULL, NULL, '%s', '%s', '%s') """ %
(str(self.get_bank_transaction_fk_bank_account_ID()),
                                                                    str(self.get_bank_transaction_date()),
                                                                    str(self.get_bank_transaction_description()),
                                                                    str(self.get_bank_transaction_withdrawal()),
                                                                    str(self.get_bank_transaction_fk_liability_category_ID()),
                                                                    str(self.get_bank_transaction_fk_credit_card_ID()))
        if(self.entryBank_Transaction_fk_asset_ID == "None") & (self.entryBank_Transaction_fk_asset_category_ID ==
"None") & (self.entryBank_Transaction_fk_credit_card_ID == "None"):
            mySql_insert_query = """INSERT INTO Bank_Transaction (idBank_Transaction,
                                    fk_bank_account_ID, date, description, deposit,
                                    fk_asset_ID, fk_asset_category_ID, withdrawal,
                                    fk_liability_category_ID, fk_credit_card_ID)
                                   VALUES
                                   (NULL, '%s', '%s', '%s', 00.00, NULL, NULL, '%s', '%s', NULL) """ %
(str(self.get_bank_transaction_fk_bank_account_ID()),
                                                                    str(self.get_bank_transaction_date()),
                                                                    str(self.get_bank_transaction_description()),
                                                                    str(self.get_bank_transaction_withdrawal()),
                                                                    str(self.get_bank_transaction_fk_liability_category_ID()))
        #THIS IS ENTERING DEPOSITS
        if(self.entryBank_Transaction_fk_liability_category_ID == "None") &
(self.entryBank_Transaction_fk_credit_card_ID == "None"):
            mySql_insert_query = """INSERT INTO Bank_Transaction (idBank_Transaction,
                                     fk_bank_account_ID, date, description, deposit,
                                     fk_asset_ID, fk_asset_category_ID, withdrawal,
                                     fk_liability_category_ID, fk_credit_card_ID)
                                    VALUES
                                    (NULL, '%s', '%s', '%s', '%s', '%s', '%s', 00.00, NULL, NULL) """ %
(str(self.get_bank_transaction_fk_bank_account_ID()),
                                                                      str(self.get_bank_transaction_date()),
                                                                      str(self.get_bank_transaction_description()),
                                                                      str(self.get_bank_transaction_deposit()),
                                                                      str(self.get_bank_transaction_fk_asset_ID()),
                                                                      str(self.get_bank_transaction_fk_asset_category_ID()))
        if(self.entryBank_Transaction_fk_liability_category_ID == "None") &
(self.entryBank_Transaction_fk_credit_card_ID == "None") & (self.entryBank_Transaction_fk_asset_ID == "None"):
            mySql_insert_query = """INSERT INTO Bank_Transaction (idBank_Transaction,
                                     fk_bank_account_ID, date, description, deposit,
                                     fk_asset_ID, fk_asset_category_ID, withdrawal,
                                     fk_liability_category_ID, fk_credit_card_ID)
                                    VALUES
                                    (NULL, '%s', '%s', '%s', '%s', NULL, '%s', 00.00, NULL, NULL) """ %
(str(self.get_bank_transaction_fk_bank_account_ID()),
                                                                      str(self.get_bank_transaction_date()),
                                                                      str(self.get_bank_transaction_description()),
                                                                      str(self.get_bank_transaction_deposit()),
                                                                      str(self.get_bank_transaction_fk_asset_category_ID()))
        cursor = connection.cursor()
        cursor.execute(mySql_insert_query)
        connection.commit()
        print(cursor.rowcount, "Record inserted successfully into Bank_Transaction table")
```

```
        cursor.close()

    except mysql.connector.Error as error:
        messagebox.showwarning('WARNING: ', "Failed to insert data into Bank Transaction table.\nTry again.\nError
Code: {}".format(error))
        print("Failed to insert data into Bank_Transaction table {}".format(error))

    finally:
        if (connection.is_connected()):
            connection.close()
            print("MySQL connection is closed")
    return


