

In [2]: ▶

```
1  #Imports for Tkinter, MySQL connection and displaying current date
2  import pymysql
3  from tkinter import *
4  from tkinter import ttk
5  from tkinter import messagebox
6  import tkinter as tk
7  from datetime import datetime
8  import csv
9  import os
10
11 #Function used to connect Python with MySQL database
12 def connection():
13     conn = pymysql.connect(
14         host='localhost',
15         user='root',
16         password='12345',
17         db='inventory')
18     return conn
19
20 def exportOut():
21     fileName = str(fileNameEntry.get())
22     conn = connection()
23     cursor= conn.cursor()
24     cursor.execute("SELECT * FROM Inventory")
25     with open(fileName+".csv", "w") as csv_file:
26         csv_writer = csv.writer(csv_file,delimiter="\t")
27         csv_writer.writerow([i[0] for i in cursor.description])
28         csv_writer.writerows(cursor)
29     dir_path = os.getcwd() + "/" +fileName+".csv"
30     messagebox.showinfo("success", "Exported Successfully")
31
32
33
34 #Read function used to read from the tables from the database
35 def read():
36     #Uses connection() function to interact with database
37     conn = connection()
38     cursor = conn.cursor()
39     #Executes query from python as if it was writing in MySQL.
40     #Selects all item from the inventory table
41     cursor.execute("SELECT * FROM Inventory")
42     results = cursor.fetchall()
43     conn.commit()
```

```

44     conn.close()
45     return results
46
47
48 #Function used to update inventory diaplay in GUI whenevr a change is made
49 def refreshTable():
50
51     for data in my_tree.get_children():
52         my_tree.delete(data)
53
54     #Uses read() function to fetch the updated table and repalces it for the old one
55     for array in read():
56         my_tree.insert(parent='', index='end', iid=array, text="", values=(array), tag="orow")
57
58     my_tree.tag_configure('orow', background='white', font=('Arial', 15))
59     my_tree.place(x=50, y=110)
60
61
62 #Initiating tkinte
63 root = Tk()
64
65 #Creating main page
66 root.title("Inventory System")
67 root.geometry("1920x1080")
68
69 #Creating Tkinter tree
70 my_tree = ttk.Treeview(root, height=31, selectmode="browse")
71
72
73 #variables assigned to tkinter tree columns
74 ph1 = tk.StringVar()
75 ph2 = tk.StringVar()
76 ph3 = tk.StringVar()
77 ph4 = tk.StringVar()
78 fileName = tk.StringVar()
79
80
81 #Function used to fetch specific data from an item on the inventory
82 def setph(word,num):
83     if num ==1:
84         ph1.set(word)
85     if num ==2:
86         ph2.set(word)

```

```

87     if num ==3:
88         ph3.set(word)
89     if num ==4:
90         ph4.set(word)
91
92
93 #Function called whenever the add button is pressed. It adds item to the inventory table in MySQL
94 def add():
95     #Assign variables for the information collected from the data entries from the GUI
96     ID = str(IDEntry.get())
97     Pname = str(PnameEntry.get())
98     quantity = str(quantityEntry.get())
99     expdate = str(expdateEntry.get())
100
101     #Error message in case one of the fields is empty
102     if (ID == "" or ID == " ") or (Pname == "" or Pname == " ") or (quantity == "" or quantity == " ") or (expdate == "" or expdate == " ")
103         messagebox.showinfo("Error", "Please fill up the blank entry")
104         return
105     else:
106         try:
107             conn = connection()
108             cursor = conn.cursor()
109             #Runs query which adds the variables containing the information from the data entries, into the Inventory table in the database
110             cursor.execute("INSERT INTO Inventory VALUES ('"+ID+"', '"+Pname+"', '"+quantity+"', '"+expdate+"') ")
111             conn.commit()
112             conn.close()
113         except:
114             messagebox.showinfo("Error", "While (ADDING)")
115             return
116     #Change is made to the table so it has to be updated
117     refreshTable()
118
119 #The reset() function is used to delete all the data from the inventory table
120 def reset():
121     #Question to ensure there isn't a misckli that might accidentally delete the inventory
122     conditional = messagebox.askquestion("Warning!!", "Do you want to reset the inventory?")
123     if conditional != "yes":
124         return
125     else:
126         try:
127             conn = connection()
128             cursor = conn.cursor()
129             #Query that resets the Inventory table from the database

