Mini Project Programming Laboratory 1

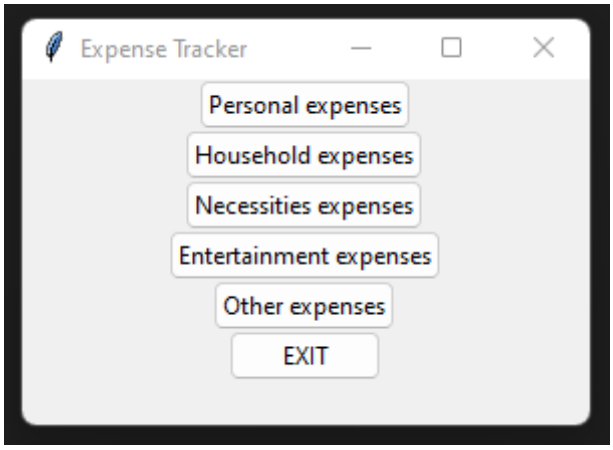
Project by:
Siddhi Anand – 16010421108
Saniya Sonawane – 16010421112

Problem Statement – Expense Tracker is a python programme which assists to manage expenses. It keeps a record of the expenses and helps to analyses all the expenditures efficiently and accurately. Tkinter module was used in the project's construction. The expenses are divided into five categories and exit option. When a category is selected then it is redirected to the corresponding window. Each category has same functionality by inserting, selecting all, deleting,clear and exiting. Tkinter Toplevel widget is used which creates a window on top of all other windows and provides some extra information. Sqlite3 is imported to create database and variables in it hold certain queries in database based on expense name.

Before using the program, the user needs to have three libraries install using ‘pip’.

- `pip install tkinter`
- `pip install tkinter.ttk`
- `pip install db`
- `pip install sqlite3`
- `pip install datetime`

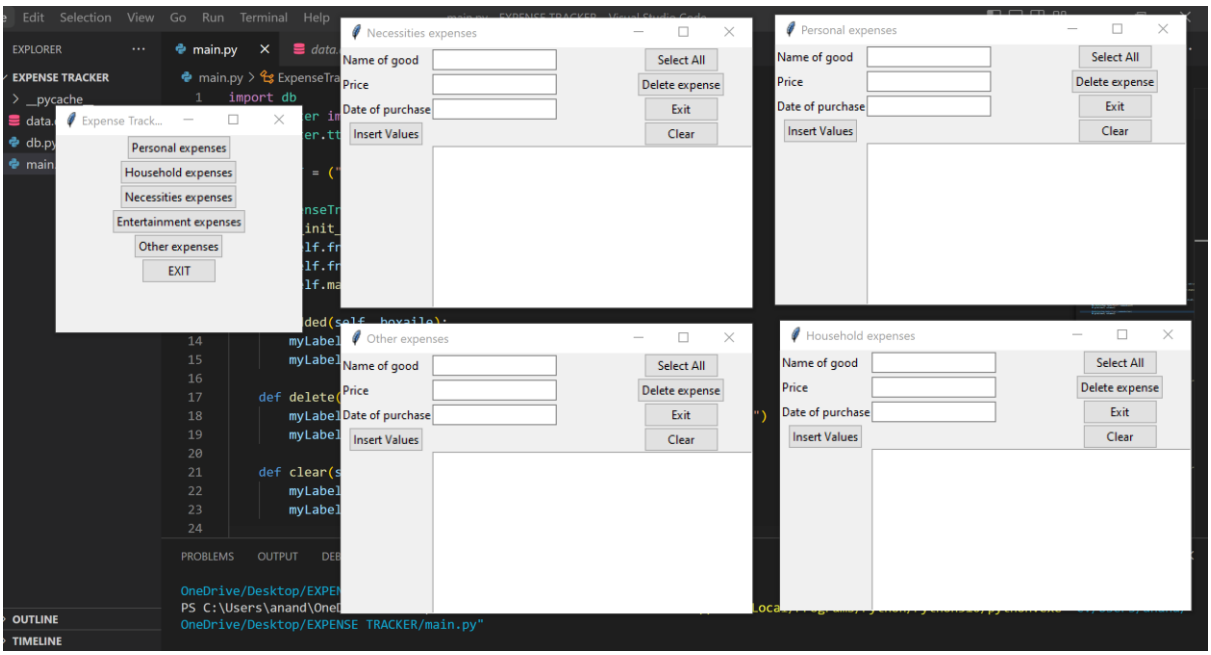
When we first run the code, we see this page:



- So, this is the Expense Tracker Window, the expenses are managed between five categories
    1. Personal expenses
    2. Household expenses
    3. Necessities expenses
    4. Entertainment expenses
    5. Other expenses

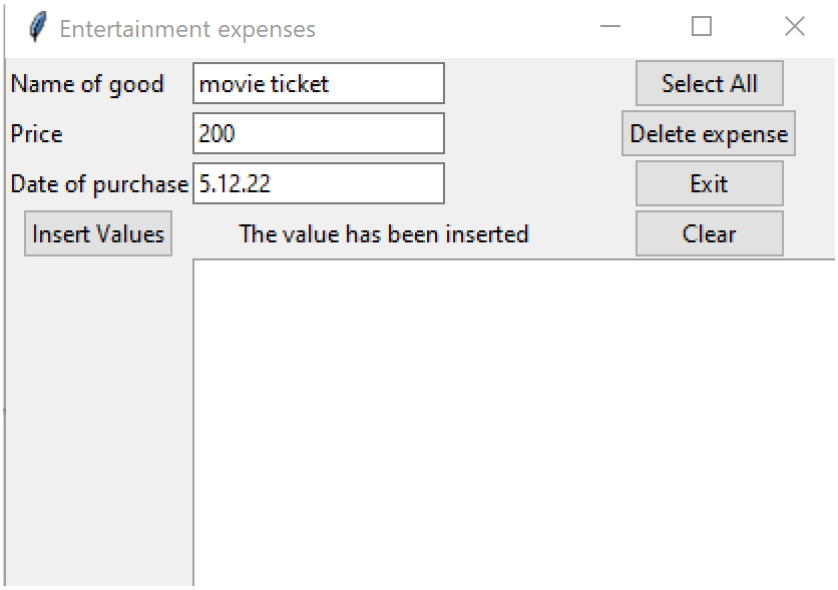
  And Lastly, by clicking Exit button on the GUI, the application stops and ends.

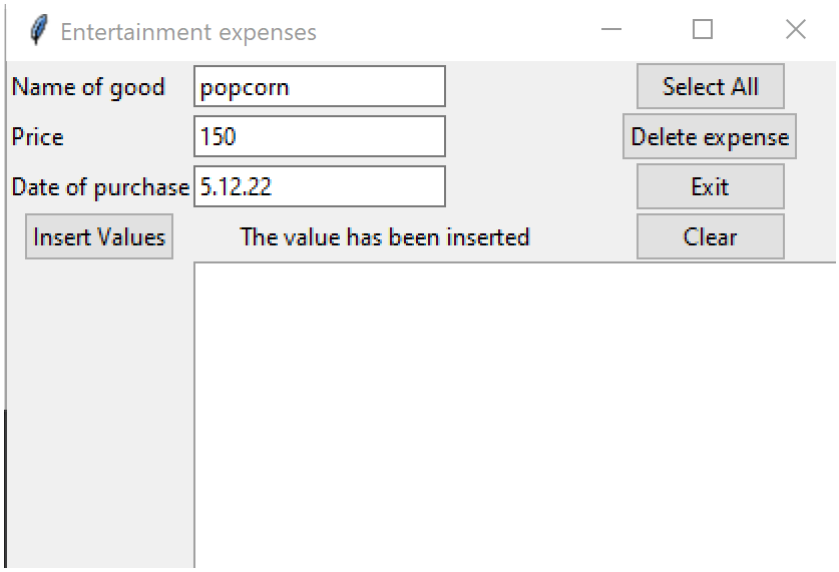
- A new window opens when one of the categories is selected. Each has the ability to insert, select, and delete records as well as exit the GUI. There is a clear button also which is provided in order to clear the entry field for user to add new/perform remaining operations.