##############################################################################
class CreditCard_ENTRY():
    def __init__(self):
        self.entryCredit_Card_owner              = np.nan
        self.entryCredit_Card_brand              = np.nan
        self.entryCredit_Card_description        = np.nan
        self.entryCredit_Card_interest_rate      = np.nan
        self.entryCredit_Card_date_acquired      = np.nan
        self.entryCredit_Card_date_expires       = np.nan
        self.entryCredit_Card_balance            = np.nan
        self.entryCredit_Card_credit_limit       = np.nan
        return

    #### Accessors and Mutators
    def set_credit_card_owner(self, input):
        self.entryCredit_Card_owner = input
        return

    def set_credit_card_brand(self, input):
        self.entryCredit_Card_brand  = input
        return

    def set_credit_card_description(self, input):
        self.entryCredit_Card_description = input
        return

    def set_credit_card_interest_rate(self, input):
        self.entryCredit_Card_interest_rate = input
        return

    def set_credit_card_date_acquired(self, input):
        self.entryCredit_Card_date_acquired = input
        return

    def set_credit_card_date_expires(self, input):
        self.entryCredit_Card_date_expires = input
        return

    def set_credit_card_balance(self, input):
        self.entryCredit_Card_balance = input
        return
```

```
def set_credit_card_limit(self, input):
    self.entryCredit_Card_credit_limit = input
    return

#######################################################################

def get_credit_card_owner(self):

    return self.entryCredit_Card_owner

def get_credit_card_brand(self):

    return self.entryCredit_Card_brand

def get_credit_card_description(self):

    return self.entryCredit_Card_description

def get_credit_card_interest_rate(self):

    return self.entryCredit_Card_interest_rate

def get_credit_card_date_acquired(self):

    return self.entryCredit_Card_date_acquired

def get_credit_card_date_expires(self):

    return self.entryCredit_Card_date_expires

def get_credit_card_balance(self):

    return self.entryCredit_Card_balance

def get_credit_card_credit_limit(self):

    return self.entryCredit_Card_credit_limit

#######################################################################
def SUBMIT_CREDIT_CARD(self):
    ### THIS IS THE FUNCTION THAT WILL WRRITE THE MEMBER DATA OUT TO SQL
    print(self.get_credit_card_owner(),
          self.get_credit_card_brand(),
          self.get_credit_card_description(),
          self.get_credit_card_interest_rate(),
          self.get_credit_card_date_acquired(),
          self.get_credit_card_date_expires(),
          self.get_credit_card_balance(),
          self.get_credit_card_credit_limit())

    #POPUP CONFIRMATION WINDOW
    warningMessage = "Credit Card Owner: " + self.entryCredit_Card_owner
    warningMessage += "\nCredit Card Brand:  " + self.entryCredit_Card_brand
    warningMessage += "\nCredit Card Description : " + self.entryCredit_Card_description
```

```
        warningMessage += "\nCredit Card Interest Rate: " + self.entryCredit_Card_interest_rate
        warningMessage += "\nDate Acquired: " + self.entryCredit_Card_date_acquired
        warningMessage += "\nDate Expires: " + self.entryCredit_Card_date_expires
        warningMessage += "\nCard Balance: " + self.entryCredit_Card_balance
        warningMessage += "\nCredit Limit: " + self.entryCredit_Card_credit_limit
        window = Tk()
        window.eval('tk::PlaceWindow %s center' % window.winfo_toplevel())
        window.withdraw()
        messagebox.showwarning('Stockwell_Financial Database Credit Card Entry', warningMessage)
        window.deiconify()
        window.destroy()

        #INPUTS ASSET DATA FROM GUI INTO MYSQL
        try:
            connection = mysql.connector.connect(host='localhost',
                                                 database='Test',
                                                 user='root',
                                                 password='Supra777')

