In [2]:

```python
#Imports for Tkinter, MySQL connection and displaying current date
import pymysql
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
import tkinter as tk
from datetime import datetime
import csv
import os

#Function used to connect Python with MySQL database
def connection():
    conn = pymysql.connect(
        host='localhost',
        user='root',
        password='12345',
        db='inventory')
    return conn

def exportOut():
    fileName = str(fileNameEntry.get())
    conn = connection()
    cursor= conn.cursor()
    cursor.execute("SELECT * FROM Inventory")
    with open(fileName+".csv", "w") as csv_file:
        csv_writer = csv.writer(csv_file,delimiter="\t")
        csv_writer.writerow([i[0] for i in cursor.description])
        csv_writer.writerows(cursor)
    dir_path = os.getcwd() + "/"+fileName+".csv"
    messagebox.showinfo("success", "Exported Successfully")



#Read function used to read from the tables from the database
def read():
    #Uses connection() function to interact with database
    conn = connection()
    cursor = conn.cursor()
    #Executes query from python as if it was writing in MySQL.
    #Selects all item from the inve ntory table
    cursor.execute("SELECT * FROM Inventory")
    results = cursor.fetchall()
    conn.commit()
    conn.close()
    return results


#Function used to update inventory diaplay in GUI whenevr a change is made
def refreshTable():

    for data in my_tree.get_children():
        my_tree.delete(data)

    #Uses read() function to fetch the updated table and repalces it for the old one
    for array in read():
        my_tree.insert(parent='', index='end', iid=array, text="", values=(array), tag="orow")

    my_tree.tag_configure('orow', background='white', font=('Arial', 15))
    my_tree.place(x=50, y=110)


#Initiating tkinte
root = Tk()

#Creating main page
root.title("Inventory System")
root.geometry("1920x1080")

#Creating Tkinter tree
my_tree = ttk.Treeview(root, height=31, selectmode="browse")


#variables assigned to tkinter tree columns
ph1 = tk.StringVar()
ph2 = tk.StringVar()
ph3 = tk.StringVar()
ph4 = tk.StringVar()
fileName = tk.StringVar()


#Function used to fetch specific data from an item on the inventory
def setph(word,num):
    if num ==1:
        ph1.set(word)
    if num ==2:
        ph2.set(word)
    if num ==3:
        ph3.set(word)
    if num ==4:
        ph4.set(word)


#Function called whenver the add button is pressed. It adds item to the inventory table in MySQL
def add():
    #Assign variables for the information colected from the data entries from the GUI
    ID = str(IDEntry.get())
    Pname = str(PnameEntry.get())
    quantity = str(quantityEntry.get())
    expdate = str(expdateEntry.get())

    #Error message in case one of the fields is empty
    if (ID == "" or ID == " ") or (Pname == "" or Pname == " ") or (quantity == "" or quantity == " ") or (expdate == "" or expdate == " "
        messagebox.showinfo("Error", "Please fill up the blank entry")
        return
    else:
        try:
            conn = connection()
            cursor = conn.cursor()
            #Runs query which adds the variables containing the information from the data entries, into the Inventory table in the databas
            cursor.execute("INSERT INTO Inventory VALUES ('"+ID+"','"+Pname+"','"+quantity+"','"+expdate+"') ")
            conn.commit()
            conn.close()
        except:
            messagebox.showinfo("Error", "While (ADDING)")
            return
    #Change is made to the table so it has to be updated
    refreshTable()

#The reset() fucntion is used to delete all the data from the inventory table
def reset():
    #Question to ensure there isn't a misckli that might accidentally delete the inventory
    conditional = messagebox.askquestion("Warning!!", "Do you want to reset the inventory?")
    if conditional != "yes":
        return
    else:
        try:
            conn = connection()
            cursor = conn.cursor()
            #Query that resets the Iventory table from the database
            cursor.execute("DELETE FROM Inventory")
            conn.commit()
            conn.close()
        except:
            messagebox.showinfo("Error", "Sorry an error occured While (RESETING)")
            return
        #Change is made to
        refreshTable()

#Function that deletes an item from the inventory that is selected from the GUI
def delete():
    #Question to ensuere that the user does indeed want to delete that item
    decision = messagebox.askquestion("Warning!!", "Delete the selected data?")
