```
1  ##########################################################
   Before Running the code ensure that following libraries are installed on your python 3.8 and
   with add to path enabled=========================================================
2  #1)reportlab  2)mysql.connector-python  3)pandas  #4)matplotlib  #5)IPython  #6)
   PDFNetPython3  7)sqlalchemy
3  ##############################If above libraries go to command prompt  execute
   the following commands one after one with internet connection
    ##########################################################################
   ########
4  #1)pip install reportlab  #2)pip install mysql.connector-python  3)#pip install IPython #4)pip
   install PDFNetPython3 #5)pip install matplotlib  #6)pip install pandas  #7)pip install
   sqlalchemy
5  #=========================================================================
   ===========All import Statements=============================================
6  from reportlab.platypus import Table
7  from tkinter import *
8  from PIL import Image as pilimage, ImageTk
9  from tkinter import messagebox
10 from tkinter import ttk
11 import site
12 site.addsitedir("../../../PDFNetC/Lib")
13 import sys
14 from PDFNetPython3 import *
15 from reportlab.platypus import SimpleDocTemplate
16 from reportlab.platypus import TableStyle
17 from reportlab.lib.pagesizes import A4
18 from reportlab.lib import colors
19 import mysql.connector as sql
20 import sqlalchemy as alsql
21 from datetime import date
22 from datetime import datetime
23 import os
24 import pandas as pd
25 import matplotlib.pyplot as plt
26 from IPython.display import display
27 #=========================================================================
   ======Preparation of Mysql
    ========================================================================
   ========================================
28 cnx =sql.connect(user='root', password='danger',host='localhost')
29 alsqlengine=alsql.create_engine('mysql+mysqlconnector://root:danger@localhost/project')
30 cursor=cnx.cursor()
31 try:
32     cursor.execute('use project')
33 except:
34     cursor.execute('create database project')
35     cursor.execute('use project')
36 try:
```

```
37      cursor.execute('select * from credentials')
38      s=cursor.fetchall()
39      for x in s:
40          print(x[0],'\t',x[1],'\t',x[2],'\t',x[3])
41  except:
42      cursor.execute(
43          'create table credentials(sno integer,username varchar(100),usertype varchar(100),
    password varchar(100))')
44      cursor.execute('alter table credentials add primary key(username)')
45      cursor.execute("insert into credentials values (1,'1','testing','1'),(2,'user1','employee','
    password1'),(3,'user2','employee','password2'),(4,'user3','employee','password3'),(5,'admin
    ','owner','admin')")
46      cursor.execute('select * from credentials')
47      s=cursor.fetchall()
48      for x in s:
49          print(x[0],'\t',x[1],'\t',x[2],'\t',x[3])
50  try:
51      cursor.execute('select * from olditems')
52      data=cursor.fetchall()
53      print(data)
54  except:
55      csvfile=os.getcwd()+'\\resources\Itemsdata.csv'
56      itemsdataframe=pd.read_csv(csvfile)
57      print(itemsdataframe)
58      alsqlcn=alsqlengine.connect()
59      itemsdataframe.to_sql('olditems',alsqlcn)
60      alsqlcn.close()
61      alsqlengine.dispose()
62      cursor.execute('drop table if exists items ')
63      cursor.execute('CREATE TABLE items(select sno,itemname,itemcode,quantity,price,
    company from project.olditems)')
64      cnx.commit()
65  try:
66      cursor.execute('use orders')
67  except:
68      cursor.execute('create database orders')
69      cnx.commit()
70  cnx.close()
71  #================================================================================
    ==============Initialising mainwindow
    ============================================================
72  ifc = Tk()
73  ifc.resizable(width=FALSE, height=FALSE)
74  # ============================================================================
    Strings and StringVars
    ================================================================================####
    ###############
75  todaysdate = str(date.today().strftime('%d-%m-%Y'))
```

```
76  datewithouthyphens = todaysdate.replace('-', '_')
77  print(datewithouthyphens)
78  user = StringVar()
79  password = StringVar()
80  ###################==========================================================
    =====All images
    ============================================================############
    ############
81  img_submit = PhotoImage(file=os.getcwd()+'\\resources\submit.png')
82  img_back = PhotoImage(file=os.getcwd()+'\\resources\\back.png')
83  img_logout = PhotoImage(file=os.getcwd()+'\\resources\logout.png')
84  img_changepassword = PhotoImage(file=os.getcwd()+'\\resources\changepassword.png')
85  img_viewitems = PhotoImage(file=os.getcwd()+'\\resources\\viewitems.png')
86  img_orderitems = PhotoImage(file=os.getcwd()+'\\resources\orderitems.png')
87  img_changeprices = PhotoImage(file=os.getcwd()+'\\resources\changeprices.png')
88  img_manageitems = PhotoImage(file=os.getcwd()+'\\resources\manageitems.png')
89  img_analyzedata = PhotoImage(file=os.getcwd()+'\\resources\\analyzedata.png')
90  img_viewandmanageitems = PhotoImage(file=os.getcwd()+'\\resources\\viewandmanageitems.
    png')
91  img_manageusers = PhotoImage(file=os.getcwd()+'\\resources\manageusers.png')
92  LOGIN = PhotoImage(file=os.getcwd()+'\\resources\\newlogin.png')
93  EXIT = PhotoImage(file=os.getcwd()+'\\resources\\newexit.png')
94  #================================================================================
    ==============Mainexit button
    ================================================
95  def mainexit():
96      os._exit(0)
97  ##########################################################################
    ######################All the Classes i.e frames
    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    ==========================================
98  class orderhistory():
99      def __init__(self,ifc):
100         self.ifc=ifc
101         ifc.geometry('1600x1000')
102         ifc.title('Order History')
103         ifc.configure(bg='green')
104         def back1dashboard():
105             mainhistoryframe.pack_forget()
106             for widget in mainhistoryframe.winfo_children():
107                 widget.pack_forget()
108             Dashboard(ifc)
109         def insert_allorders():
110             cn = sql.connect(host='localhost', user='root', password='danger', database='
    orders')
111             c = cn.cursor()
112             c.execute('select table_name,date_format(create_time,\'%d%m%y%h%m%s\')
    from information_schema.tables where table_schema=\'orders\'')
```

```
113             table_names=[]
114             table_times=[]
115             amounts=[]
116             historydata=[]
117             for ordertuple in list(c.fetchall()):
118                 historydata.append(list(ordertuple))
119             print(historydata)
120             for order in historydata:
121                 table_names.append(order[0])
122                 entryatetime=str(order[1])
123                 table_times.append(entryatetime[0:2]+'-'+entryatetime[2:4]+'-'+entryatetime[
    4:6]+'  '+entryatetime[6:8]+':'+entryatetime[8:10]+':'+entryatetime[10:12])
124             print(table_times)
125             print(table_names)
126             for table in table_names:
127                 c.execute('select sum(price) from '+str(table))
128                 amounts.append(int(str(c.fetchall()).replace('[(Decimal(\'', '').replace('\'),)]',
    '')))
129             print(amounts)
130             for info in range(0,len(table_names)):
131                 ordervalue=(table_names[info],table_times[info],amounts[info])
132                 records_orders.insert('',END,values=ordervalue)
133             cn.commit()
134             cn.close()
135         def order_info(ev):