            mySql_insert_query = """INSERT INTO Credit_Card (idCredit_Card,
                                    owner, brand, description, interest_rate,
                                    date_acquired, date_expires, balance,
                                    credit_limit)
                                   VALUES
                                   (NULL, '%s', '%s', '%s', '%s', '%s', '%s', '%s', '%s') """ %
(str(self.get_credit_card_owner()),
                                                                    str(self.get_credit_card_brand()),
                                                                    str(self.get_credit_card_description()),
                                                                    str(self.get_credit_card_interest_rate()),
                                                                    str(self.get_credit_card_date_acquired()),
                                                                    str(self.get_credit_card_date_expires()),
                                                                    str(self.get_credit_card_balance()),
                                                                    str(self.get_credit_card_credit_limit()))
            cursor = connection.cursor()
            cursor.execute(mySql_insert_query)
            connection.commit()
            print(cursor.rowcount, "Record inserted successfully into Credit_Card table")
            cursor.close()

        except mysql.connector.Error as error:
            messagebox.showwarning('WARNING: ', "Failed to insert data into Credit Card table.\nTry again.\nError
Code: {}".format(error))
            print("Failed to insert data into Credit_Card table {}".format(error))

        finally:
            if (connection.is_connected()):
                connection.close()
                print("MySQL connection is closed")
        return


##############################################################################
class CreditTransaction_ENTRY():
    def __init__(self):
        self.entryCredit_Transaction_fk_credit_card_ID          = np.nan
```

```
    self.entryCredit_Transaction_description                = np.nan
    self.entryCredit_Transaction_fk_liability_category_ID    = np.nan
    self.entryCredit_Transaction_charge_date                 = np.nan
    self.entryCredit_Transaction_charge                      = np.nan
    return

#### Accessors and Mutators
def set_credit_transaction_fk_credit_card_ID(self, input):
    self.entryCredit_Transaction_fk_credit_card_ID = input
    return

def set_credit_transaction_description(self, input):
    self.entryCredit_Transaction_description  = input
    return

def set_credit_transaction_fk_liability_category_ID(self, input):
    self.entryCredit_Transaction_fk_liability_category_ID = input
    return

def set_credit_transaction_charge_date(self, input):
    self.entryCredit_Transaction_charge_date = input
    return

def set_credit_transaction_charge(self, input):
    self.entryCredit_Transaction_charge = input
    return

#######################################################################

def get_credit_transaction_fk_credit_card_ID(self):

    return self.entryCredit_Transaction_fk_credit_card_ID

def get_credit_transaction_description(self):

    return self.entryCredit_Transaction_description

def get_credit_transaction_fk_liability_category_ID(self):

    return self.entryCredit_Transaction_fk_liability_category_ID

def get_credit_transaction_charge_date(self):

    return self.entryCredit_Transaction_charge_date

def get_credit_transaction_charge(self):

    return self.entryCredit_Transaction_charge

#######################################################################
def SUBMIT_CREDIT_TRANSACTION(self):
    ### THIS IS THE FUNCTION THAT WILL WRRITE THE MEMBER DATA OUT TO SQL
    print(self.get_credit_transaction_fk_credit_card_ID(),
          self.get_credit_transaction_description(),
          self.get_credit_transaction_fk_liability_category_ID(),
```

```
            self.get_credit_transaction_charge_date(),
            self.get_credit_transaction_charge())

        #POPUP CONFIRMATION WINDOW
        warningMessage = "Credit Card ID: " + self.entryCredit_Transaction_fk_credit_card_ID
        warningMessage += "\nCredit Transaction Description:  " + self.entryCredit_Transaction_description
        warningMessage += "\nLiability Category: " + self.entryCredit_Transaction_fk_liability_category_ID
        warningMessage += "\nTransaction Date: " + self.entryCredit_Transaction_charge_date
        warningMessage += "\nCharge Amount: " + self.entryCredit_Transaction_charge
        window = Tk()
        window.eval('tk::PlaceWindow %s center' % window.winfo_toplevel())
        window.withdraw()
        messagebox.showwarning('Stockwell_Financial Database Credit Transaction Entry', warningMessage)
        window.deiconify()
        window.destroy()