    if decision != "yes":
        return
    else:
        #Selects in MySQL Inventory table, the item selected in the GUI
        selected_item = my_tree.selection()[0]
        deleteData = str(my_tree.item(selected_item)['values'][0])
        try:
            conn = connection()
            cursor = conn.cursor()
            #Runs query that deletes the selected item from the inventory table
            cursor.execute("DELETE FROM Inventory WHERE ID='"+str(deleteData)+"'")
            conn.commit()
            conn.close()
        except:
            messagebox.showinfo("Error", "Sorry an error occured While (DELETING)")
            return
        #change is made to a table so it has to be updated
        refreshTable()

#Select() function is used to select an item form the inventory
def select():
    try:
        #Values from item from inventory are stored as an array (with indexes)
        selected_item = my_tree.selection()[0]
        ID = str(my_tree.item(selected_item)['values'][0])
        Pname = str(my_tree.item(selected_item)['values'][1])
        quantity = str(my_tree.item(selected_item)['values'][2])
        expdate = str(my_tree.item(selected_item)['values'][3])

        #Function assigns these variables to ph values from setph() function
        setph(ID,1)
        setph(Pname,2)
        setph(quantity,3)
        setph(expdate,4)
    except:
        messagebox.showinfo("Error", "Please select a data row While (SELECTING)")

#Search function find the information of an item from the inventory
def search():
    #data from data entries is assgined to a variaable
    ID = str(IDEntry.get())
    Pname = str(PnameEntry.get())
    quantity = str(quantityEntry.get())
    ID_SUG = str(ID_SUGEntry.get())
    expdate = str(expdateEntry.get())

    conn = connection()
    cursor = conn.cursor()
    #Selects the data from the an item from the inven tory that has a compatible ID number
    cursor.execute("SELECT * FROM Inventory WHERE ID='"+ID+"' or PNAME='"+Pname+"' or QUANTITY='"+quantity+"' or EXPDATE='"+expdate+"' ")

    try:
        result = cursor.fetchall()
        #Re-writes the data from the found item in the data entries
        for num in range(0,5):
            setph(result[0][num],(num+1))

        conn.commit()
        conn.close()
    except:
        messagebox.showinfo("Error", "No data found While (SEARCHING)")



#This is the initiation for the pop up window as a Class
class popup:

    #Add pop_up window for customers
    def open_popup():

        #Creating the popup window
        top = Toplevel(root)
        top.geometry("1000x1080")

        #Creating tkinter tree for popup window
        top.title("Customer System")
        my_tree2 = ttk.Treeview(top, height=15, selectmode="browse")

        #Refresh functrion but for pop up window customers table
        def refreshTablePU():
            for data in my_tree2.get_children():
                my_tree2.delete(data)

            for array in readPU():
                my_tree2.insert(parent='', index='end', iid=array, text="", values=(array), tag="orow")

            my_tree2.tag_configure('orow', background='White', font=('Arial', 15))
            my_tree2.place(x=20, y=20)


        #Variables values from Tkinter tree columns
        var1 = tk.StringVar()
        var2 = tk.StringVar()
        var3 = tk.StringVar()
        var4 = tk.StringVar()

        #Read function for pop up window
        def readPU():
            conn = connection()
            cursor = conn.cursor()
            #Query fetches all the data from Customers table
            cursor.execute("SELECT * FROM CUSTOMER")
            results = cursor.fetchall()
            conn.commit()
            conn.close()
            return results

        #Adding item function to customers table
        def addPU():
            #Assigning the information from the data entries to variables
            Customer_ID = str(Customer_IDEntry.get())
            Customer_name = str(Customer_nameEntry.get())
            Customer_OwedToUs = str(Customer_OwedToUsEntry.get())
            Customer_OwedToThem = str(Customer_OwedToThemEntry.get())

            #If any data entry is empty, give an error
            if (Customer_ID == "" or Customer_ID == " ") or (Customer_name == "" or Customer_name == " ") or (Customer_OwedToUs == "" or C
                messagebox.showinfo("Error", "Please fill up the blank entry")
                return
            else:
                try:
                    conn = connection()
                    cursor = conn.cursor()
                    #Query that inserts previosuly created variables, into the customers table
                    cursor.execute("INSERT INTO CUSTOMER VALUES ('"+Customer_ID+"','"+Customer_name+"','"+Customer_OwedToUs+"','"+Customer
                    conn.commit()
                    conn.close()
                except:
                    messagebox.showinfo("Error", "While (ADDING) Customer")
                    return
            #Change is amde to CUSTOMER table so it has to be updated
            refreshTablePU()


        #Function used to fetch specific data from an item on the customers table
        def setphPU(word,num):
            if num ==1:
                var1.set(word)
            if num ==2:
                var2.set(word)
            if num ==3:
                var3.set(word)
            if num ==4:
                var4.set(word)


        #Function that deletes selecteed item from CUSTOMER table
        def deletePU():
            decision = messagebox.askquestion("Warning!!", "Delete the selected data?")