136             records_orderinfo.delete(*records_orderinfo.get_children())
137             cn = sql.connect(host='localhost', user='root', password='danger', database='
    orders')
138             c = cn.cursor()
139             view_orderinfo = records_orders.focus()
140             getorderinfo = records_orders.item(view_orderinfo)
141             ordernoinfo= list(getorderinfo['values'])
142             c.execute('select * from '+str(ordernoinfo[0]))
143             order_data=c.fetchall()
144             print(order_data)
145             for itemdetail in order_data:
146                 itemdetailtuple=(itemdetail[0],itemdetail[1],itemdetail[2],itemdetail[3],
    itemdetail[4],itemdetail[5],itemdetail[6])
147                 records_orderinfo.insert('',END,values=itemdetailtuple)
148             cn.commit()
149             cn.close()
150         def order_reciept(ev):
151             view_orderinfo = records_orders.focus()
152             getorderinfo = records_orders.item(view_orderinfo)
153             ordernoinfo= list(getorderinfo['values'])
154             print('ordernoinfo',ordernoinfo)
155             entryate=str(ordernoinfo[0])[-10:]
156             print('entryate',entryate)
```

```
157             recieptfile=str(os.getcwd())+'\Order_Reciepts\Reciepts_dated_'+str(entryate)+'\
    Reciept_of_'+str(ordernoinfo[0])+'.pdf'
158             os.startfile(recieptfile)
159 
160         mainhistoryframe=Frame(ifc,bd=10,bg='blue')
161         mainhistoryframe.pack(fill=BOTH,expand=YES)
162         topframeforhistory=Frame(mainhistoryframe,bd=10,bg='blue',width=1600,height=900)
163         topframeforhistory.pack(fill=BOTH, expand=YES)
164         bottomframeforhistory=Frame(mainhistoryframe,bd=10,bg='blue',width=1600,height=
    900)
165         bottomframeforhistory.pack(fill=BOTH, expand=YES)
166         leftframeforhistory=Frame(topframeforhistory,width=600,height=900,bd=10,bg='gold'
    )
167         leftframeforhistory.pack(fill=BOTH,expand=YES,side=LEFT)
168         orderinfoframe=Frame(topframeforhistory,width=1000,height=900,bd=10,bg='gold')
169         orderinfoframe.pack(fill=BOTH,expand=YES,side=RIGHT)
170         instruction='''Double Click To view Order Details,Right Click To View Order Reciept'''
171 
172         historyframe=Frame(leftframeforhistory,width=600,height=950,bd=10,bg='gold')
173         historyframe.pack(fill=BOTH,expand=YES,side=BOTTOM)
174         label_instructions=Label(leftframeforhistory,text=instruction,font='rockwell 14 bold')
175         label_instructions.pack(side=TOP)
176 
177         button_back = Button(bottomframeforhistory, text='Back to Dashboard', font='
    comicsansms 14 bold', bg='gold',
178                              fg='blue', bd=10, command=back1dashboard)
179         button_back.pack(side=LEFT)
180         button_exit = Button(bottomframeforhistory, text='Exit', bg='green',fg='blue', font=
    'segoeuiblack 14 bold italic', bd=10,
181                              command=mainexit)
182         button_exit.pack(side=RIGHT)
183         global records_orders
184         records_orders = ttk.Treeview(historyframe, height=26,columns=('Orderno','Order
    Date and Time','Order Amount'))
185         scroll_x = ttk.Scrollbar(historyframe, orient=HORIZONTAL, command=
    records_orders.xview)
186         scroll_y = ttk.Scrollbar(historyframe, orient=VERTICAL, command=records_orders.
    yview)
187         records_orders.configure(yscrollcommand=scroll_y.set, xscrollcommand=scroll_x.set)
188         scroll_x.pack(side=BOTTOM, fill=X)
189         scroll_y.pack(side=RIGHT, fill=Y)
190         records_orders.heading('Orderno', text='Orderno')
191         records_orders.heading('Order Date and Time', text='Order Date and Time')
192         records_orders.heading('Order Amount', text='Order Amount')
193 
194         records_orders['show'] = 'headings'
195         records_orders.column('#0', width=150, minwidth=130)
196         records_orders.column('#1', width=150, minwidth=130)
```

```
197         records_orders.column('#2', width=100, minwidth=50)
198         records_orders.pack(fill=BOTH, expand=YES, side=BOTTOM)
199         insert_allorders()
200         records_orders.bind('<Double-1>',order_info)
201         records_orders.bind('<Button-3>',order_reciept)
202 
203 
    ##########################################################################
    #######Another Treeview for Order Details
    ##########################################################################
    ######
204         records_orderinfo = ttk.Treeview(orderinfoframe, height=26,columns=('Itemno','Sno'
    , 'Item Name', 'Item Code', 'Quantity', 'Price', 'Brand'))
205         scrollr_x = ttk.Scrollbar(orderinfoframe, orient=HORIZONTAL, command=
    records_orderinfo.xview)
206         scrollr_y = ttk.Scrollbar(orderinfoframe, orient=VERTICAL, command=
    records_orderinfo.yview)
207         records_orderinfo.configure(yscrollcommand=scrollr_y.set, xscrollcommand=scrollr_x.
    set)
208         scrollr_x.pack(side=BOTTOM, fill=X)
209         scrollr_y.pack(side=RIGHT, fill=Y)
210         records_orderinfo.heading('Itemno', text='Item No')
211         records_orderinfo.heading('Sno', text='Sno')
212         records_orderinfo.heading('Item Name', text='Item Name')
213         records_orderinfo.heading('Item Code', text='Item Code')
214         records_orderinfo.heading('Quantity', text='Quantity')
215         records_orderinfo.heading('Price', text='Price')
216         records_orderinfo.heading('Brand', text='Brand')
217         records_orderinfo['show'] = 'headings'
218 
219         records_orderinfo.column('#0', width=100, minwidth=60)
220         records_orderinfo.column('#1', width=100, minwidth=60)
221         records_orderinfo.column('#2', width=200, minwidth=180)
222         records_orderinfo.column('#3', width=200, minwidth=140)
223         records_orderinfo.column('#4', width=100, minwidth=80)
224         records_orderinfo.column('#5', width=100, minwidth=80)
225         records_orderinfo.column('#6', width=200, minwidth=150)
226         records_orderinfo.pack(fill=BOTH, expand=YES)
227 
228 class analysedata():
229     def __init__(self, ifc):
230         cn = sql.connect(host='localhost', user='root', password='danger', database='orders')
231         c = cn.cursor()
232         maindataframe = pd.DataFrame()
233         tablelist = []
234         c.execute("select table_name from information_schema.tables where table_schema=\'
    orders\'")
235         for table in c.fetchall():
```

```
            tablelist += table
        for tablename in tablelist:
            query_todf = 'select * from ' + str(tablename)
            tempdf = pd.read_sql(query_todf, cn)
            maindataframe = pd.concat([tempdf, maindataframe], axis=0, ignore_index=True)
        cn.commit()
        cn.close()
        print('maindataframe', maindataframe)
        maindataframe['count'] = 1
        print('maindataframe', maindataframe)
        items = list(maindataframe['itemname'])
        print(items)
        itemcount = {}
        # itemcount=dict(zip(items,list(maindataframe['count'])))
        print(itemcount)
        print(len(items))
        for i in range(0, len(items)):
            itemcount[str(items[i])] = int(items.count(items[i]))
        print(itemcount)

        profitlist = maindataframe[['itemname', 'price']]
        profitlist.loc[:, 'profitperitem'] = round((0.1) * profitlist.loc[:, 'price'])
        print('profitlist', profitlist)
        profitlist = profitlist.drop(index=profitlist[profitlist.duplicated(['itemname'])].index)
        print('profitlist')
        print(profitlist)
        profitlist['itemcount']=itemcount.values()
        profitlist['totalprofitfromitem']=profitlist['profitperitem']*profitlist['itemcount']
        print('profitlist')
        print(profitlist)
        print('Duplicate values', profitlist[profitlist.duplicated(['itemname'])])
        def analysefrequentitems():
            itemsx=list(range(0,len(itemcount.keys())))