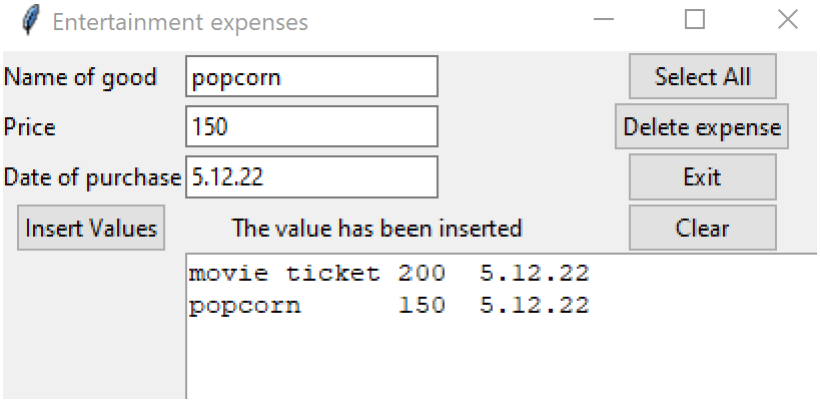
- After filling values in respective fields now let us look at the functionality of each button one by one

- When a user clicks the insert values button, the corresponding data is saved in the database and a message is shown.

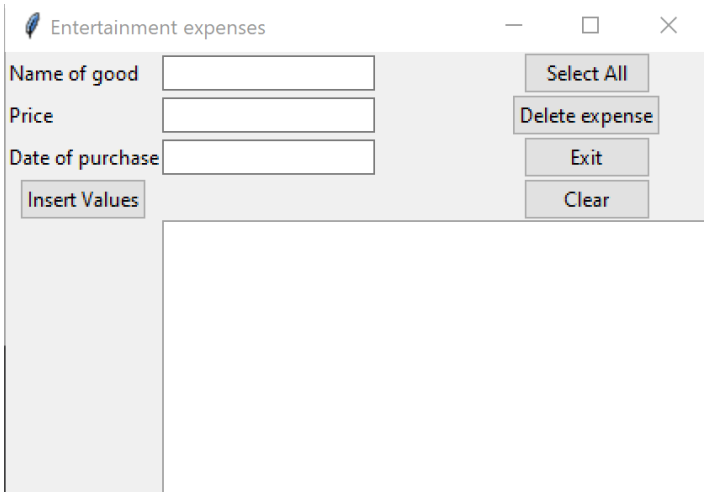




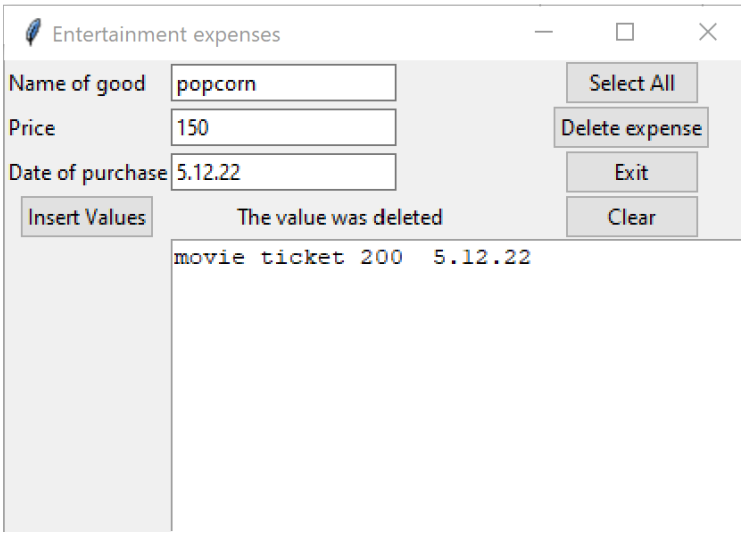
- On clicking the select all button the data filled is displayed in box.