            if decision != "yes":
                return
            else:
                #Gathering the information of the selected item
                selected_item = my_tree2.selection()[0]
                deleteData = str(my_tree2.item(selected_item)['values'][0])
            try:
                conn = connection()
                cursor = conn.cursor()
                #Deleting the selected item using the deleteData variable which is created above,
```

```python
                #to know which item is the one that needs to be deleted in the database
                cursor.execute("DELETE FROM CUSTOMER WHERE Customer_ID='"+str(deleteData)+"'")
                conn.commit()
                conn.close()
            except:
                messagebox.showinfo("Error", "Sorry an error occured While (DELETING)")
                return

            refreshTablePU()


        def resetPU():
            decision = messagebox.askquestion("Warning!!", "Delete all data?")
            if decision != "yes":
                return
            else:
                try:
                    conn = connection()
                    cursor = conn.cursor()
                    cursor.execute("DELETE FROM CUSTOMER")
                    conn.commit()
                    conn.close()
                except:
                    messagebox.showinfo("Error", "Sorry an error occured While (RESETING)")
                    return

                refreshTablePU()

        def selectPU():
            try:
                selected_item = my_tree2.selection()[0]
                Customer_ID = str(my_tree2.item(selected_item)['values'][0])
                Customer_name = str(my_tree2.item(selected_item)['values'][1])
                Customer_OwedToUs = str(my_tree2.item(selected_item)['values'][2])
                Customer_OwedToThem = str(my_tree2.item(selected_item)['values'][3])

                setphPU(Customer_ID,1)
                setphPU(Customer_name,2)
                setphPU(Customer_OwedToUs,3)
                setphPU(Customer_OwedToThem,4)
            except:
                messagebox.showinfo("Error", "Please select a data row While (SELECTING)")

```

```

    def searchPU():
        Customer_ID = str(Customer_IDEntry.get())
        Customer_name = str(Customer_nameEntry.get())
        Customer_OwedToUs = str(Customer_OwedToUsEntry.get())
        Customer_OwedToThem = str(Customer_OwedToThemEntry.get())

        conn = connection()
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM CUSTOMER WHERE Customer_ID='"+Customer_ID+"' or Customer_name='"+Customer_name+"' or Customer_Ow

        try:
            result = cursor.fetchall()
            #range needs to be set to the number of data inputs each item customer has. EG Name and ID are two of these data inputs.