```

```

130         cursor.execute("DELETE FROM Inventory")
131         conn.commit()
132         conn.close()
133     except:
134         messagebox.showinfo("Error", "Sorry an error occured While (RESETTING)")
135         return
136     #Change is made to
137     refreshTable()
138
139 #Function that deletes an item from the inventory that is selected from the GUI
140 def delete():
141     #Question to ensuere that the user does indeed want to delete that item
142     decision = messagebox.askquestion("Warning!!", "Delete the selected data?")
143     if decision != "yes":
144         return
145     else:
146         #Selects in MySQL Inventory table, the item selected in the GUI
147         selected_item = my_tree.selection()[0]
148         deleteData = str(my_tree.item(selected_item)['values'][0])
149         try:
150             conn = connection()
151             cursor = conn.cursor()
152             #Runs query that deletes the selected item from the inventory table
153             cursor.execute("DELETE FROM Inventory WHERE ID='"+str(deleteData)+"'")
154             conn.commit()
155             conn.close()
156         except:
157             messagebox.showinfo("Error", "Sorry an error occured While (DELETING)")
158             return
159         #change is made to a table so it has to be updated
160         refreshTable()
161
162 #Select() function is used to select an item form the inventory
163 def select():
164     try:
165         #Values from item from inventory are stored as an array (with indexes)
166         selected_item = my_tree.selection()[0]
167         ID = str(my_tree.item(selected_item)['values'][0])
168         Pname = str(my_tree.item(selected_item)['values'][1])
169         quantity = str(my_tree.item(selected_item)['values'][2])
170         expdate = str(my_tree.item(selected_item)['values'][3])
171
172         #Function assigns these variables to ph values from setph() function

```

```

173         setph(ID,1)
174         setph(Pname,2)
175         setph(quantity,3)
176         setph(expdate,4)
177     except:
178         messagebox.showinfo("Error", "Please select a data row While (SELECTING)")
179
180 #Search function find the information of an item from the inventory
181 def search():
182     #data from data entries is assigned to a variable
183     ID = str(IDEntry.get())
184     Pname = str(PnameEntry.get())
185     quantity = str(quantityEntry.get())
186     ID_SUG = str(ID_SUGEntry.get())
187     expdate = str(expdateEntry.get())
188
189     conn = connection()
190     cursor = conn.cursor()
191     #Selects the data from the an item from the inven tory that has a compatible ID number
192     cursor.execute("SELECT * FROM Inventory WHERE ID='"+ID+"' or PNAME='"+Pname+"' or QUANTITY='"+quantity+"' or EXPDATE='"+expdate+"' ")
193
194     try:
195         result = cursor.fetchall()
196         #Re-writes the data from the found item in the data entries
197         for num in range(0,5):
198             setph(result[0][num],(num+1))
199
200         conn.commit()
201         conn.close()
202     except:
203         messagebox.showinfo("Error", "No data found While (SEARCHING)")
204
205
206
207 #This is the initiation for the pop up window as a Class
208 class popup:
209
210     #Add pop_up window for customers
211     def open_popup():
212
213         #Creating the popup window
214         top = Toplevel(root)
215         top.geometry("1000x1080")

