```
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():
                     conn = pymysql.connect(
             13
             14
                         host='localhost',
             15
                         user='root',
                         password='12345',
             16
                         db='inventory')
             17
             18
                     return conn
             19
             20 def exportOut():
                     fileName = str(fileNameEntry.get())
             21
                     conn = connection()
             22
             23
                     cursor= conn.cursor()
             24
                     cursor.execute("SELECT * FROM Inventory")
             25
                     with open(fileName+".csv", "w") as csv_file:
                         csv writer = csv.writer(csv file,delimiter="\t")
             26
                         csv writer.writerow([i[0] for i in cursor.description])
             27
                         csv writer.writerows(cursor)
             28
             29
                     dir path = os.getcwd() + "/"+fileName+".csv"
                     messagebox.showinfo("success", "Exported Successfully")
             30
             31
             32
             33
             34 #Read function used to read from the tables from the database
             35 def read():
                     #Uses connection() function to interact with database
             36
                     conn = connection()
             37
                     cursor = conn.cursor()
             38
             39
                     #Executes query from python as if it was writing in MySQL.
                     #Selects all item from the inve ntory table
             40
                     cursor.execute("SELECT * FROM Inventory")
             41
                     results = cursor.fetchall()
             42
                     conn.commit()
             43
```

```
conn.close()
44
45
       return results
46
47
48 #Function used to update inventory diaplay in GUI whenevr a change is made
49 def refreshTable():
50
       for data in my tree.get children():
51
           my tree.delete(data)
52
53
       #Uses read() function to fetch the updated table and repalces it for the old one
54
       for array in read():
55
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)
       if num ==2:
85
           ph2.set(word)
86
```

```
if num ==3:
 87
 88
            ph3.set(word)
 89
        if num ==4:
 90
            ph4.set(word)
 91
 92
 93 #Function called whenver the add button is pressed. It adds item to the inventory table in MySQL
 94 def add():
         #Assign variables for the information colected from the data entries from the GUI
 95
        ID = str(IDEntry.get())
 96
        Pname = str(PnameEntry.get())
 97
        quantity = str(quantityEntry.get())
        expdate = str(expdateEntry.get())
 99
100
101
        #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 == "
102
103
            messagebox.showinfo("Error", "Please fill up the blank entry")
104
            return
105
        else:
106
            try:
                conn = connection()
107
108
                 cursor = conn.cursor()
                 #Runs query which adds the variables containing the information from the data entries, into the Inventory table in the databas
109
                cursor.execute("INSERT INTO Inventory VALUES ('"+ID+"','"+Pname+"','"+quantity+"','"+expdate+"') ")
110
111
                 conn.commit()
112
                conn.close()
113
            except:
                messagebox.showinfo("Error", "While (ADDING)")
114
115
                 return
        #Change is made to the table so it has to be updated
116
117
        refreshTable()
118
119 #The reset() fucntion 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
        conditional = messagebox.askquestion("Warning!!", "Do you want to reset the inventory?")
122
123
        if conditional != "yes":
124
            return
125
        else:
126
            try:
127
                conn = connection()
128
                 cursor = conn.cursor()
129
                #Query that resets the Iventory table from the database
```

```
130
                 cursor.execute("DELETE FROM Inventory")
131
                 conn.commit()
                 conn.close()
132
133
             except:
134
                 messagebox.showinfo("Error", "Sorry an error occured While (RESETING)")
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
         decision = messagebox.askquestion("Warning!!", "Delete the selected data?")