        #INPUTS ASSET DATA FROM GUI INTO MYSQL
        try:
            connection = mysql.connector.connect(host='localhost',
                                                 database='Test',
                                                 user='root',
                                                 password='Supra777')


            mySql_insert_query = """INSERT INTO Credit_Transaction (idCredit_Transaction,
                                     fk_credit_card_ID, description,
                                     fk_liability_category_ID, charge_date, charge)
                                    VALUES
                                    (NULL, '%s', '%s', '%s', '%s', '%s') """ %
(str(self.get_credit_transaction_fk_credit_card_ID()),
                                                                  str(self.get_credit_transaction_description()),
                                                                  str(self.get_credit_transaction_fk_liability_category_ID()),
                                                                  str(self.get_credit_transaction_charge_date()),
                                                                  str(self.get_credit_transaction_charge()))
            cursor = connection.cursor()
            cursor.execute(mySql_insert_query)
            connection.commit()
            print(cursor.rowcount, "Record inserted successfully into Credit_Transaction table")
            cursor.close()

        except mysql.connector.Error as error:
            messagebox.showwarning('WARNING: ', "Failed to insert data into Credit Transaction table.\nTry again.
\nError Code: {}".format(error))
            print("Failed to insert data into Credit_Transaction table {}".format(error))

        finally:
            if (connection.is_connected()):
                connection.close()
                print("MySQL connection is closed")
        return


##############################################################################
class LiabilityCategory_ENTRY():
    def __init__(self):
```

```
        self.entryLiability_Category_name = np.nan
        return

    #### Accessors and Mutators are legit
    def set_liability_category_name(self, input):
        self.entryLiability_Category_name = input
        return

    ######################################################################

    def get_liability_category_name(self):

        return self.entryLiability_Category_name

    ######################################################################

    def SUBMIT_LIABILITY_CATEGORY(self):
        ### THIS IS THE FUNCTION THAT WILL WRRITE THE MEMBER DATA OUT TO SQL
        print(self.get_liability_category_name())

        #POPUP CONFIRMATION WINDOW
        warningMessage = "Liability Category Name: " + self.entryLiability_Category_name
        window = Tk()
        window.eval('tk::PlaceWindow %s center' % window.winfo_toplevel())
        window.withdraw()
        messagebox.showwarning('Stockwell_Financial Database Liability Category Entry', warningMessage)
        window.deiconify()
        window.destroy()

        #INPUTS ASSET DATA FROM GUI INTO MYSQL
        try:
            connection = mysql.connector.connect(host='localhost',
                                                 database='Test',
                                                 user='root',
                                                 password='Supra777')

            mySql_insert_query = """INSERT INTO Liability_Category (idLiability_Category,
                                     name)
                                    VALUES
                                    (NULL, '%s') """ % (str(self.get_liability_category_name()))
            cursor = connection.cursor()
            cursor.execute(mySql_insert_query)
            connection.commit()
            print(cursor.rowcount, "Record inserted successfully into Liability_Category table")
            cursor.close()

        except mysql.connector.Error as error:
            messagebox.showwarning('WARNING: ', "Failed to insert data into Liability Category table.\nTry again.\nError
Code: {}".format(error))
            print("Failed to insert data into Liability_Category table {}".format(error))

finally:
        if (connection.is_connected()):
            connection.close()
            print("MySQL connection is closed")
    return
```