- On clicking the clear button, the entry fields are reset.



- On clicking the delete expense button the data previously filled is deleted from the database.



- On clicking on exit button, the window is closed.

Code:

```
import db
from tkinter import *
from tkinter.ttk import *

LARGE_FONT = ("Verdana", 32)

class ExpenseTracker:
    def __init__(self, master):
        self.frame = Frame(master)
        self.frame.pack()
        self.main_window()

    def added(self, boxaile):
        myLabel = Label(boxaile, text="The value has been inserted")
        myLabel.grid(row=4, column=1)

    def delete(self, boxaile):
        myLabel = Label(boxaile, text="    The value was deleted     ")
        myLabel.grid(row=4, column=1)

    def clear(self, boxaile):
        myLabel = Label(boxaile, text="    Field Cleared.     ")
        myLabel.grid(row=4, column=1)

    def display_all(self, database):
        select_all = database
        return select_all

    def insert(self, database, val1, val2, val3):
        goods = val1.get()
        price = val2.get()
        date = val3.get()
        insertion = database(goods, price, date)
        return insertion

    def find_expense(self, database, val1, val2):
        goods = val1.get()
        price = val2.get()
        find = database(goods, price)
        return find

    def delete_expense(self,database, val1, val2):
        goods = val1.get()
        price = val2.get()
        delete = database(goods, price)
        return delete
```

```
    ###MAIN WINDOW###
    def main_window(self):
        button1 = Button(self.frame, text="Personal expenses",
command=self.personal)
        button1.pack()

        button2 = Button(self.frame, text="Household expenses",
command=self.household)
        button2.pack()

        button3 = Button(self.frame, text="Necessities expenses",
command=self.necessities)
        button3.pack()

        button4 = Button(self.frame, text="Entertainment
expenses",command=self.entertainment)
        button4.pack()

        button5 = Button(self.frame, text="Other expenses",
command=self.other)
        button5.pack()

        button6 = Button(self.frame, text="EXIT", command=exit)
        button6.pack()

    ###INSERT VALUES###
    def personal(self):
            top = Toplevel(self.frame)
            top.title('Personal expenses')
            l1 = Label(top, text="Name of good").grid(row = 1, column = 0,
sticky = W, pady = 2)
            l2 = Label(top, text="Price").grid(row = 2, column = 0, sticky =
W, pady = 2)
            l3 = Label(top, text="Date of purchase").grid(row = 3, column = 0,
sticky = W, pady = 2)

            e1 = Entry(top)
            e1.grid(row=1, column=1, sticky=W, pady=2)
            e2 = Entry(top)
            e2.grid(row=2, column=1, sticky=W, pady=2)
            e3 = Entry(top)
            e3.grid(row=3, column=1, sticky=W, pady=2)

            text = Text(top, width=40, height=10)
```

```
            text.grid(row=5, column=1, columnspan=2)

            def clear():
                e1.delete(0,END)
                e2.delete(0,END)
                e3.delete(0,END)



         #BUTTONS###

            B1 = Button(top, text="Insert Values", command=lambda:
(self.insert(db.insert_personal,e1,e2,e3), self.added(top)))
            B1.grid(row=4, column=0)

            B2 = Button(top, text="Select All", command=lambda:
(text.delete(1.0, END), text.insert(END,
self.display_all(db.select_all_personal())))
            B2.grid(row=1, column=2)



            B4 = Button(top, text="Delete expense", command=lambda:
(self.delete_expense(db.delete_personal, e1,e2), self.delete(top)))
            B4.grid(row=2, column=2)