            plt.bar(itemsx, itemcount.values(), color='green')
            plt.xticks(itemsx,itemcount.keys(),rotation='vertical')
            plt.title('Items Analysis')
            plt.margins(0.2)
            plt.ylabel('-----Number of times it has been orderd ---------')
            plt.xlabel('-----------Item-----------------')
            # Tweak spacing to prevent clipping of tick-labels
            plt.subplots_adjust(bottom=0.30)
            plt.show()
        def analyseprofitperitem():
            plt.bar(profitlist['itemname'],profitlist['totalprofitfromitem'], color='green')
            plt.xticks(profitlist['itemname'],profitlist['itemname'],rotation='vertical')
            plt.title('Items Analysis')
            plt.margins(0.2)
            plt.ylabel('-----Profit per unit item---------')
```

```
284             plt.xlabel('-----------Item-----------------')
285             # Tweak spacing to prevent clipping of tick-labels
286             plt.subplots_adjust(bottom=0.30)
287             plt.show()
288         def backdashboard():
289             frameforanalysis.pack_forget()
290             for widget in frameforanalysis.winfo_children():
291                 widget.pack_forget()
292             Dashboard(ifc)
293         ifc.geometry('1000x800')
294         ifc.configure(bg='red')
295         ifc.title('DataAnalysis')
296         global frameforanalysis
297         frameforanalysis = Frame(ifc, bg='brown', bd=10)
298         frameforanalysis.pack(fill=BOTH, expand=YES)
299         button_mostlysolditems = Button(frameforanalysis,
300                                         text='Analyze which items are mostly sold out from
    orders till now',
301                                         font='comicsansms 14 bold', bg='gold', fg='blue', bd=10,
302                                         command=analysefrequentitems)
303         button_mostlysolditems.pack()
304         button_profitperitem = Button(frameforanalysis,
305                                         text='Analyze which items are profit per item',
306                                         font='comicsansms 14 bold', bg='gold', fg='blue', bd=10,
307                                         command=analyseprofitperitem)
308         button_profitperitem.pack()
309         button_back = Button(frameforanalysis, text='Back to Dashboard', font='comicsansms
     14 bold', bg='gold',
310                              fg='blue', bd=10, command=backdashboard)
311         button_back.pack(side=LEFT)
312         button_exit = Button(frameforanalysis, text='Exit', bg='green',fg='blue', font='
    segoeuiblack 14 bold italic', bd=10,
313                              command=mainexit)
314         button_exit.pack(side=RIGHT)
315 class items_management():
316     def __init__(self, ifc):
317         self.ifc = ifc
318         ifc.title('Items Management')
319         ifc.geometry('1800x1000+0+0')
320         ifc.configure(bg='black')
321
322         # ======================================Variables
    ========================================#
323         sno = StringVar()
324         itemname = StringVar()
325         itemcode = StringVar()
326         quantity = StringVar()
327         price = StringVar()
```

```
328         brand = StringVar()
329         searchwith = StringVar()
330         searchtext = StringVar()
331         searchorder = StringVar()
332 
333         # ==================================functions
    ==========================================#
334         def back5():
335             mainframeforitems.place_forget()
336             for widget in mainframeforitems.winfo_children():
337                 widget.place_forget()
338             ifc.title('ShopKeeper Dashboard')
339             ifc.geometry('800x600')
340             # ifc.geometry('800x600')
341             # Dashboard(ifc)
342             # optional if u use keeperdashboard.pack_forget() in command_itemsmgt()
343 
344         def clear():
345             sno.set('')
346             itemname.set('')
347             itemcode.set('')
348             quantity.set('')
349             price.set('')
350             brand.set('')
351             searchwith.set('itemname')
352             searchtext.set('')
353             searchorder.set('asc')
354             search()
355 
356         def add_item():
357             if sno.get() == '' or itemname.get() == '' or itemcode == '':
358                 messagebox._show(message='Few Entries are Empty')
359             else:
360                 cn = sql.connect(host='localhost', user='root', password='danger', database='
    project')
361                 c = cn.cursor()
362                 query_add = 'insert into items values (%s,%s,%s,%s,%s,%s)'
363                 tuple_add = (sno.get(), itemname.get(), itemcode.get(), quantity.get(), price.
    get(), brand.get())
364                 c.execute(query_add, tuple_add)
365                 cn.commit()
366                 records_items.delete(*records_items.get_children())
367                 insert_items()
368                 cn.close()
369                 messagebox._show(message='Record Added Successfully')
370                 clear()
371 
372         def delete_item():
```

```
373             cn = sql.connect(host='localhost', user='root', password='danger', database='
    project')
374             c = cn.cursor()
375             query_delete = 'delete from items where sno=' + str(sno.get())
376             c.execute(query_delete)
377             cn.commit()
378             records_items.delete(*records_items.get_children())
379             insert_items()
380             cn.close()
381             messagebox._show(message='Record Deleted Successfully')
382 
383             clear()
384 
385         def update_item():
386             cn = sql.connect(host='localhost', user='root', password='danger', database='
    project')
387             c = cn.cursor()
388             query_update = 'update items set itemname=%s,itemcode=%s,quantity=%s,price=%
    s,company=%s where sno=%s'
389             tuple_update = (itemname.get(), itemcode.get(), quantity.get(), price.get(), brand.
    get(), sno.get())
390             c.execute(query_update, tuple_update)
391             cn.commit()
392             records_items.delete(*records_items.get_children())
393             insert_items()
394             cn.close()
395             messagebox._show(message='Record Updated Successfully')
396             clear()
397 
398         global insert_items
399 
400         def insert_items():
401             cn = sql.connect(host='localhost', user='root', password='danger', database='
    project')
402             c = cn.cursor()
403             c.execute('select * from items')
404             rows = c.fetchall()
405             for row in rows:
406                 records_items.insert('', END, values=row)
407                 cn.commit()
408             cn.close()
409 
410         def item_info(ev):
411             view_iteminfo = records_items.focus()
412             getiteminfo = records_items.item(view_iteminfo)
413             record = getiteminfo['values']
414             sno.set(record[0])
415             itemname.set(record[1])
```

```
416             itemcode.set(record[2])
417             quantity.set(record[3])
418             price.set(record[4])
419             brand.set(record[5])
420 
421         def delete_all():
422             cn = sql.connect(host='localhost', user='root', password='danger', database='
    project')
423             c = cn.cursor()
424             c.execute('delete from items')
425             cn.commit()
426             cn.close()
427             records_items.delete(*records_items.get_children())
428 
429         def search():
430             cn = sql.connect(host='localhost', user='root', password='danger', database='
    project')
431             c = cn.cursor()
432             query_search = 'select * from items where ' + str(searchwith.get()) + ' like ' + '
    \'%' + str(