            for num in range(0,4):
                setphUP(result[0][num],(num+1))

            conn.commit()
            conn.close()
        except:
            messagebox.showinfo("Error", "No data found While (SEARCHING)")



#POP UP WINDOW GUI:_____________________________________________________________________________________________________________

    #Tkinter tree columns creation
    my_tree2['columns'] = ("Customer_ID","Customer_name","Customer_OwedToUs","Customner_OwedToThem")
    my_tree2.column("#0", width=0, stretch=YES)
    my_tree2.column("Customer_ID", anchor="center", width=30, stretch=YES)
    my_tree2.column("Customer_name", anchor="center", width=300, stretch=YES)
    my_tree2.column("Customer_OwedToUs", anchor="center", width=150, stretch=YES)
    my_tree2.column("Customner_OwedToThem", anchor="center", width=250, stretch=YES)

    #Heading for the columns
    my_tree2.heading("Customer_ID", text="ID", anchor="center")
    my_tree2.heading("Customer_name", text="Customer", anchor="center")
    my_tree2.heading("Customer_OwedToUs", text="Owed to Us", anchor="center")
    my_tree2.heading("Customner_OwedToThem", text="Owed to Customer", anchor="center")

    refreshTablePU()

    #Creating labels for data entries
```

```python
        Customer_IDLabel = Label(top, text="|             ID             |", font=('Arial bold', 15), bg="Dark Green", fg="white")
        Customer_nameLabel = Label(top, text="|       Customer        |", font=('Arial bold', 15), bg="Dark Green", fg="white")
        Customer_OwedToUsLabel = Label(top, text="|      Owed to Us      |", font=('Arial bold', 15), bg="Dark Green", fg="white")
        Customner_OwedToThemLabel = Label(top, text="| Owed to Customer |", font=('Arial bold', 15), bg="Dark Green", fg="white")

        #Creating data entries
        #Assign values from each entry to a text variable. EG for ID Entry this value is assgined to var1 from the setPU() function
        Customer_IDEntry = Entry(top, width=45, bd=5, font=('Arial', 15), textvariable = var1)
        Customer_nameEntry = Entry(top, width=45, bd=5, font=('Arial', 15), textvariable = var2)
        Customer_OwedToUsEntry = Entry(top, width=45, bd=5, font=('Arial', 15), textvariable = var3)
        Customer_OwedToThemEntry = Entry(top, width=45, bd=5, font=('Arial', 15), textvariable = var4)

        #Positioning data entries labels for pop up window
        Customer_IDLabel.place(x=10, y=400)
        Customer_nameLabel.place(x=10, y=460)
        Customer_OwedToUsLabel.place(x=10, y=520)
        Customner_OwedToThemLabel.place(x=10, y=580)

        #Positioning Data entries in pop up window
        Customer_IDEntry.place(x=210, y=400)
        Customer_nameEntry.place(x=210, y=460)
        Customer_OwedToUsEntry.place(x=210, y=520)
        Customer_OwedToThemEntry.place(x=210, y=580)

        #Creating buttons for pop up window
        #have to use "top" instead of "root", beacuse top makes reference to widgets on the pop up window
        #Each button has a Command which calls one of the functions.
        deleteCustomersBtn = Button(
            top, text="-", padx=5, pady=5, width=5, font=('Helvetica Bold', 15), bg="Red", fg="white", command=deletePU, bd=3)
        addCustomersBtn = Button(
            top, text="+", padx=5, pady=5, width=5, font=('Arial', 15), bg="Dark Green", command=addPU, fg="white",bd=3)
        searchCustomersBtn = Button(
            top, text="Search", padx=15, pady=15, width=10, font=('Arial', 15), bg="Dark Green", command=searchPU, fg="white", bd=3)
        resetCustomersBtn = Button(
            top, text="Reset", padx=15, pady=15, width=10, font=('Arial', 15), bg="Dark Green", command=resetPU, fg="white", bd=3)
        selectCustomersBtn = Button(
            top, text="Select", padx=15, pady=15, width=10, font=('Arial', 15), bg="Dark Green", command=selectPU, fg="white", bd=3)

        #Positioning buttons in pop up window
        deleteCustomersBtn.place(x=20, y=660)
        addCustomersBtn.place(x=110, y=660)

        selectCustomersBtn.place(x=210, y=650)
        searchCustomersBtn.place(x=390, y=650)
        resetCustomersBtn.place(x=570, y=650)







#Main page GUI:_______________________________________________________________________________________________

#Creating date variable that gathers todays date
now = datetime.now()

#Formating the date variable
date_time_str = now.strftime("%Y-%m-%d")
date = "[ Date: "+date_time_str+" ]"

#Creating label that displays the company name on the main page
label = Label(root, text="            LCacao Inventario            ", font=('Arial Bold', 40), bg= "#ace5ee", bd=10, fg="#65350F")
#Placing label
label.place(x=45, y=10)
#Creating label that dipalys todays date