```

```

216
217 #Creating tkinter tree for popup window
218 top.title("Customer System")
219 my_tree2 = ttk.Treeview(top, height=15, selectmode="browse")
220
221 #Refresh functrion but for pop up window customers table
222 def refreshTablePU():
223     for data in my_tree2.get_children():
224         my_tree2.delete(data)
225
226     for array in readPU():
227         my_tree2.insert(parent='', index='end', iid=array, text="", values=(array), tag="orow")
228
229     my_tree2.tag_configure('orow', background='White', font=('Arial', 15))
230     my_tree2.place(x=20, y=20)
231
232
233 #Variables values from Tkinter tree columns
234 var1 = tk.StringVar()
235 var2 = tk.StringVar()
236 var3 = tk.StringVar()
237 var4 = tk.StringVar()
238
239 #Read function for pop up window
240 def readPU():
241     conn = connection()
242     cursor = conn.cursor()
243     #Query fetches all the data from Customers table
244     cursor.execute("SELECT * FROM CUSTOMER")
245     results = cursor.fetchall()
246     conn.commit()
247     conn.close()
248     return results
249
250 #Adding item function to customers table
251 def addPU():
252     #Assigning the information from the data entries to variables
253     Customer_ID = str(Customer_IDEntry.get())
254     Customer_name = str(Customer_nameEntry.get())
255     Customer_OwedToUs = str(Customer_OwedToUsEntry.get())
256     Customer_OwedToThem = str(Customer_OwedToThemEntry.get())
257
258     #If any data entry is empty, give an error

```

```

259     if (Customer_ID == "" or Customer_ID == " ") or (Customer_name == "" or Customer_name == " ") or (Customer_OwedToUs == "" or C
260         messagebox.showinfo("Error", "Please fill up the blank entry")
261         return
262     else:
263         try:
264             conn = connection()
265             cursor = conn.cursor()
266             #Query that inserts previosuly created variables, into the customers table
267             cursor.execute("INSERT INTO CUSTOMER VALUES ('"+Customer_ID+"', '"+Customer_name+"', '"+Customer_OwedToUs+"', '"+Customer
268             conn.commit()
269             conn.close()
270         except:
271             messagebox.showinfo("Error", "While (ADDING) Customer")
272             return
273         #Change is amde to CUSTOMER table so it has to be updated
274         refreshTablePU()
275
276
277     #Function used to fetch specific data from an item on the customers table
278     def setphPU(word,num):
279         if num ==1:
280             var1.set(word)
281         if num ==2:
282             var2.set(word)
283         if num ==3:
284             var3.set(word)
285         if num ==4:
286             var4.set(word)
287
288
289     #Function that deletes selecteed item from CUSTOMER table
290     def deletePU():
291         decision = messagebox.askquestion("Warning!!", "Delete the selected data?")
292         if decision != "yes":
293             return
294         else:
295             #Gathering the information of the selected item
296             selected_item = my_tree2.selection()[0]
297             deleteData = str(my_tree2.item(selected_item)['values'][0])
298         try:
299             conn = connection()
300             cursor = conn.cursor()
301             #DeLeting the selected item using the deleteData variable which is created above,

```

```

302         #to know which item is the one that needs to be deleted in the database
303         cursor.execute("DELETE FROM CUSTOMER WHERE Customer_ID='"+str(deleteData)+"'")
304         conn.commit()
305         conn.close()
306     except:
307         messagebox.showinfo("Error", "Sorry an error occured While (DELETING)")
308     return
309
310     refreshTablePU()
311
312
313 def resetPU():
314     decision = messagebox.askquestion("Warning!!", "Delete all data?")
315     if decision != "yes":
316         return
317     else:
318         try:
319             conn = connection()
320             cursor = conn.cursor()
321             cursor.execute("DELETE FROM CUSTOMER")
322             conn.commit()
323             conn.close()
324         except:
325             messagebox.showinfo("Error", "Sorry an error occured While (RESETTING)")
326         return
327
328     refreshTablePU()
329
330 def selectPU():
331     try:
332         selected_item = my_tree2.selection()[0]
333         Customer_ID = str(my_tree2.item(selected_item)['values'][0])
334         Customer_name = str(my_tree2.item(selected_item)['values'][1])
335         Customer_OwedToUs = str(my_tree2.item(selected_item)['values'][2])
336         Customer_OwedToThem = str(my_tree2.item(selected_item)['values'][3])
337
338         setphPU(Customer_ID,1)
339         setphPU(Customer_name,2)
340         setphPU(Customer_OwedToUs,3)
341         setphPU(Customer_OwedToThem,4)
342     except:
343         messagebox.showinfo("Error", "Please select a data row While (SELECTING)")
344