            B5= Button(top, text="Exit", command=exit)
            B5.grid(row=3, column=2)

            B6=Button(top,text='Clear',command=clear)
            B6.grid(row=4, column=2)




    def household(self):
        top = Toplevel(self.frame)
        top.title('Household expenses')
        l1 = Label(top, text="Name of good").grid(row=1, column=0, sticky=W,
pady=2)
        l2 = Label(top, text="Price").grid(row=2, column=0, sticky=W, pady=2)
        l3 = Label(top, text="Date of purchase").grid(row=3, column=0,
sticky=W, pady=2)

        e1 = Entry(top)
        e1.grid(row=1, column=1, sticky=W, pady=2)
        e2 = Entry(top)
```

```
        e2.grid(row=2, column=1, sticky=W, pady=2)
        e3 = Entry(top)
        e3.grid(row=3, column=1, sticky=W, pady=2)

        text = Text(top, width=40, height=10)
        text.grid(row=5, column=1, columnspan=2)

        def clear():
                e1.delete(0,END)
                e2.delete(0,END)
                e3.delete(0,END)

        # BUTTONS###

        B1 = Button(top, text="Insert Values",
                    command=lambda: (self.insert(db.insert_household, e1, e2,
e3), self.added(top)))
        B1.grid(row=4, column=0)

        B2 = Button(top, text="Select All", command=lambda: (text.delete(1.0,
END), text.insert(END, self.display_all(db.select_all_household()))))
        B2.grid(row=1, column=2)



        B3 = Button(top, text="Delete expense",
                    command=lambda: (self.delete_expense(db.delete_household,
e1, e2), self.delete(top)))
        B3.grid(row=2, column=2)

        B4 = Button(top, text="Exit", command=exit)
        B4.grid(row=3, column=2)

        B5=Button(top,text='Clear',command=clear)
        B5.grid(row=4, column=2)


    def necessities(self):
        top = Toplevel(self.frame)
        top.title('Necessities expenses')
        l1 = Label(top, text="Name of good").grid(row=1, column=0, sticky=W,
pady=2)
        l2 = Label(top, text="Price").grid(row=2, column=0, sticky=W, pady=2)
        l3 = Label(top, text="Date of purchase").grid(row=3, column=0,
sticky=W, pady=2)

        e1 = Entry(top)
        e1.grid(row=1, column=1, sticky=W, pady=2)
        e2 = Entry(top)
```

```
        e2.grid(row=2, column=1, sticky=W, pady=2)
        e3 = Entry(top)
        e3.grid(row=3, column=1, sticky=W, pady=2)

        text = Text(top, width=40, height=10)
        text.grid(row=5, column=1, columnspan=2)

        def clear():
                e1.delete(0,END)
                e2.delete(0,END)
                e3.delete(0,END)

        # BUTTONS###

        B1 = Button(top, text="Insert Values",
                    command=lambda: (self.insert(db.insert_necessities, e1,
e2, e3), self.added(top)))
        B1.grid(row=4, column=0)

        B2 = Button(top, text="Select All", command=lambda: (text.delete(1.0,
END), text.insert(END, self.display_all(db.select_all_necessities))))
        B2.grid(row=1, column=2)



        B3 = Button(top, text="Delete expense",
                    command=lambda:
(self.delete_expense(db.delete_necessities, e1, e2), self.delete(top)))
        B3.grid(row=2, column=2)

        B4 = Button(top, text="Exit", command=exit)
        B4.grid(row=3, column=2)

        B5=Button(top,text='Clear',command=clear)
        B5.grid(row=4, column=2)


    def entertainment(self):
        top = Toplevel(self.frame)
        top.title('Entertainment expenses')
        l1 = Label(top, text="Name of good").grid(row=1, column=0, sticky=W,
pady=2)
        l2 = Label(top, text="Price").grid(row=2, column=0, sticky=W, pady=2)
        l3 = Label(top, text="Date of purchase").grid(row=3, column=0,
sticky=W, pady=2)

        e1 = Entry(top)
        e1.grid(row=1, column=1, sticky=W, pady=2)