label1 = Label(root, text= date, font=('Arial Bold', 38), bd=5, fg="black")
label1.place(x=940, y=10)


#LABELS _________________________________________________________________________________________

#Creating data entry labels for main page
IDLabel = Label(root, text="|                                           ID                                           |",
                font=('Arial bold', 17), bg="#001C57", fg="white", bd = 1)
PnameLabel = Label(root, text="|                                 Product Name                                 |",
                   font=('Arial bold', 17), bg="#001C57", fg="white")
quantityLabel = Label(root, text="|                                      Quantity                                      |",
                      font=('Arial bold', 17), bg="#001C57", fg="white")
expdateLabel = Label(root, text="|                                Expiration Date                                |",
                     font=('Arial bold', 17), bg="#001C57", fg="white")
fileNameLabel = Label(root, text="|                   Export File Name                    |",
                     font=('Arial bold', 17), bg="#001C57", fg="white")

#Creating data entries for main page
IDEntry = Entry(root, width=50, bd=3, font=('Arial', 15), textvariable = ph1)
PnameEntry = Entry(root, width=50, bd=3, font=('Arial', 15), textvariable = ph2)
```

```python
quantityEntry = Entry(root, width=50, bd=3, font=('Arial', 15), textvariable = ph3)
expdateEntry = Entry(root, width=50, bd=3, font=('Arial', 15), textvariable = ph4)
fileNameEntry = Entry(root, width=38, bd=3, font=('Arial', 15), textvariable = fileName)

#Positioning data entry labels on main page
fileNameLabel.place(x=900, y=115)
IDLabel.place(x=900, y=204)
PnameLabel.place(x=900, y=294)
quantityLabel.place(x=900, y=384)
#ID_SUGLabel.place(x=930, y=384)
expdateLabel.place(x=900, y=474)


#Positioning the data entries on the main page
fileNameEntry.place(x=900, y=145)
IDEntry.place(x=900, y=235)
PnameEntry.place(x=900, y=325)
quantityEntry.place(x=900, y=415)
#ID_SUGEntry.place(x=900, y =415)
expdateEntry.place(x=900, y=505)



#Buttons________________________________________________________________________________________________________________________

#Creating buttons for main page
#This buttons use the "root" instead of "top", because "root" makes reference to the widgets on the main page
deleteBtn = Button(
    root, text="-", padx=5, pady=5, width=5, font=('Helvetica Bold', 18), bg="Red", fg="white", command=delete, bd=3)
addBtn = Button(
    root, text="+", padx=5, pady=5, width=5, font=('Arial', 18), bg="Dark Green", command=add, fg="white",bd=3)
searchBtn = Button(
    root, text="Search", padx=15, pady=15, width=10, font=('Arial', 18), bg="#001C57", command=search, fg="white", bd=3)
resetBtn = Button(
    root, text="Reset", padx=15, pady=15, width=10, font=('Arial', 18), bg="#001C57", command=reset, fg="white", bd=3)
selectBtn = Button(
    root, text="Select", padx=15, pady=15, width=10, font=('Arial', 18), bg="#001C57", command=select, fg="white", bd=3)
customerBtn = Button(
    root, text="Customers", padx=15, pady=15, width=10, font=('Arial', 18), bg="#001C57", command=popup.open_popup, fg="white", bd=3)
Exportbttn = Button(
            root, text="Export", padx=7, pady=5, width=5, font=('Helvetica Bold', 22), bg="#001C57", fg="white", command=exportOut, bd=3)

#Positioning buttons on main page
```

```python
Exportbttn.place(x=1343, y=111)

deleteBtn.place(x=940, y=570)
addBtn.place(x=940, y=690)

customerBtn.place(x=1070, y=560)
selectBtn.place(x=1275, y=560)

searchBtn.place(x=1070, y=680)
resetBtn.place(x=1275, y=680)


#Tkinter Tree (for inventory)____________________________________________________________________________________________
style = ttk.Style()
style.configure("Treeview.Heading", font=('Helvetica Bold', 12), relief="flat", fieldbackground="black")
style.theme_use("default")

#Creating columns
my_tree['columns'] = ("ID","Pname","quantity", "expdate")
my_tree.column("#0", width=0, stretch=YES)
my_tree.column("ID", anchor="center", width=40, stretch=YES)
my_tree.column("Pname", anchor="center", width=380,stretch=YES)
my_tree.column("quantity", anchor="center", width=130, stretch=YES)
my_tree.column("expdate", anchor="center", width=240,stretch=YES)

#Headings for the columns
my_tree.heading("ID", text="ID", anchor="center")
my_tree.heading("Pname", text="Product Name", anchor="center")
my_tree.heading("quantity", text="Quantity (KG)", anchor="center")
my_tree.heading("expdate", text="Expiration Date (YYYY-MM-DD)", anchor="center")

refreshTable()

#Continues to call the "root" which is the main page.
root.mainloop()
```

In [ ]:

```
1
```