433                 searchtext.get()) + '%\'' + ' order by ' + str(searchwith.get()) + ' ' + str(
    entry_searchorder.get())
434             if len(str(searchwith)) != 0 or len(str(searchorder)) != 0:
435                 if len(str(searchtext)) != 0:
436                     c.execute(query_search)
437                     searched_rows = c.fetchall()
438                     records_items.delete(*records_items.get_children())
439                     for searchedrow in searched_rows:
440                         records_items.insert('', END, values=searchedrow)
441                 else:
442                     c.execute(query_search)
443                     searched_rows = c.fetchall()
444                     records_items.delete(*records_items.get_children())
445                     for searchedrow in searched_rows:
446                         records_items.insert('', END, values=searchedrow)
447             cn.commit()
448             cn.close()
449 
450         # ================================Frames
    ================================================#
451         mainframeforitems = Frame(ifc, bg='gold', width=1780, height=994, bd=10, relief=
    RIDGE)
452         mainframeforitems.place(x=0, y=0)
453         leftframe = Frame(mainframeforitems, bg='white', width=580, height=690, bd=10,
    relief=RIDGE)
454         leftframe.place(x=2, y=0)
455         rightframe = Frame(mainframeforitems, bg='white', width=1200, height=590, bd=10
    , relief=RIDGE)
```

```
456         rightframe.place(x=610, y=100)
457         itemslabelframe = Frame(leftframe, width=200, height=696, relief=RIDGE, bg='brown
    ', bd=8)
458         itemslabelframe.grid(row=0, column=0)
459         itementriesframe = Frame(leftframe, width=400, height=696, relief=RIDGE, bg='
    darkviolet', bd=8)
460         itementriesframe.grid(row=0, column=1)
461         recordsframe = Frame(rightframe, bg='darkblue', width=1200, height=590, bd=10,
    relief=RIDGE)
462         recordsframe.place(x=0, y=0)
463         searchframe = Frame(mainframeforitems, bg='violet', width=1200, height=95, bd=5)
464         searchframe.place(x=600, y=0)
465         bottomframe = Frame(mainframeforitems, width=1790, height=396, bd=5, relief=
    RIDGE, bg='red')
466         bottomframe.place(x=0, y=697)
467         # ==================================Labels&Entries
    ==========================================#
468         lbl_sno = Label(itemslabelframe, text='Sno', font='segoeuiblack 24 bold', bd=10)
469         lbl_sno.grid(row=0, column=0, padx=18, pady=18)
470         entry_sno = Entry(itementriesframe, bd=10, width=16, font='rockwell 24',
    textvariable=sno)
471         entry_sno.grid(row=0, column=0, padx=18, pady=18)
472         lbl_itemname = Label(itemslabelframe, text='Item Name', font='segoeuiblack 24 bold
    ', bd=10)
473         lbl_itemname.grid(row=1, column=0, padx=18, pady=18)
474         entry_itemname = Entry(itementriesframe, bd=10, width=16, font='rockwell 24',
    textvariable=itemname)
475         entry_itemname.grid(row=1, column=0, padx=18, pady=18)
476         lbl_itemcode = Label(itemslabelframe, text='Item Code', font='segoeuiblack 24 bold'
    , bd=10)
477         lbl_itemcode.grid(row=2, column=0, padx=18, pady=18)
478         entry_itemcode = Entry(itementriesframe, bd=10, width=16, font='rockwell 24',
    textvariable=itemcode)
479         entry_itemcode.grid(row=2, column=0, padx=18, pady=18)
480         entry_quantity = ttk.Combobox(itementriesframe, width=16, font='rockwell 24',
    textvariable=quantity)
481         entry_quantity['values'] = ('', '1kg', '1', '0.5 kg', '0.25kg', '100g', '1ltr', '0.5ltr')
482         entry_quantity.grid(row=3, column=0, padx=18, pady=18)
483         lbl_quantity = Label(itemslabelframe, text='Quantity', font='segoeuiblack 24 bold',
    bd=10)
484         lbl_quantity.grid(row=3, column=0, padx=18, pady=18)
485         lbl_price = Label(itemslabelframe, text='Price', font='segoeuiblack 24 bold', bd=10)
486         lbl_price.grid(row=4, column=0, padx=18, pady=18)
487         entry_price = Entry(itementriesframe, bd=10, width=16, font='rockwell 24',
    textvariable=price)
488         entry_price.grid(row=4, column=0, padx=18, pady=18)
489         lbl_brand = Label(itemslabelframe, text='Brand', font='segoeuiblack 24 bold', bd=10)
490         lbl_brand.grid(row=5, column=0, padx=18, pady=18)
```

```
491         entry_brand = Entry(itementriesframe, bd=10, width=16, font='rockwell 24',
    textvariable=brand)
492         entry_brand.grid(row=5, column=0, padx=18, pady=18)
493         lbl_searchwith = Label(searchframe, text='Search With', font='segoeuiblack 24 bold'
    , bd=5)
494         lbl_searchwith.grid(row=0, column=0)
495         entry_searchwith = ttk.Combobox(searchframe, font='segoeuiblack 24 bold',
    textvariable=searchwith)
496         entry_searchwith.grid(row=0, column=1)
497         entry_searchwith['values'] = ('sno', 'itemname', 'itemcode', 'quantity', 'price', '
    company')
498         entry_searchwith.current(1)
499         lbl_searchorder = Label(searchframe, text='Order By', font='segoeuiblack 24 bold',
    bd=5)
500         lbl_searchorder.grid(row=0, column=2)
501         entry_searchorder = ttk.Combobox(searchframe, font='segoeuiblack 24 bold')
502         entry_searchorder.grid(row=0, column=3)
503         entry_searchorder['values'] = ('asc', 'desc')
504         entry_searchorder.current(0)
505         lbl_searchtext = Label(searchframe, text='Search Text', font='segoeuiblack 24 bold'
    , bd=5)
506         lbl_searchtext.grid(row=1, column=0)
507         entry_searchtext = Entry(searchframe, font='segoeuiblack 24 bold', textvariable=
    searchtext)
508         entry_searchtext.grid(row=1, column=1)
509         button_search = Button(searchframe, text='Search', bg='yellow', fg='blue', bd=6,
    command=search,
510                                font='segoeuiblack 20 bold ')
511         button_search.grid(row=1, column=2)
512         # ==================================Buttons
    ================================================#
513         button_additem = Button(bottomframe, text='Add Item', bg='pink', font='
    segoeuiblack 24 bold italic', bd=8,width=20, command=add_item)
514         button_additem.grid(row=0, column=1)
515         button_deleteitem = Button(bottomframe, text='Delete Item', bg='pink', font='
    segoeuiblack 24 bold italic', bd=8,width=20, command=delete_item)
516         button_deleteitem.grid(row=0, column=2)
517         button_updateitem = Button(bottomframe, text='Update Item', bg='pink', font='
    segoeuiblack 24 bold italic', bd=8, width=20, command=update_item)
518         button_updateitem.grid(row=0, column=3)
519         button_clearitem = Button(bottomframe, text='Clear ', bg='pink', font='segoeuiblack
    24 bold italic', bd=8,command=clear, width=20)
520         button_clearitem.grid(row=0, column=4)
521         button_delete_all = Button(bottomframe, text='Delete All', bg='pink', font='
    segoeuiblack 24 bold italic', bd=8,width=20, command=delete_all)
522         button_delete_all.grid(row=1, column=2)
523         button_back = Button(bottomframe, text='Back', bg='pink', font='segoeuiblack 24
    bold italic', bd=8,command=back5, width=20)
```

```
524         button_back.grid(row=1, column=1)
525         button_exit = Button(bottomframe, text='Exit', bg='pink', font='segoeuiblack 24 bold
     italic', bd=8, command=mainexit, width=20, padx=50)
526         button_exit.grid(row=1, column=4, columnspan=2)
527         # ========================================Treeview for viewing items in
    records frame=======================
528 
529         records_items = ttk.Treeview(recordsframe, height=26,
530                                      columns=('Sno', 'Item Name', 'Item Code', 'Quantity', '
    Price', 'Brand'))
531         scroll_x = ttk.Scrollbar(recordsframe, orient=HORIZONTAL, command=records_items
    .xview)
532         scroll_y = ttk.Scrollbar(recordsframe, orient=VERTICAL, command=records_items.