        e2 = Entry(top)
```

```
        e2.grid(row=2, column=1, sticky=W, pady=2)
        e3 = Entry(top)
        e3.grid(row=3, column=1, sticky=W, pady=2)

        text = Text(top, width=40, height=10)
        text.grid(row=5, column=1, columnspan=2)

        def clear():
                e1.delete(0,END)
                e2.delete(0,END)
                e3.delete(0,END)

        # BUTTONS###

        B1 = Button(top, text="Insert Values",
                    command=lambda: (self.insert(db.insert_entertainment, e1,
e2, e3), self.added(top)))
        B1.grid(row=4, column=0)

        B2 = Button(top, text="Select All", command=lambda: (
        text.delete(1.0, END), text.insert(END,
self.display_all(db.select_all_entertainment())))
        B2.grid(row=1, column=2)



        B3 = Button(top, text="Delete expense",
                    command=lambda:
(self.delete_expense(db.delete_entertainment, e1, e2), self.delete(top)))
        B3.grid(row=2, column=2)

        B4 = Button(top, text="Exit", command=exit)
        B4.grid(row=3, column=2)

        B5=Button(top,text='Clear',command=clear)
        B5.grid(row=4, column=2)

    def other(self):
        top = Toplevel(self.frame)
        top.title('Other expenses')
        l1 = Label(top, text="Name of good").grid(row=1, column=0, sticky=W,
pady=2)
        l2 = Label(top, text="Price").grid(row=2, column=0, sticky=W, pady=2)
        l3 = Label(top, text="Date of purchase").grid(row=3, column=0,
sticky=W, pady=2)

        e1 = Entry(top)
        e1.grid(row=1, column=1, sticky=W, pady=2)
```

```
        e2 = Entry(top)
        e2.grid(row=2, column=1, sticky=W, pady=2)
        e3 = Entry(top)
        e3.grid(row=3, column=1, sticky=W, pady=2)

        text = Text(top, width=40, height=10)
        text.grid(row=5, column=1, columnspan=2)

        def clear():
                e1.delete(0,END)
                e2.delete(0,END)
                e3.delete(0,END)

        # BUTTONS###

        B1 = Button(top, text="Insert Values",
                    command=lambda: (self.insert(db.insert_other, e1, e2, e3),
self.added(top)))
        B1.grid(row=4, column=0)

        B2 = Button(top, text="Select All", command=lambda: (
            text.delete(1.0, END), text.insert(END,
self.display_all(db.select_all_other())))
        B2.grid(row=1, column=2)



        B3 = Button(top, text="Delete expense",
                    command=lambda: (self.delete_expense(db.delete_other, e1,
e2), self.delete(top)))
        B3.grid(row=2, column=2)

        B4 = Button(top, text="Exit", command=exit)
        B4.grid(row=3, column=2)

        B5=Button(top,text='Clear',command=clear)
        B5.grid(row=4, column=2)


def main():
        root = Tk()
        root.geometry('250x200')
        root.title("Expense Tracker")
        ExpenseTracker(root)


        root.mainloop()
```

```
main()
```

```
import sqlite3
import datetime
now = datetime.datetime.utcnow()

CREATE_PERSONAL = "CREATE TABLE IF NOT EXISTS personal (id INTEGER PRIMARY
KEY,good TEXT, price INTEGER, date DATE);"
CREATE_HOUSEHOLD = "CREATE TABLE IF NOT EXISTS household (id INTEGER PRIMARY
KEY,good TEXT, price INTEGER, date DATE);"
CREATE_NECESSITIES = "CREATE TABLE IF NOT EXISTS necessities (id INTEGER
PRIMARY KEY,good TEXT, price INTEGER, date DATE);"
CREATE_ENTERTAINMENT = "CREATE TABLE IF NOT EXISTS entertainment (id INTEGER
PRIMARY KEY,good TEXT, price INTEGER, date DATE);"
CREATE_OTHER = "CREATE TABLE IF NOT EXISTS other (id INTEGER PRIMARY KEY,good
TEXT, price INTEGER, date DATE);"



INSERT_PERSONAL = "INSERT INTO personal (good, price, date) VALUES(?,?,?);"
INSERT_HOUSEHOLD = "INSERT INTO household (good, price, date) VALUES(?,?,?);"