```
##############################################################################
#    Author: Austin Stockwell
#    Date: 03-02-2020
#    Description:  This file contains methods that recieve the appropriate
#                  selected value of the GUIs menu objects and returns
#                  a single number that the MySQL database can understand.
#    File: MenuFunctions.py
##############################################################################
from entry import *
from page_interface import *
######################## ######################## ################################
#Asset_Category Table Functions
######################## ######################## ################################
def GetAsset_AssetCategoryMenu(varAsset_fk_asset_category_ID):
    if  varAsset_fk_asset_category_ID.get() == '(1) Stock':
        passThis = varAsset_fk_asset_category_ID.get()
        passThis = '1'
        return passThis
    if  varAsset_fk_asset_category_ID.get() == '(2) Bond':
        passThis = varAsset_fk_asset_category_ID.get()
        passThis = '2'
        return passThis
    if  varAsset_fk_asset_category_ID.get() == '(3) Index Fund':
        passThis = varAsset_fk_asset_category_ID.get()
        passThis = '3'
        return passThis
    if  varAsset_fk_asset_category_ID.get() == '(4) Mutual Fund':
        passThis = varAsset_fk_asset_category_ID.get()
        passThis = '4'
        return passThis
    if  varAsset_fk_asset_category_ID.get() == '(5) IRA':
        passThis = varAsset_fk_asset_category_ID.get()
        passThis = '5'
        return passThis
    if  varAsset_fk_asset_category_ID.get() == '(6) Roth IRA':
        passThis = varAsset_fk_asset_category_ID.get()
        passThis = '6'
        return passThis
    if  varAsset_fk_asset_category_ID.get() == '(7) 401k':
        passThis = varAsset_fk_asset_category_ID.get()
        passThis = '7'
        return passThis
    if  varAsset_fk_asset_category_ID.get() == '(8) Roth 401k':
        passThis = varAsset_fk_asset_category_ID.get()
        passThis = '8'
        return passThis
    if  varAsset_fk_asset_category_ID.get() == '(9) CD':
        passThis = varAsset_fk_asset_category_ID.get()
        passThis = '9'
        return passThis
    if  varAsset_fk_asset_category_ID.get() == '(10) Commodity':
        passThis = varAsset_fk_asset_category_ID.get()
        passThis = '10'
        return passThis
    if  varAsset_fk_asset_category_ID.get() == '(11) Cryptocurrency':
```

```
        passThis = varAsset_fk_asset_category_ID.get()
        passThis = '11'
        return passThis
    if  varAsset_fk_asset_category_ID.get() == '(12) Real Estate':
        passThis = varAsset_fk_asset_category_ID.get()
        passThis = '12'
        return passThis

######################## ######################## ################################
# Bank_Transaction Table Functions
######################## ######################## ################################
def GetBankTransaction_BankAccountMenu(varBank_Transaction_fk_bank_account_ID):
    if  varBank_Transaction_fk_bank_account_ID.get() == '(1) Austin Centier Saving':
        passThis = varBank_Transaction_fk_bank_account_ID.get()
        passThis = '1'
        return passThis
    if  varBank_Transaction_fk_bank_account_ID.get() == '(2) Austin Centier Checking':
        passThis = varBank_Transaction_fk_bank_account_ID.get()
        passThis = '2'
        return passThis
    if  varBank_Transaction_fk_bank_account_ID.get() == '(3) Belle CHASE Saving':
        passThis = varBank_Transaction_fk_bank_account_ID.get()
        passThis = '3'
        return passThis
    if  varBank_Transaction_fk_bank_account_ID.get() == '(4) Joint CHASE Checking':
        passThis = varBank_Transaction_fk_bank_account_ID.get()
        passThis = '4'
        return passThis
    if  varBank_Transaction_fk_bank_account_ID.get() == '(5) Joint CHASE EMERGENCY SAVING':
        passThis = varBank_Transaction_fk_bank_account_ID.get()
        passThis = '5'
        return passThis

def GetBankTransaction_AssetMenu(varBank_Transaction_fk_asset_ID):
    if  varBank_Transaction_fk_asset_ID.get() == "None":
        passThis = varBank_Transaction_fk_asset_ID.get()
        passThis = "None"
        return passThis
    if  varBank_Transaction_fk_asset_ID.get() == '(1) Ethereum':
        passThis = varBank_Transaction_fk_asset_ID.get()
        passThis = '1'
        return passThis
    if  varBank_Transaction_fk_asset_ID.get() == '(2) Ripple':
        passThis = varBank_Transaction_fk_asset_ID.get()
        passThis = '2'
        return passThis

def GetBankTransaction_AssetCategoryMenu(varBank_Transaction_fk_asset_category_ID):
    if  varBank_Transaction_fk_asset_category_ID.get() == "None":
        passThis = varBank_Transaction_fk_asset_category_ID.get()
        passThis = "None"
        return passThis
    if  varBank_Transaction_fk_asset_category_ID.get() == '(1) Stock':
        passThis = varBank_Transaction_fk_asset_category_ID.get()
        passThis = '1'
```