    yview)
533         records_items.configure(yscrollcommand=scroll_y.set, xscrollcommand=scroll_x.set)
534         scroll_x.pack(side=BOTTOM, fill=X)
535         scroll_y.pack(side=RIGHT, fill=Y)
536 
537         records_items.heading('Sno', text='Sno')
538         records_items.heading('Item Name', text='Item Name')
539         records_items.heading('Item Code', text='Item Code')
540         records_items.heading('Quantity', text='Quantity')
541         records_items.heading('Price', text='Price')
542         records_items.heading('Brand', text='Brand')
543         records_items['show'] = 'headings'
544         records_items.column('#0', width=100, minwidth=30)
545         records_items.column('#1', width=205, minwidth=30)
546         records_items.column('#2', width=205, minwidth=30)
547         records_items.column('#3', width=200, minwidth=30)
548         records_items.column('#4', width=100, minwidth=30)
549         records_items.column('#5', width=200, minwidth=30)
550         records_items.pack(fill=BOTH, expand=YES, side=BOTTOM)
551 
552         insert_items()
553         records_items.bind('<ButtonRelease-1>', item_info)
554 
555         # records_items.bind('<ButtonRelease-1>',item_info)
556         # records_items.bind('<Return>', item_info)
557 
558 
559 class orders_management():
560     def __init__(self, ifc):
561         self.ifc = ifc
562         ifc.title('Orders Management')
563         ifc.geometry('1600x1000+0+0')
564         ifc.configure(bg='black')
565         ifc.resizable(width=FALSE, height=FALSE)
566 
```

```
566 # ================================================================
    Variables=============================================================
567         entry_sno = StringVar()
568         entry_itemname = StringVar()
569         entry_itemcode = StringVar()
570         entry_quantity = StringVar()
571         entry_price = StringVar()
572         entry_brand = StringVar()
573 
574 
    # ================================================================================
    Functions=============================================================
575 
576         def back_dashboard():
577             mainframefororders.pack_forget()
578             Dashboard(ifc)
579 
580         def frameforreciept():
581             mainframefororders.pack_forget()
582             for widget in mainframefororders.winfo_children():
583                 widget.pack_forget()
584             print_reciept(ifc)
585 
586         def insert_items():
587             cn = sql.connect(host='localhost', user='root', password='danger', database='
    project')
588             c = cn.cursor()
589             c.execute('select * from items order by itemname')
590             rows = c.fetchall()
591 
592             for row in rows:
593                 records_items.insert('', END, values=row)
594                 cn.commit()
595             cn.close()
596 
597         rows_of_order = []
598 
599         def order_item(ev):
600             view_iteminfo = records_items.focus()
601             getiteminfo = records_items.item(view_iteminfo)
602             record = getiteminfo['values']
603             print('record was in this format', record)
604             entry_sno.set(record[0])
605             entry_itemname.set(record[1])
606             entry_itemcode.set(record[2])
607             entry_quantity.set(record[3])
608             entry_price.set(record[4])
609             entry_brand.set(record[5])
```

```
610             ordered_row = ((entry_sno.get()),
611                            (entry_itemname.get()),
612                            (entry_itemcode.get()),
613                            (entry_quantity.get()),
614                            (entry_price.get()),
615                            (entry_brand.get()))
616
617             rows_of_order.append(ordered_row)
618             ordered_items.insert('', END, values=ordered_row)
619
620         def ordereditem_remove():
621             # view_iteminfo=records_items.focus()
622
623             # ordered_items.delete(getiteminfo['values'])
624             selected_item = ordered_items.selection()
625             print(selected_item)
626             getiteminfo = ordered_items.item(selected_item)['values']
627             ordered_items.delete(selected_item)
628             print(rows_of_order)
629
630             getiteminfo = (
631                 str(getiteminfo[0]), getiteminfo[1], str(getiteminfo[2]), str(getiteminfo[3]), str
    (getiteminfo[4]),
632                 getiteminfo[5])
633             print(type(getiteminfo))
634
635             print('format igven to remove is', tuple(getiteminfo))
636
637             rows_of_order.remove(tuple(getiteminfo))
638             print('after removing', rows_of_order)
639
640         def ordereditems_clear():
641             # for orderedrow in rows_of_order:
642             # rows_of_order.remove(orderedrow)
643             rows_of_order.clear()
644             ordered_items.delete(*ordered_items.get_children())
645
646         def placeorder():
647             cn = sql.connect(host='localhost', user='root', password='danger', database='
    orders')
648             c = cn.cursor()
649             c.execute('use orders')
650             c.execute('show tables in orders')
651             c.fetchall()
652             c.execute(
653                 'SELECT count(*) AS TOTALNUMBEROFTABLES FROM
    INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA = \'orders\' and TABLE_NAME
     like ' + '\'%' + '_dated_' + str(
```

```
654                     datewithouthyphens) + '%\'')
655             ordertillnow = int(str(c.fetchall()).replace('[(', '').replace(',)]', ''))
656             # print(ordertillnow)
657             global orderno
658             orderno = ordertillnow + 1
659             print(orderno)
660             c.execute('create table order' + str(orderno) + '_dated_' + str(
661                 datewithouthyphens) + '(itemno int auto_increment,sno int,itemname varchar(45
    ),itemcode varchar(45),quantity varchar(45),price int,company varchar(45),unique(itemno))')
662             for row in rows_of_order:
663                 query_insertorder = ('insert into Order' + str(
664                     orderno) + '_dated_' + str(
665                     datewithouthyphens) + '(sno,itemname,itemcode,quantity,price,company)
    values (%s,%s,%s,%s,%s,%s)')
666                 ordered_rowintomysql = (str(row[0]), str(row[1]), str(row[2]), str(row[3]), str(
    row[4]), str(row[5]))
667                 c.execute(query_insertorder, ordered_rowintomysql)
668                 cn.commit()
669             cn.close()
670             print('Ooder' + str(orderno) + '_dated_' + str(datewithouthyphens) + 'Placed
    Successfully')
671             ordereditems_clear()
672             frameforreciept()
673
674
    # ================================================================================
    Frames=========================================================
675         global mainframefororders
676         mainframefororders = Frame(ifc, bg='gold', bd=5, relief=RIDGE)
677         mainframefororders.pack(fill=BOTH, expand=YES)
678         Topframe = Frame(mainframefororders, width=1600, height=900, bg='red', bd=3,
    relief=RIDGE)
679         Topframe.pack(side=TOP, fill=BOTH, expand=YES)  # grid(row=0,column=0,sticky=
    NSEW)
680         Bottomframe = Frame(mainframefororders, width=1600, height=100, bg='blue', bd=3
    , relief=RIDGE)
681         Bottomframe.pack(side=BOTTOM, fill=BOTH, expand=YES)  # grid(row=1,column=0,
    sticky='nsew')
682         Bottommostframe = Frame(Bottomframe, width=1600, height=100, bg='silver', bd=3
    , relief=RIDGE)
683         Bottommostframe.pack(fill=BOTH, expand=YES)
684         leftframe = Frame(Topframe, bg='white', width=800, height=900, bd=10, relief=
    RIDGE)
685         leftframe.pack(side=LEFT, fill=BOTH, expand=YES)
686         recordsframe = Frame(leftframe, width=800, height=900, bd=10, relief=RIDGE)
687         recordsframe.pack(fill=BOTH, expand=YES)
688         rightframe = Frame(Topframe, bg='silver', width=800, height=900, bd=10, relief=
    RIDGE)
```

```
689         rightframe.pack(side=RIGHT, fill=BOTH, expand=YES)
690         orderframe = Frame(rightframe, width=800, height=900, bd=8, relief=RIDGE)
691         orderframe.pack(fill=BOTH, expand=YES)
692         cartframe = Frame(orderframe, width=800, height=800, bd=10, relief=RIDGE)
693         cartframe.grid(row=1, column=0, columnspan=3, sticky=NSEW)
694 
695         # Topframe.grid_columnconfigure(0, weight=1, uniform="group1")
696         # Topframe.grid_columnconfigure(1, weight=1, uniform="group1")
697         # Topframe.grid_rowconfigure(0,weight=1)
698 
    # ================================================================
    General Buttons and Labels
    ================================================================
699         button_back = Button(Bottommostframe, text='Back', font='comicsansms 18 bold', bd
    =5, command=back_dashboard)
700         button_back.pack(side=LEFT, fill=BOTH, expand=YES)
701         button_exit = Button(Bottommostframe, text='Exit', bg='pink', font='comicsansms 18
     bold', bd=8,
702                              command=mainexit)
703         button_exit.pack(side=RIGHT, fill=BOTH, expand=YES)
704         label_selectrecords = Label(recordsframe, text='Double Click To Add Items To Cart'
    , font='comicsansms 18 bold',
705                                     bd=5)
706         label_selectrecords.pack(fill=X)
707         label_selectedrecords = Label(orderframe, text='Items Added  To Cart', font='
    comicsansms 18 bold', bd=5)
708         label_selectedrecords.grid(row=0, column=0, columnspan=3)
709         # ================================Buttons for orderframe
    ================================================================
710         button_placeorder = Button(orderframe, text='Place Order', font='comicsansms 18
    bold', bd=5, padx=70,
711                                    command=placeorder)
712         button_clear = Button(orderframe, text='Remove All', font='comicsansms 18 bold',
    bd=5, padx=30,
713                               command=ordereditems_clear)
714         button_remove = Button(orderframe, text='Remove Item', font='comicsansms 18 bold
    ', bd=5, padx=30,
715                                command=ordereditem_remove)
716         button_placeorder.grid(row=2, column=1, sticky=NS)
717         button_clear.grid(row=2, column=2, sticky=NS)
718         button_remove.grid(row=2, column=0, sticky=NS)
719 
    # ================================================================
    Treeviews
    ================================================================
720         records_items = ttk.Treeview(recordsframe, height=35,
721                                      columns=('Sno', 'Item Name', 'Item Code', 'Quantity', '
    Price', 'Brand'))
```

```
722         sty = ttk.Style()
723         sty.configure('Treeview', rowheight=20)
724         scroll_x = ttk.Scrollbar(recordsframe, orient=HORIZONTAL, command=records_items
    .xview)
725         scroll_y = ttk.Scrollbar(recordsframe, orient=VERTICAL, command=records_items.