```



```

345
346 def searchPU():
347     Customer_ID = str(Customer_IDEntry.get())
348     Customer_name = str(Customer_nameEntry.get())
349     Customer_OwedToUs = str(Customer_OwedToUsEntry.get())
350     Customer_OwedToThem = str(Customer_OwedToThemEntry.get())
351
352     conn = connection()
353     cursor = conn.cursor()
354     cursor.execute("SELECT * FROM CUSTOMER WHERE Customer_ID='"+Customer_ID+"' or Customer_name='"+Customer_name+"' or Customer_Ow
355
356     try:
357         result = cursor.fetchall()
358         #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.
359         for num in range(0,4):
360             setphUP(result[0][num],(num+1))
361
362         conn.commit()
363         conn.close()
364     except:
365         messagebox.showinfo("Error", "No data found While (SEARCHING)")
366
367
368
369 #POP UP WINDOW GUI:_____
370
371 #Tkinter tree columns creation
372 my_tree2['columns'] = ("Customer_ID", "Customer_name", "Customer_OwedToUs", "Customner_OwedToThem")
373 my_tree2.column("#0", width=0, stretch=YES)
374 my_tree2.column("Customer_ID", anchor="center", width=30, stretch=YES)
375 my_tree2.column("Customer_name", anchor="center", width=300, stretch=YES)
376 my_tree2.column("Customer_OwedToUs", anchor="center", width=150, stretch=YES)
377 my_tree2.column("Customner_OwedToThem", anchor="center", width=250, stretch=YES)
378
379 #Heading for the columns
380 my_tree2.heading("Customer_ID", text="ID", anchor="center")
381 my_tree2.heading("Customer_name", text="Customer", anchor="center")
382 my_tree2.heading("Customer_OwedToUs", text="Owed to Us", anchor="center")
383 my_tree2.heading("Customner_OwedToThem", text="Owed to Customer", anchor="center")
384
385 refreshTablePU()
386
387 #Creating Labels for data entries