```
        return passThis
    if  varBank_Transaction_fk_asset_category_ID.get() == '(2) Bond':
        passThis = varBank_Transaction_fk_asset_category_ID.get()
        passThis = '2'
        return passThis
    if  varBank_Transaction_fk_asset_category_ID.get() == '(3) Index Fund':
        passThis = varBank_Transaction_fk_asset_category_ID.get()
        passThis = '3'
        return passThis
    if  varBank_Transaction_fk_asset_category_ID.get() == '(4) Mutual Fund':
        passThis = varBank_Transaction_fk_asset_category_ID.get()
        passThis = '4'
        return passThis
    if  varBank_Transaction_fk_asset_category_ID.get() == '(5) IRA':
        passThis = varBank_Transaction_fk_asset_category_ID.get()
        passThis = '5'
        return passThis
    if  varBank_Transaction_fk_asset_category_ID.get() == '(6) Roth IRA':
        passThis = varBank_Transaction_fk_asset_category_ID.get()
        passThis = '6'
        return passThis
    if  varBank_Transaction_fk_asset_category_ID.get() == '(7) 401k':
        passThis = varBank_Transaction_fk_asset_category_ID.get()
        passThis = '7'
        return passThis
    if  varBank_Transaction_fk_asset_category_ID.get() == '(8) Roth 401k':
        passThis = varBank_Transaction_fk_asset_category_ID.get()
        passThis = '8'
        return passThis
    if  varBank_Transaction_fk_asset_category_ID.get() == '(9) CD':
        passThis = varBank_Transaction_fk_asset_category_ID.get()
        passThis = '9'
        return passThis
    if  varBank_Transaction_fk_asset_category_ID.get() == '(10) Commodity':
        passThis = varBank_Transaction_fk_asset_category_ID.get()
        passThis = '10'
        return passThis
    if  varBank_Transaction_fk_asset_category_ID.get() == '(11) Cryptocurrency':
        passThis = varBank_Transaction_fk_asset_category_ID.get()
        passThis = '11'
        return passThis
    if  varBank_Transaction_fk_asset_category_ID.get() == '(12) Real Estate':
        passThis = varBank_Transaction_fk_asset_category_ID.get()
        passThis = '12'
        return passThis
    if  varBank_Transaction_fk_asset_category_ID.get() == '(100) Paycheck':
        passThis = varBank_Transaction_fk_asset_category_ID.get()
        passThis = '100'
        return passThis

def GetBankTransaction_LiabilityCategoryMenu(varBank_Transaction_fk_liability_category_ID):
    if  varBank_Transaction_fk_liability_category_ID.get() == "None":
        passThis = varBank_Transaction_fk_liability_category_ID.get()
        passThis = "None"
        return passThis
```

```
if  varBank_Transaction_fk_liability_category_ID.get() == '(1) Rent':
    passThis = varBank_Transaction_fk_liability_category_ID.get()
    passThis = '1'
    return passThis
if  varBank_Transaction_fk_liability_category_ID.get() == '(2) Mortgage':
    passThis = varBank_Transaction_fk_liability_category_ID.get()
    passThis = '2'
    return passThis
if  varBank_Transaction_fk_liability_category_ID.get() == '(3) Water':
    passThis = varBank_Transaction_fk_liability_category_ID.get()
    passThis = '3'
    return passThis
if  varBank_Transaction_fk_liability_category_ID.get() == '(4) Electricity':
    passThis = varBank_Transaction_fk_liability_category_ID.get()
    passThis = '4'
    return passThis
if  varBank_Transaction_fk_liability_category_ID.get() == '(5) Home Insurance':
    passThis = varBank_Transaction_fk_liability_category_ID.get()
    passThis = '5'
    return passThis
if  varBank_Transaction_fk_liability_category_ID.get() == '(6) Trash':
    passThis = varBank_Transaction_fk_liability_category_ID.get()
    passThis = '6'
    return passThis
if  varBank_Transaction_fk_liability_category_ID.get() == '(7) Gas Bill':
    passThis = varBank_Transaction_fk_liability_category_ID.get()
    passThis = '7'
    return passThis
if  varBank_Transaction_fk_liability_category_ID.get() == '(8) Car Payment':
    passThis = varBank_Transaction_fk_liability_category_ID.get()
    passThis = '8'
    return passThis
if  varBank_Transaction_fk_liability_category_ID.get() == '(9) Car Insurance':
    passThis = varBank_Transaction_fk_liability_category_ID.get()
    passThis = '9'
    return passThis
if  varBank_Transaction_fk_liability_category_ID.get() == '(10) Gasoline':
    passThis = varBank_Transaction_fk_liability_category_ID.get()
    passThis = '10'
    return passThis
if  varBank_Transaction_fk_liability_category_ID.get() == '(11) Car Repairs':
    passThis = varBank_Transaction_fk_liability_category_ID.get()
    passThis = '11'
    return passThis
if  varBank_Transaction_fk_liability_category_ID.get() == '(12) Groceries':
    passThis = varBank_Transaction_fk_liability_category_ID.get()
    passThis = '12'
    return passThis
if  varBank_Transaction_fk_liability_category_ID.get() == '(13) Cellphone':
    passThis = varBank_Transaction_fk_liability_category_ID.get()
    passThis = '13'
    return passThis
if  varBank_Transaction_fk_liability_category_ID.get() == '(14) Wifi':
    passThis = varBank_Transaction_fk_liability_category_ID.get()
    passThis = '14'
```