    yview)
726         records_items.configure(yscrollcommand=scroll_y.set, xscrollcommand=scroll_x.set)
727         scroll_x.pack(side=BOTTOM, fill=X)
728         scroll_y.pack(side=RIGHT, fill=Y)
729         records_items.heading('Sno', text='Sno')
730         records_items.heading('Item Name', text='Item Name')
731         records_items.heading('Item Code', text='Item Code')
732         records_items.heading('Quantity', text='Quantity')
733         records_items.heading('Price', text='Price')
734         records_items.heading('Brand', text='Brand')
735         records_items['show'] = 'headings'
736         records_items.column('#0', width=100, minwidth=30)
737         records_items.column('#1', width=105, minwidth=30)
738         records_items.column('#2', width=105, minwidth=30)
739         records_items.column('#3', width=100, minwidth=30)
740         records_items.column('#4', width=100, minwidth=30)
741         records_items.column('#5', width=100, minwidth=30)
742         insert_items()
743         records_items.pack(fill=BOTH, expand=YES, side=LEFT)
744         records_items.bind('<Double-1>', order_item)
745         # ==============================Ordered items treeview
    ================================================================
746         global ordered_items
747         ordered_items = ttk.Treeview(cartframe, height=35,
748                                      columns=('Sno', 'Item Name', 'Item Code', 'Quantity', '
    Price', 'Brand'))
749         sty = ttk.Style()
750         sty.configure('Treeview', rowheight=20)
751         scroll_x = ttk.Scrollbar(cartframe, orient=HORIZONTAL, command=ordered_items.
    xview)
752         scroll_y = ttk.Scrollbar(cartframe, orient=VERTICAL, command=ordered_items.yview)
753         ordered_items.configure(yscrollcommand=scroll_y.set, xscrollcommand=scroll_x.set)
754         scroll_x.pack(side=BOTTOM, fill=X, expand=YES)
755         scroll_y.pack(side=RIGHT, fill=Y, expand=YES)
756         ordered_items.heading('Sno', text='Sno')
757         ordered_items.heading('Item Name', text='Item Name')
758         ordered_items.heading('Item Code', text='Item Code')
759         ordered_items.heading('Quantity', text='Quantity')
760         ordered_items.heading('Price', text='Price')
761         ordered_items.heading('Brand', text='Brand')
762         ordered_items['show'] = 'headings'
763         ordered_items.column('#0', width=100, minwidth=30)
764         ordered_items.column('#1', width=105, minwidth=30)
```

```
765         ordered_items.column('#2', width=105, minwidth=30)
766         ordered_items.column('#3', width=100, minwidth=30)
767         ordered_items.column('#4', width=100, minwidth=30)
768         ordered_items.column('#5', width=100, minwidth=30)
769         ordered_items.pack(side=LEFT, fill=BOTH, expand=YES)
770
771
772 class print_reciept():
773     def __init__(self, ifc):
774         self.ifc = ifc
775         ifc.geometry('600x900+0+0')
776         ifc.title('Order Finalization')
777         def backtoorders():
778             mainframeforreciept.pack_forget()
779             for widget in mainframeforreciept.winfo_children():
780                 widget.pack_forget()
781             orders_management(ifc)
782         def saveaspdf():
783             Time = str(datetime.now())[11:]
784             cn = sql.connect(host='localhost', user='root', password='danger', database='
    orders')
785             c = cn.cursor()
786             c.execute('use orders')
787             c.execute('select * from order' + str(orderno) + '_dated_' + str(
    datewithouthyphens))
788             data = [['Itemno', 'Sno', 'Itemname', 'Itemcode', 'Quantity', 'Price', 'Company'
    ]]
789             for row in c.fetchall():
790                 data.append(row)
791             print(data)
792             global filename
793             currentworkingdirectory = os.getcwd()
794             recieptsfolder = currentworkingdirectory + '\Order_Reciepts'
795             todaysfolder = recieptsfolder + '\Reciepts_dated_' + str(datewithouthyphens)
796             if os.path.exists(recieptsfolder) == True:
797                 pass
798             else:
799                 os.mkdir(recieptsfolder)
800             if os.path.exists(todaysfolder) == True:
801                 pass
802             else:
803                 os.mkdir(todaysfolder)
804
805             filename = todaysfolder + '\Reciept_of_order' + str(orderno) + '_dated_' + str(
    datewithouthyphens)+ '.pdf'
806             print(filename)
807             pdf = SimpleDocTemplate(filename, pagesize=A4)
808             width, height = A4
```

```
809             # SubTables
810             headingTable = Table([['AMBEDHKAR GROCERY STORE'], ['Peerzadiguda,Uppal-
    500039'] , ['Phone:9876543210'], ['-' * 177],['Date: ' + str(todaysdate) + '        Order
    Invoice          Time:' + Time],['-' * 177],
811                                   ['Customer Name:' + str(entry_customername.get()) +
    '          Customer Phone:' + str(entry_customerphone.get())]], width)
812             headingTableStyle = TableStyle([('ALIGN', (0, 0), (-1, -1), 'CENTER'),('
    FONTNAME', (0, 0), (-1, -1), 'Helvetica-Bold')
813                                             ,
814                                           ('BACKGROUND', (0, 0), (-1, -1), colors.green), ('
    FONTSIZE', (0, 0), (-1, 0), 19),('TOPPADDING', (0, 0), (-1, 0), 5),('BOTTOMPADDING'
    , (0, 0), (-1, 0), 8)
815
816                                             , ('TOPPADDING', (0, 1), (-1, 1), 2),('
    BOTTOMPADDING', (0, 1), (-1, 1), 2),('FONTSIZE', (0, 1), (-1, 2), 12),
817
818                                           ('BOTTOMPADDING', (0, 2), (-1, 2), 0),('
    TOPPADDING', (0, 2), (-1, 2), 1),
819
820                                           ('BOTTOMPADDING', (0, 3), (-1, 3), 0),('
    TOPPADDING', (0, 3), (-1, 3), 0),
821
822                                           ('FONTSIZE', (0, 4), (-1, 4), 16),('
    BOTTOMPADDING', (0, 4), (-1, 4), 0),('TOPPADDING', (0, 4), (-1, 4), 0),
823
824                                           ('FONTSIZE', (0, 6), (-1, 6), 14), ('
    BOTTOMPADDING', (0, 6), (-1, 6), 5)])
825             headingTable.setStyle(headingTableStyle)
826             order_details_table = Table(data)
827             order_details_tableStyle = TableStyle([('ALIGN', (0, 0), (-1, -1), 'CENTER'),
828                                                    ('FONTNAME', (0, 0), (-1, -1), 'Helvetica-
    Bold'),
829                                                    ('FONTSIZE', (0, 0), (-1, 0), 14),
830                                                    ('BACKGROUND', (0, 0), (-1, 0), colors.brown),
831                                                    ('BOTTOMPADDING', (0, 0), (-1, 0), 10),
832                                                    ('LEFTPADDING', (0, 0), (-1, -1), 16),
833                                                    ('RIGHTPADDING', (0, 0), (-1, -1), 16),
834                                                    ('GRID', (0, 0), (-1, -1), 2, colors.red),
835                                                    ('BOTTOMPADDING', (0, 1), (-1, -1), 5),
836                                                    ('FONTSIZE', (0, 1), (-1, -1), 12),
837                                                    ('BACKGROUND', (0, 1), (-1, -1), colors.gold),
838                                                    ])