142
         if decision != "yes":
143
144
             return
145
         else:
146
             #Selects in MySQL Inventory table, the item selected in the GUI
             selected item = my tree.selection()[0]
147
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
                 cursor.execute("DELETE FROM Inventory WHERE ID='"+str(deleteData)+"'")
153
154
                 conn.commit()
                 conn.close()
155
156
             except:
                 messagebox.showinfo("Error", "Sorry an error occured While (DELETING)")
157
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)
             selected item = my tree.selection()[0]
166
             ID = str(my tree.item(selected item)['values'][0])
167
            Pname = str(my tree.item(selected item)['values'][1])
168
             quantity = str(my tree.item(selected item)['values'][2])
169
             expdate = str(my tree.item(selected item)['values'][3])
170
171
172
             #Function assigns these variables to ph values from setph() function
```

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

```
216
217
            #Creating tkinter tree for popup window
            top.title("Customer System")
218
219
            my tree2 = ttk.Treeview(top, height=15, selectmode="browse")
220
            #Refresh functrion but for pop up window customers table
221
222
            def refreshTablePU():
223
                for data in my tree2.get children():
                    my tree2.delete(data)
224
225
226
                for array in readPU():
227
                    my tree2.insert(parent='', index='end', iid=array, text="", values=(array), tag="orow")
228
                my tree2.tag configure('orow', background='White', font=('Arial', 15))
229
230
                my tree2.place(x=20, y=20)
231
232
            #Variables values from Tkinter tree columns
233
234
            var1 = tk.StringVar()
            var2 = tk.StringVar()
235
236
            var3 = tk.StringVar()
237
            var4 = tk.StringVar()
238
            #Read function for pop up window
239
240
            def readPU():
                conn = connection()
241
242
                cursor = conn.cursor()
                #Query fetches all the data from Customers table
243
                cursor.execute("SELECT * FROM CUSTOMER")
244
245
                 results = cursor.fetchall()
246
                conn.commit()
247
                 conn.close()
248
                 return results
249
250
            #Adding item function to customers table
            def addPU():
251
252
                 #Assigning the information from the data entries to variables
                Customer ID = str(Customer IDEntry.get())
253
                Customer name = str(Customer nameEntry.get())
254
255
                Customer OwedToUs = str(Customer OwedToUsEntry.get())
                Customer OwedToThem = str(Customer OwedToThemEntry.get())
256
257
258
                 #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 Customer_name == " ")
259
                     messagebox.showinfo("Error", "Please fill up the blank entry")
260
261
                     return
262
                 else:
263
                     try:
264
                         conn = connection()
265
                         cursor = conn.cursor()
266
                         #Query that inserts previosuly created variables, into the customers table
                         cursor.execute("INSERT INTO CUSTOMER VALUES ('"+Customer ID+"', '"+Customer name+"', '"+Customer OwedToUs+"', '"+Customer
267
268
                         conn.commit()
                         conn.close()
269
270
                     except:
                         messagebox.showinfo("Error", "While (ADDING) Customer")
271
272
                         return
273
                 #Change is amde to CUSTOMER table so it has to be updated
274
                 refreshTablePU()
275
276
            #Function used to fetch specific data from an item on the customers table
277
278
             def setphPU(word,num):
                 if num ==1:
279
280
                     var1.set(word)
281
                 if num ==2:
282
                     var2.set(word)
283
                 if num ==3:
284
                     var3.set(word)
                 if num ==4:
285
286
                     var4.set(word)
287
288
289
             #Function that deletes selecteed item from CUSTOMER table
290
             def deletePU():
                 decision = messagebox.askquestion("Warning!!", "Delete the selected data?")
291
                 if decision != "yes":
292
293
                     return
294
                 else:
295
                     #Gathering the information of the selected item
                     selected item = my tree2.selection()[0]
296
                     deleteData = str(my tree2.item(selected item)['values'][0])
297
298
                 try:
299
                     conn = connection()
300
                     cursor = conn.cursor()
                     #Deleting the selected item using the deleteData variable which is created above,
301
```

```
302
                     #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)+"'")
303
304
                     conn.commit()
                     conn.close()
305
306
                 except:
                     messagebox.showinfo("Error", "Sorry an error occured While (DELETING)")
307
308
                     return
309
                refreshTablePU()
310
311
312
            def resetPU():
313
                decision = messagebox.askquestion("Warning!!", "Delete all data?")
314
                if decision != "yes":
315
316
                     return
317
                 else:
318
                     try:
                         conn = connection()
319
                         cursor = conn.cursor()
320
321
                         cursor.execute("DELETE FROM CUSTOMER")
322
                         conn.commit()
                         conn.close()
323
324
                     except:
                         messagebox.showinfo("Error", "Sorry an error occured While (RESETING)")
325
326
                         return
327
                     refreshTablePU()
328
329
            def selectPU():
330
331
                 try:
                     selected item = my tree2.selection()[0]
332
                     Customer ID = str(my tree2.item(selected item)['values'][0])
333
                     Customer name = str(my tree2.item(selected item)['values'][1])
334
                     Customer OwedToUs = str(my tree2.item(selected item)['values'][2])
335
336
                     Customer OwedToThem = str(my tree2.item(selected item)['values'][3])
337
338
                     setphPU(Customer ID,1)
                     setphPU(Customer name, 2)
339
                     setphPU(Customer OwedToUs,3)
340
                     setphPU(Customer OwedToThem,4)
341
342
                 except:
                     messagebox.showinfo("Error", "Please select a data row While (SELECTING)")
343
344
```

```
345
346
            def searchPU():
                Customer ID = str(Customer_IDEntry.get())
347
                Customer name = str(Customer nameEntry.get())
348
                Customer OwedToUs = str(Customer OwedToUsEntry.get())
349
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()
                    #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.