```

```

388 Customer_IDLabel = Label(top, text="| ID |", font=('Arial bold', 15), bg="Dark Green", fg="white")
389 Customer_nameLabel = Label(top, text="| Customer |", font=('Arial bold', 15), bg="Dark Green", fg="white")
390 Customer_OwedToUsLabel = Label(top, text="| Owed to Us |", font=('Arial bold', 15), bg="Dark Green", fg="white")
391 Customner_OwedToThemLabel = Label(top, text="| Owed to Customer |", font=('Arial bold', 15), bg="Dark Green", fg="white")
392
393 #Creating data entries
394 #Assign values from each entry to a text variable. EG for ID Entry this value is assgined to var1 from the setPU() function
395 Customer_IDEntry = Entry(top, width=45, bd=5, font=('Arial', 15), textvariable = var1)
396 Customer_nameEntry = Entry(top, width=45, bd=5, font=('Arial', 15), textvariable = var2)
397 Customer_OwedToUsEntry = Entry(top, width=45, bd=5, font=('Arial', 15), textvariable = var3)
398 Customer_OwedToThemEntry = Entry(top, width=45, bd=5, font=('Arial', 15), textvariable = var4)
399
400 #Positioning data entries labels for pop up window
401 Customer_IDLabel.place(x=10, y=400)
402 Customer_nameLabel.place(x=10, y=460)
403 Customer_OwedToUsLabel.place(x=10, y=520)
404 Customner_OwedToThemLabel.place(x=10, y=580)
405
406 #Positioning Data entries in pop up window
407 Customer_IDEntry.place(x=210, y=400)
408 Customer_nameEntry.place(x=210, y=460)
409 Customer_OwedToUsEntry.place(x=210, y=520)
410 Customer_OwedToThemEntry.place(x=210, y=580)
411
412 #Creating buttons for pop up window
413 #have to use "top" instead of "root", beacuse top makes reference to widgets on the pop up window
414 #Each button has a Command which calls one of the functions.
415 deleteCustomersBtn = Button(
416     top, text="-", padx=5, pady=5, width=5, font=('Helvetica Bold', 15), bg="Red", fg="white", command=deletePU, bd=3)
417 addCustomersBtn = Button(
418     top, text="+", padx=5, pady=5, width=5, font=('Arial', 15), bg="Dark Green", command=addPU, fg="white", bd=3)
419 searchCustomersBtn = Button(
420     top, text="Search", padx=15, pady=15, width=10, font=('Arial', 15), bg="Dark Green", command=searchPU, fg="white", bd=3)
421 resetCustomersBtn = Button(
422     top, text="Reset", padx=15, pady=15, width=10, font=('Arial', 15), bg="Dark Green", command=resetPU, fg="white", bd=3)
423 selectCustomersBtn = Button(
424     top, text="Select", padx=15, pady=15, width=10, font=('Arial', 15), bg="Dark Green", command=selectPU, fg="white", bd=3)
425
426 #Positioning buttons in pop up window
427 deleteCustomersBtn.place(x=20, y=660)
428 addCustomersBtn.place(x=110, y=660)
429
430 selectCustomersBtn.place(x=210, y=650)

```

```

431         searchCustomersBtn.place(x=390, y=650)
432         resetCustomersBtn.place(x=570, y=650)
433
434
435
436
437
438
439 #Main page GUI:_____
440
441 #Creating date variable that gathers todays date
442 now = datetime.now()
443
444 #Formating the date variable
445 date_time_str = now.strftime("%Y-%m-%d")
446 date = "[ Date: "+date_time_str+" ]"
447
448 #Creating label that displays the company name on the main page
449 label = Label(root, text="                LCacao Inventario                ", font=('Arial Bold', 40), bg= "#ace5ee", bd=10, fg="#65350F")
450 #Placing label
451 label.place(x=45, y=10)
452 #Creating label that dipalys todays date
453 label1 = Label(root, text= date, font=('Arial Bold', 38), bd=5, fg="black")
454 label1.place(x=940, y=10)
455
456
457 #LABELS _____
458
459 #Creating data entry labels for main page
460 IDLabel = Label(root, text="|                                ID                                |",
461                 font=('Arial bold', 17), bg="#001C57", fg="white", bd = 1)
462 PnameLabel = Label(root, text="|                                Product Name                                |",
463                      font=('Arial bold', 17), bg="#001C57", fg="white")
464 quantityLabel = Label(root, text="|                                Quantity                                |",
465                        font=('Arial bold', 17), bg="#001C57", fg="white")
466 expdateLabel = Label(root, text="|                                Expiration Date                                |",
467                       font=('Arial bold', 17), bg="#001C57", fg="white")
468 fileNameLabel = Label(root, text="|                                Export File Name                                |",
469                          font=('Arial bold', 17), bg="#001C57", fg="white")
470
471 #Creating data entries for main page
472 IDEntry = Entry(root, width=50, bd=3, font=('Arial', 15), textvariable = ph1)
473 PnameEntry = Entry(root, width=50, bd=3, font=('Arial', 15), textvariable = ph2)