INSERT_NECESSITIES = "INSERT INTO necessities (good, price, date)
VALUES(?,?,?);"
INSERT_ENTERTAINMENT = "INSERT INTO entertainment (good, price, date)
VALUES(?,?,?);"
INSERT_OTHER = "INSERT INTO other (good, price, date) VALUES(?,?,?);"


SELECT_ALL1 = "SELECT * FROM personal;"
SELECT_ALL2 = "SELECT * FROM household;"
SELECT_ALL3 = "SELECT * FROM necessities;"
SELECT_ALL4 = "SELECT * FROM entertainment;"
SELECT_ALL5 = "SELECT * FROM other;"

SELECT_PERSONAL = "SELECT * FROM personal WHERE good = ? AND price = ?;"
SELECT_HOUSEHOLD = "SELECT * FROM household WHERE good = ? AND price = ?;"
SELECT_NECESSITIES = "SELECT * FROM necessities; WHERE good = ? AND price =
?;"
SELECT_ENTERTAINMENT = "SELECT * FROM entertainment WHERE good = ? AND price =
?;"
SELECT_OTHER = "SELECT * FROM other WHERE good = ? AND price = ?;"

DELETE_PERSONAL = "DELETE FROM personal WHERE good = ? AND price = ?;"
DELETE_HOUSEHOLD = "DELETE FROM household WHERE good = ? AND price = ?;"
DELETE_NECESSITIES = "DELETE FROM necessities; WHERE good = ? AND price = ?;"
```

```
DELETE_ENTERTAINMENT = "DELETE FROM entertainment WHERE good = ? AND price =
?;"
DELETE_OTHER = "DELETE FROM other WHERE good = ? AND price = ?;"

columns=
[CREATE_PERSONAL,CREATE_HOUSEHOLD,CREATE_NECESSITIES,CREATE_ENTERTAINMENT,CREA
TE_OTHER]

def creating_tables():
    conn = sqlite3.connect('data.db')
    with conn:
        for db in columns:
            conn.execute(db)

creating_tables()

#INSERTING VALUES

def insert_personal(good, price, date):
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(INSERT_PERSONAL, (good, price, date))
        conn.commit()

def insert_household(good, price, date):
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(INSERT_HOUSEHOLD, (good, price, date))
        conn.commit()

def insert_necessities(good, price, date):
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(INSERT_NECESSITIES, (good, price, date))
        conn.commit()

def insert_entertainment(good, price, date):
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(INSERT_ENTERTAINMENT, (good, price, date))
        conn.commit()

def insert_other(good, price, date):
    conn = sqlite3.connect('data.db')
```

```
    with conn:
        c = conn.cursor()
        c.execute(INSERT_OTHER, (good, price, date))
        conn.commit()

#SELECTION

def select_all_personal():
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(SELECT_ALL1)
        #have to store data into a list of Tuple
        list = c.fetchall()
        c.close()
        output = ''
        for x in list:
            output = output + str(x[1]) + ' ' + str(x[2]) + ' ' + ' ' +
str(x[3]) + '\n'
        return output

def select_personal(good, price):
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(SELECT_PERSONAL, (good, price))
        # have to store data into a list of Tuple
        list = c.fetchone()
        c.close()
        output = ''
        for x in list:
            output = output + str(x[1]) + ' ' + str(x[2]) + ' ' + ' ' +
str(x[3]) + '\n'
        return output


def select_all_household():
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(SELECT_ALL2)
        #have to store data into a list of Tuple
        list = c.fetchall()
        c.close()
        output = ''
        for x in list:
            output = output + str(x[1]) + ' ' + str(x[2]) + ' ' + ' ' +
str(x[3]) + '\n'
        return output
```

```
def select_household(good, price):
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(SELECT_HOUSEHOLD, (good, price))
        # have to store data into a list of Tuple
        list = c.fetchone()
        c.close()
        output = ''
        for x in list:
            output = output + str(x[1]) + ' ' + str(x[2]) + ' ' + ' ' +
str(x[3]) + '\n'
        return output


def select_all_necessities():
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(SELECT_ALL3)
        #have to store data into a list of Tuple
        list = c.fetchall()
        c.close()
        output = ''
        for x in list:
            output = output + str(x[1]) + ' ' + str(x[2]) + ' ' + ' ' +
str(x[3]) + '\n'
        return output

def select_necessities(good, price):
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(SELECT_NECESSITIES, (good, price))
        # have to store data into a list of Tuple
        list = c.fetchone()
        c.close()
        output = ''
        for x in list:
            output = output + str(x[1]) + ' ' + str(x[2]) + ' ' + ' ' +
str(x[3]) + '\n'
        return output


def select_all_entertainment():
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
```

```
        c.execute(SELECT_ALL4)
        #have to store data into a list of Tuple
        list = c.fetchall()
        c.close()
        output = ''
        for x in list:
            output = output + str(x[1]) + ' ' + str(x[2]) + ' ' + ' ' +
str(x[3]) + '\n'
        return output

def select_entertainment(good, price):
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(SELECT_ENTERTAINMENT, (good, price))
        # have to store data into a list of Tuple
        list = c.fetchone()
        c.close()
        output = ''
        for x in list:
            output = output + str(x[1]) + ' ' + str(x[2]) + ' ' + ' ' +
str(x[3]) + '\n'
        return output


def select_all_others():
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(SELECT_ALL5)
        #have to store data into a list of Tuple
        list = c.fetchall()
        c.close()
        output = ''
        for x in list:
            output = output + str(x[1]) + ' ' + str(x[2]) + ' ' + ' ' +
str(x[3]) + '\n'
        return output

def select_others(good, price):
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(SELECT_OTHER, (good, price))
        # have to store data into a list of Tuple
        list = c.fetchone()
        c.close()
        output = ''
        for x in list:
```

```
            output = output + str(x[1]) + ' ' + str(x[2]) + ' ' + ' ' +
str(x[3]) + '\n'
        return output

# DELETING

def delete_personal(good, price):
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(DELETE_PERSONAL, (good, price))
        conn.commit()
        c.close()

def delete_household(good, price):
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(DELETE_HOUSEHOLD, (good, price))
        conn.commit()
        c.close()

def delete_necessities(good, price):
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(DELETE_NECESSITIES, (good, price))
        conn.commit()
        c.close()

def delete_entertainment(good, price):
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(DELETE_ENTERTAINMENT, (good, price))
        conn.commit()
        c.close()

def delete_others(good, price):
    conn = sqlite3.connect('data.db')
    with conn:
        c = conn.cursor()
        c.execute(DELETE_OTHER, (good, price))
        conn.commit()
        c.close()
```

References:

https://www.tutorialspoint.com/python/python_gui_programming.htm

https://medium.datadriveninvestor.com/

https://www.geeksforgeeks.org/python-sqlite/#:~:text=Python%20SQLite3%20module%20is%20used,after%20the%202.5x%20version.

Conclusion:

With the aid of this mini project, we were able to put our knowledge and concepts to use in creating a Python-based expense tracker programme. We used the tkinter, datetime, and db modules. We were able to create a graphical user interface (GUI) using Python's tkinter module.