358
359
                    for num in range(0,4):
                        setphUP(result[0][num],(num+1))
360
361
                    conn.commit()
362
363
                    conn.close()
364
                except:
                    messagebox.showinfo("Error", "No data found While (SEARCHING)")
365
366
367
368
369
        #POP UP WINDOW GUI:
370
371
            #Tkinter tree columns creation
            my tree2['columns'] = ("Customer ID","Customer name","Customer OwedToUs","Customner OwedToThem")
372
            my tree2.column("#0", width=0, stretch=YES)
373
374
            my tree2.column("Customer ID", anchor="center", width=30, stretch=YES)
            my tree2.column("Customer name", anchor="center", width=300, stretch=YES)
375
            my tree2.column("Customer OwedToUs", anchor="center", width=150, stretch=YES)
376
377
            my tree2.column("Customner OwedToThem", anchor="center", width=250, stretch=YES)
378
379
            #Heading for the columns
            my tree2.heading("Customer ID", text="ID", anchor="center")
380
            my tree2.heading("Customer name", text="Customer", anchor="center")
381
            my tree2.heading("Customer OwedToUs", text="Owed to Us", anchor="center")
382
            my tree2.heading("Customner OwedToThem", text="Owed to Customer", anchor="center")
383
384
385
            refreshTablePU()
386
387
            #Creating labels for data entries
```

```
|", font=('Arial bold', 15), bg="Dark Green", fg="white")
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
            Customner OwedToThemLabel = Label(top, text=" | Owed to Customer | ", font=('Arial bold', 15), bg="Dark Green", fg="white")
391
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
            Customer IDEntry = Entry(top, width=45, bd=5, font=('Arial', 15), textvariable = var1)
395
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
            #Positioning Data entries in pop up window
406
407
            Customer IDEntry.place(x=210, y=400)
            Customer nameEntry.place(x=210, y=460)
408
409
            Customer OwedToUsEntry.place(x=210, y=520)
410
            Customer OwedToThemEntry.place(x=210, y=580)
411
412
            #Creating buttons for pop up window
            #have to use "top" instead of "root", beacuse top makes reference to widgets on the pop up window
413
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(
                top, text="+", padx=5, pady=5, width=5, font=('Arial', 15), bg="Dark Green", command=addPU, fg="white",bd=3)
418
419
            searchCustomersBtn = Button(
                top, text="Search", padx=15, pady=15, width=10, font=('Arial', 15), bg="Dark Green", command=searchPU, fg="white", bd=3)
420
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)
            selectCustomersBtn = Button(
423
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)
            resetCustomersBtn.place(x=570, y=650)
432
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="
                                                                      ", font=('Arial Bold', 40), bg= "#ace5ee", bd=10, fg="#65350F")
                               LCacao Inventario
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="
                   font=('Arial bold', 17), bg="#001C57", fg="white", bd = 1)
461
                                                                                                               |",
462 PnameLabel = Label(root, text="
                                                                    Product Name
                      font=('Arial bold', 17), bg="#001C57", fg="white")
463
464 quantityLabel = Label(root, text="
                                                                            Quantity
                         font=('Arial bold', 17), bg="#001C57", fg="white")
465
                                                                                                                  |",
466 expdateLabel = Label(root, text="
                                                                    Expiration Date
                        font=('Arial bold', 17), bg="#001C57", fg="white")
467
                                                                                            |",
468 fileNameLabel = Label(root, text="
                                                         Export File Name
                         font=('Arial bold', 17), bg="#001C57", fg="white")
469
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(
        root, text="Reset", padx=15, pady=15, width=10, font=('Arial', 18), bg="#001C57", command=reset, fg="white", bd=3)
508
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(
        root, text="Customers", padx=15, pady=15, width=10, font=('Arial', 18), bg="#001C57", command=popup.open popup, fg="white", bd=3)
512
513 Exportbttn = 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 Exportbttn.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()
```