```

```

474 quantityEntry = Entry(root, width=50, bd=3, font=('Arial', 15), textvariable = ph3)
475 expdateEntry = Entry(root, width=50, bd=3, font=('Arial', 15), textvariable = ph4)
476 fileNameEntry = Entry(root, width=38, bd=3, font=('Arial', 15), textvariable = fileName)
477
478 #Positioning data entry Labels on main page
479 fileNameLabel.place(x=900, y=115)
480 IDLabel.place(x=900, y=204)
481 PnameLabel.place(x=900, y=294)
482 quantityLabel.place(x=900, y=384)
483 #ID_SUGLabel.place(x=930, y=384)
484 expdateLabel.place(x=900, y=474)
485
486
487 #Positioning the data entries on the main page
488 fileNameEntry.place(x=900, y=145)
489 IDEntry.place(x=900, y=235)
490 PnameEntry.place(x=900, y=325)
491 quantityEntry.place(x=900, y=415)
492 #ID_SUGEntry.place(x=900, y =415)
493 expdateEntry.place(x=900, y=505)
494
495
496
497 #Buttons
498
499 #Creating buttons for main page
500 #This buttons use the "root" instead of "top", because "root" makes reference to the widgets on the main page
501 deleteBtn = Button(
502     root, text="-", padx=5, pady=5, width=5, font=('Helvetica Bold', 18), bg="Red", fg="white", command=delete, bd=3)
503 addBtn = Button(
504     root, text="+", padx=5, pady=5, width=5, font=('Arial', 18), bg="Dark Green", command=add, fg="white", bd=3)
505 searchBtn = Button(
506     root, text="Search", padx=15, pady=15, width=10, font=('Arial', 18), bg="#001C57", command=search, fg="white", bd=3)
507 resetBtn = Button(
508     root, text="Reset", padx=15, pady=15, width=10, font=('Arial', 18), bg="#001C57", command=reset, fg="white", bd=3)
509 selectBtn = Button(
510     root, text="Select", padx=15, pady=15, width=10, font=('Arial', 18), bg="#001C57", command=select, fg="white", bd=3)
511 customerBtn = Button(
512     root, text="Customers", padx=15, pady=15, width=10, font=('Arial', 18), bg="#001C57", command=popup.open_popup, fg="white", bd=3)
513 Exportbtn = Button(
514     root, text="Export", padx=7, pady=5, width=5, font=('Helvetica Bold', 22), bg="#001C57", fg="white", command=exportOut, bd=3)
515
516 #Positioning buttons on main page

```

```

517 ExportBtn.place(x=1343, y=111)
518
519 deleteBtn.place(x=940, y=570)
520 addBtn.place(x=940, y=690)
521
522 customerBtn.place(x=1070, y=560)
523 selectBtn.place(x=1275, y=560)
524
525 searchBtn.place(x=1070, y=680)
526 resetBtn.place(x=1275, y=680)
527
528
529 #Tkinter Tree (for inventory)
530 style = ttk.Style()
531 style.configure("Treeview.Heading", font=('Helvetica Bold', 12), relief="flat", fieldbackground="black")
532 style.theme_use("default")
533
534 #Creating columns
535 my_tree['columns'] = ("ID", "Pname", "quantity", "expdate")
536 my_tree.column("#0", width=0, stretch=YES)
537 my_tree.column("ID", anchor="center", width=40, stretch=YES)
538 my_tree.column("Pname", anchor="center", width=380, stretch=YES)
539 my_tree.column("quantity", anchor="center", width=130, stretch=YES)
540 my_tree.column("expdate", anchor="center", width=240, stretch=YES)
541
542 #Headings for the columns
543 my_tree.heading("ID", text="ID", anchor="center")
544 my_tree.heading("Pname", text="Product Name", anchor="center")
545 my_tree.heading("quantity", text="Quantity (KG)", anchor="center")
546 my_tree.heading("expdate", text="Expiration Date (YYYY-MM-DD)", anchor="center")
547
548 refreshTable()
549
550 #Continues to call the "root" which is the main page.
551 root.mainloop()

```

In [ ]: ▶ 1