839             order_details_table.setStyle(order_details_tableStyle)
840             c.execute('select sum(price) from order' + str(orderno) + '_dated_' + str(
    datewithouthyphens))
841             # print(c.fetchall())
842             sum = (str(c.fetchall()).replace('[(Decimal(\'', '').replace('\'),)]', ''))
843             print(type(sum))
```

```
844             print(sum)
845             cn.commit()
846             cn.close()
847             totalTable = Table([['-' * 177], ['Total Cost : Rupees ' + sum], ['-' * 177], ['
    Visit Again']])
848             totalTableStyle = TableStyle([('ALIGN', (0, 0), (-1, -1), 'CENTER'),
849                                           ('FONTNAME', (0, 0), (-1, -1), 'Helvetica-Bold')])
850
851             totalTable.setStyle(totalTableStyle)
852
853             mainTable = Table([[headingTable], [order_details_table], [totalTable]])  # ,[
    orderTable],[order1Table]
854             mainTableStyle = TableStyle([('ALIGN', (0, 0), (-1, -1), 'CENTER'),
855                                          ('FONTNAME', (0, 0), (-1, -1), 'Helvetica-Bold'),
856                                          ('FONTSIZE', (0, 0), (-1, 0), 14),
857                                          ('GRID', (0, 0), (-1, -1), 2, colors.black)
858                                          ])
859
860             mainTable.setStyle(mainTableStyle)
861
862             elems = []
863             elems.append(mainTable)
864             pdf.build(elems)
865             os.startfile(filename)
866             messagebox._show(message='Opening Generated Reciept Which was saved at \n'
     + str(filename))
867         def printpdf():
868             PDFNet.Initialize()
869             doc = PDFDoc(filename)
870             doc.InitSecurityHandler()
871             # Set our PrinterMode options
872             printerMode = PrinterMode()
873             printerMode.SetCollation(True)
874             printerMode.SetCopyCount(1)
875             printerMode.SetDPI(100);  # regardless of ordering, an explicit DPI setting
    overrides the OutputQuality setting
876             printerMode.SetDuplexing(PrinterMode.e_Duplex_Auto)
877             # If the XPS print path is being used, then the printer spooler file will
878             # ignore the grayscale option and be in full color
879             printerMode.SetOutputColor(PrinterMode.e_OutputColor_Grayscale)
880             printerMode.SetOutputQuality(PrinterMode.e_OutputQuality_Medium)
881             # printerMode.SetNUp(2,1)
882             # printerMode.SetScaleType(PrinterMode.e_ScaleType_FitToOutPage)
883             # Print the PDF document to the default printer, using "tiger.pdf" as the
    document
884             # name, send the file to the printer not to an output file, print all pages, set the
     printerMode
885             # and don't provide a cancel flag.
```

```
886             Print.StartPrintJob(doc, "", doc.GetFileName(), "", None, printerMode, None)
887
888         global mainframeforreciept
889         mainframeforreciept = Frame(ifc, bg='gold', bd=5, relief=RIDGE)
890         mainframeforreciept.pack(fill=BOTH, expand=YES)
891         buttonsframe = Frame(mainframeforreciept, bg='white', width=400, height=900, bd=
    10, relief=RIDGE)
892         buttonsframe.pack(fill=BOTH, expand=YES)
893         label_customername = Label(buttonsframe, text='Enter Customername', font='
    sansberi 20 bold', bg='black',
894                                    fg='violet',
895                                    padx=50, pady=20)
896         label_customerphone = Label(buttonsframe, text='Enter Customer PhoneNumber', font
    ='sansberi 20 bold',
897                                     bg='black',
898                                     fg='violet', pady=20)
899         entry_customername = Entry(buttonsframe, fg='green', bg='gold', font='sansberi 20
    bold', width=30,
900                                    borderwidth=5)
901         entry_customerphone = Entry(buttonsframe, fg='green', font='sansberi 20 bold', bg=
    'gold', width=30,
902                                     borderwidth=10)
903         label_customername.grid(row=0, column=0, columnspan=2)
904         entry_customername.grid(row=1, column=0, columnspan=2)
905         label_customerphone.grid(row=2, column=0, columnspan=2)
906         entry_customerphone.grid(row=3, column=0, columnspan=2)
907         label_space = Label(buttonsframe, text='-' * 150, font='sansberi 6 bold')
908         label_space.grid(row=4, column=0, columnspan=2)
909         button_saveaspdf = Button(buttonsframe, text='Generate Reciept as pdf and View',
    fg='green', bg='gold',
910                                   font='sansberi 20 bold', borderwidth=5, padx=20, command=
    saveaspdf)
911         button_saveaspdf.grid(row=6, column=0, columnspan=2)
912         button_printpdf = Button(buttonsframe, text='Print Reciept as pdf', fg='green', bg='
    gold',
913                                  font='sansberi 20 bold', borderwidth=5, padx=20, command=
    printpdf)
914         button_printpdf.grid(row=7, column=0, columnspan=2)
915         label_space = Label(buttonsframe, text='-' * 150, font='sansberi 6 bold')
916         label_space.grid(row=8, column=0, columnspan=2)
917         button_finaliseorder = Button(buttonsframe, text='Finalise Order', fg='green', bg='
    gold',
918                                       font='sansberi 20 bold', borderwidth=5)
919         button_finaliseorder.grid(row=9, column=0, columnspan=2)
920         label_space = Label(buttonsframe, text='-' * 150, font='sansberi 6 bold')
921         label_space.grid(row=10, column=0, columnspan=2)
922         button_exit = Button(buttonsframe, image=EXIT, padx=80, pady=100, command=
    mainexit)
```

```
923         button_exit.grid(row=11, column=0)
924         button_back = Button(buttonsframe, text='Back', bg='Black', fg='Red', padx=90,
    pady=24, command=backtoorders)
925         button_back.grid(row=11, column=1)
926 class changepasswordclass():
927     def __init__(self,ifc):
928         self.ifc=ifc
929         ifc.geometry('800x600')
930         ifc.title('Change Password')
931         def back3():
932             passwordchangeframe.pack_forget()
933             for widget in passwordchangeframe.winfo_children():
934                 widget.pack_forget()
935             Dashboard(ifc)
936         def changepassword():
937             i=0
938 
939             for i in range(0,count):
940                 cn = sql.connect(host='localhost', user='root', password='danger', database='
    project')
941                 c = cn.cursor()
942                 c.execute('select username,usertype,password from credentials')
943                 userdata = []
944                 for row in c.fetchall():
945                     userdata.append(row)
946                 credentials = pd.DataFrame(data=userdata, columns=['username', 'usertype', '
    password'])
947                 cn.close()
948                 if credentials.loc[i,'username']==str(verify_username.get()):
949                     if credentials.loc[i,'password']==str(verify_password.get()):
950                         cn = sql.connect(host='localhost', user='root', password='danger',
    database='project')
951                         c = cn.cursor()
952                         query_changepassword='update credentials set password=%s where
    username=%s'