```
        return passThis
    if  varBank_Transaction_fk_liability_category_ID.get() == '(15) Bachelor Degree Loan':
        passThis = varBank_Transaction_fk_liability_category_ID.get()
        passThis = '15'
        return passThis
    if  varBank_Transaction_fk_liability_category_ID.get() == '(16) Credit Card Payment':
        passThis = varBank_Transaction_fk_liability_category_ID.get()
        passThis = '16'
        return passThis
    if  varBank_Transaction_fk_liability_category_ID.get() == '(17) Books':
        passThis = varBank_Transaction_fk_liability_category_ID.get()
        passThis = '17'
        return passThis
    if  varBank_Transaction_fk_liability_category_ID.get() == '(18) Hobbies':
        passThis = varBank_Transaction_fk_liability_category_ID.get()
        passThis = '18'
        return passThis
    if  varBank_Transaction_fk_liability_category_ID.get() == '(19) Music':
        passThis = varBank_Transaction_fk_liability_category_ID.get()
        passThis = '19'
        return passThis
    if  varBank_Transaction_fk_liability_category_ID.get() == '(20) Restaurant':
        passThis = varBank_Transaction_fk_liability_category_ID.get()
        passThis = '20'
        return passThis
    if  varBank_Transaction_fk_liability_category_ID.get() == '(21) Entertainment':
        passThis = varBank_Transaction_fk_liability_category_ID.get()
        passThis = '21'
        return passThis
    if  varBank_Transaction_fk_liability_category_ID.get() == '(22) Clothing':
        passThis = varBank_Transaction_fk_liability_category_ID.get()
        passThis = '22'
        return passThis
    if  varBank_Transaction_fk_liability_category_ID.get() == '(23) Travel':
        passThis = varBank_Transaction_fk_liability_category_ID.get()
        passThis = '23'
        return passThis
    if  varBank_Transaction_fk_liability_category_ID.get() == '(24) Jewelery':
        passThis = varBank_Transaction_fk_liability_category_ID.get()
        passThis = '24'
        return passThis
    if  varBank_Transaction_fk_liability_category_ID.get() == '(25) Home Improvement':
        passThis = varBank_Transaction_fk_liability_category_ID.get()
        passThis = '25'
        return passThis
    if  varBank_Transaction_fk_liability_category_ID.get() == '(26) Self Improvement':
        passThis = varBank_Transaction_fk_liability_category_ID.get()
        passThis = '26'
        return passThis

def GetBankTransaction_CreditCardMenu(varBank_Transaction_fk_credit_card_ID):
    if  varBank_Transaction_fk_credit_card_ID.get() == "None":
        passThis = varBank_Transaction_fk_credit_card_ID.get()
        passThis = "None"
        return passThis
```

```
    if  varBank_Transaction_fk_credit_card_ID.get() == '(1) Austin: Sweetwater':
        passThis = varBank_Transaction_fk_credit_card_ID.get()
        passThis = '1'
        return passThis
    if  varBank_Transaction_fk_credit_card_ID.get() == '(2) Austin: Discover':
        passThis = varBank_Transaction_fk_credit_card_ID.get()
        passThis = '2'
        return passThis
    if  varBank_Transaction_fk_credit_card_ID.get() == '(3) Austin: CHASE Freedom':
        passThis = varBank_Transaction_fk_credit_card_ID.get()
        passThis = '3'
        return passThis
    if  varBank_Transaction_fk_credit_card_ID.get() == '(4) Austin: Citi':
        passThis = varBank_Transaction_fk_credit_card_ID.get()
        passThis = '4'
        return passThis
    if  varBank_Transaction_fk_credit_card_ID.get() == '(5) Belle: CHASE Freedom':
        passThis = varBank_Transaction_fk_credit_card_ID.get()
        passThis = '5'
        return passThis

######################## ######################## ################################
# Credit_Transaction Table
######################## ######################## ################################
def GetCreditTransaction_CreditCardMenu(varCredit_Transaction_fk_credit_card_ID):
    if  varCredit_Transaction_fk_credit_card_ID.get() == '(1) Austin: Sweetwater':
        passThis = varCredit_Transaction_fk_credit_card_ID.get()
        passThis = '1'
        return passThis
    if  varCredit_Transaction_fk_credit_card_ID.get() == '(2) Austin: Discover':
        passThis = varCredit_Transaction_fk_credit_card_ID.get()
        passThis = '2'
        return passThis
    if  varCredit_Transaction_fk_credit_card_ID.get() == '(3) Austin: CHASE Freedom':
        passThis = varCredit_Transaction_fk_credit_card_ID.get()
        passThis = '3'
        return passThis
    if  varCredit_Transaction_fk_credit_card_ID.get() == '(4) Austin: Citi':
        passThis = varCredit_Transaction_fk_credit_card_ID.get()
        passThis = '4'
        return passThis
    if  varCredit_Transaction_fk_credit_card_ID.get() == '(5) Belle: CHASE Freedom':
        passThis = varCredit_Transaction_fk_credit_card_ID.get()
        passThis = '5'
        return passThis

def GetCreditTransaction_LiabilityCategoryMenu(varCredit_Transaction_fk_liability_category_ID):
    if  varCredit_Transaction_fk_liability_category_ID.get() == '(1) Rent':
        passThis = varCredit_Transaction_fk_liability_category_ID.get()
        passThis = '1'
        return passThis
    if  varCredit_Transaction_fk_liability_category_ID.get() == '(2) Mortgage':
        passThis = varCredit_Transaction_fk_liability_category_ID.get()
        passThis = '2'
        return passThis
```

```
if  varCredit_Transaction_fk_liability_category_ID.get() == '(3) Water':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '3'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(4) Electricity':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '4'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(5) Home Insurance':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '5'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(6) Trash':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '6'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(7) Gas Bill':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '7'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(8) Car Payment':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '8'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(9) Car Insurance':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '9'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(10) Gasoline':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '10'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(11) Car Repairs':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '11'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(12) Groceries':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '12'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(13) Cellphone':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '13'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(14) Wifi':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '14'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(15) Bachelor Degree Loan':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '15'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(16) Credit Card Payment':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '16'
```

```
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(17) Books':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '17'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(18) Hobbies':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '18'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(19) Music':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '19'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(20) Restaurant':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '20'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(21) Entertainment':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '21'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(22) Clothing':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '22'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(23) Travel':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '23'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(24) Jewelery':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '24'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(25) Home Improvement':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '25'
    return passThis
if  varCredit_Transaction_fk_liability_category_ID.get() == '(26) Self Improvement':
    passThis = varCredit_Transaction_fk_liability_category_ID.get()
    passThis = '26'
    return passThis
```

7 Appendix B: Development Tools

The following software tools were utilized for the creation of this project:

- Python: 3.7.4
- MySQL Server: 5.7.19
- MySQL Workbench: 8.0.12
- TCL/TK: 8.6.9