953                         tuple_changepassword=(str(entry_newpassword.get()),str(
    verify_username.get()))
954                         c.execute(query_changepassword,tuple_changepassword)
955                         cn.commit()
956                         cn.close()
957                         messagebox._show(message='Password Change was Successful')
958                         entry_newpassword.delete(0,END)
959                         verify_username.delete(0,END)
960                         verify_password.delete(0,END)
961                         break
962                     else:
963                         i+=1
964                 else:
```

```
965                     i+=1
966                 if i == count:
967                     messagebox.showerror(message='Wrong Credentials Entered Verification
    Failed')
968                     entry_newpassword.delete(0, END)
969                     verify_username.delete(0, END)
970                     verify_password.delete(0, END)
971         passwordchangeframe=Frame(ifc,width=800,height=600)
972         passwordchangeframe.pack(fill=BOTH,expand=YES)
973         label_changepassword = Label(passwordchangeframe, text='Verify username and
    existing password',font='sansberi 28 bold', bg='black', fg='violet', bd=5, relief='sunken',
    padx=150, pady=8)
974         label_changepassword.pack()
975         global verify_username
976         global verify_password
977         verify_username = Entry(passwordchangeframe, fg='red', bg='pink', width=50,
    borderwidth=5)
978         verify_password = Entry(passwordchangeframe, fg='red', bg='pink', width=50,
    borderwidth=5)
979         global entry_newpassword
980         entry_newpassword = Entry(passwordchangeframe, fg='red', bg='pink', width=50,
    borderwidth=5)
981         label_username = Label(passwordchangeframe, text='Enter Username', font='
    sansberi 20 bold', bg='black', fg='violet')
982         label_password = Label(passwordchangeframe, text='Enter Existing  Password', font
    ='sansberi 20 bold', bg='black',
983                                fg='violet')
984         label_newpassword = Label(passwordchangeframe, text='Enter New Password', font=
    'sansberi 20 bold', bg='black',
985                                   fg='violet')
986         button_change = Button(passwordchangeframe, image=img_changepassword, command
    =changepassword)
987         button_back3 = Button(passwordchangeframe, image=img_back, border=0, command=
    back3)
988         button_exit = Button(passwordchangeframe, image=EXIT, padx=80, pady=10, border
    =0, command=mainexit)
989         label_username.pack()
990         verify_username.pack()
991         label_password.pack()
992         verify_password.pack()
993         label_newpassword.pack()
994         entry_newpassword.pack()
995         button_change.pack()
996         button_back3.pack()
997         button_exit.pack()
998 class usermanagementclass():
999     def __init__(self,ifc):
1000         self.ifc=ifc
```

```
1001        ifc.title('User Management System')
1002        ifc.geometry('1400x800')
1003        ifc.configure(bg='black')
1004        usrmgt_username=StringVar()
1005        usrmgt_password=StringVar()
1006        usrmgt_usertype=StringVar()
1007        usrmgt_sno=StringVar()
1008        def back4():
1009            usermanagementframe.pack_forget()
1010            for widget in usermanagementframe.winfo_children():
1011                widget.pack_forget()
1012            Dashboard(ifc)
1013        def add_user():
1014            cn = sql.connect(host='localhost', user='root', password='danger', database='
    project')
1015            c = cn.cursor()
1016            insertquery = 'insert into credentials values (%s,%s,%s,%s)'
1017            # c.execute(+str(count+1)+','+str(entry_username.get())+','+str(entry_usertype.
    get()+','+str(entry_password.get())+')'))
1018            c.execute(insertquery,(str(count + 1),str(usrmgt_username.get()),str(
    usrmgt_usertype.get()),str(usrmgt_password.get())))
1019            cn.commit()
1020            cn.close()
1021            records_users.delete(*records_users.get_children())
1022            insert_users()
1023            messagebox._show(message='User Added Successfully')
1024        def delete_user():
1025            cn = sql.connect(host='localhost', user='root', password='danger', database='
    project')
1026            c = cn.cursor()
1027            c.execute('delete from credentials where username=' + '\'' + str(
    entry_username.get()) + '\'')
1028            cn.commit()
1029            cn.close()
1030            records_users.delete(*records_users.get_children())
1031            insert_users()
1032            messagebox._show(message='User Deleted Successfully')
1033        def update_user():
1034            cn = sql.connect(host='localhost', user='root', password='danger', database='
    project')
1035            c = cn.cursor()
1036            tuple_updateuser = (usrmgt_password.get(),usrmgt_username.get(),
    usrmgt_usertype.get(),usrmgt_sno.get())
1037            query_updateuser='update credentials set password=%s,username=%s,usertype=%
    s where sno=%s'
1038            c.execute(query_updateuser,tuple_updateuser)
1039            cn.commit()
1040            cn.close()
```

```
1041            records_users.delete(*records_users.get_children())
1042            insert_users()
1043            messagebox._show(message='User Credentials updated successfully')
1044        def insert_users():
1045            cn = sql.connect(host='localhost', user='root', password='danger', database='
    project')
1046            c = cn.cursor()
1047            c.execute('select * from credentials')
1048            userlist=c.fetchall()
1049            cn.commit()
1050            cn.close()
1051            for userdetail in userlist:
1052                records_users.insert('',END,values=userdetail)
1053        def usermgt_clear():
1054            entry_sno.delete(0,END)
1055            entry_username.delete(0,END)
1056            entry_password.delete(0,END)
1057            entry_usertype.delete(0,END)
1058        def userinfo(ev):
1059            view_userinfo = records_users.focus()
1060            getuserinfo = records_users.item(view_userinfo)
1061            record = getuserinfo['values']
1062            usrmgt_sno.set(str(record[0]))
1063            usrmgt_username.set(record[1])
1064            usrmgt_usertype.set(record[2])
1065            usrmgt_password.set(record[3])
1066        usermanagementframe=Frame(ifc,bd=10,bg='yellow')
1067        usermanagementframe.pack(fill=BOTH,expand=YES)
1068        usrmgtleftframe=Frame(usermanagementframe,bd=10,width=400,height=800,bg='blue
    ')
1069        usrmgtleftframe.pack(fill=BOTH,expand=YES,side=LEFT)
1070        usrmgtrightframe=Frame(usermanagementframe,width=600,height=800,bg='green',bd
    =10)
1071        usrmgtrightframe.pack(fill=BOTH,expand=YES,side=RIGHT)
1072        instruction='''Double Click To view User Details in Fields'''
1073        label_instructions=Label(usrmgtleftframe,text=instruction,font='rockwell 14 bold')
1074        label_instructions.grid(row=0,column=0,columnspan=2)
1075        lbl_sno = Label(usrmgtleftframe, text='Sno', font='segoeuiblack 14 bold', bd=10)
1076        lbl_sno.grid(row=1, column=0, padx=18, pady=18)
1077        entry_sno = Entry(usrmgtleftframe, bd=10, width=16, font='rockwell 14',
    textvariable=usrmgt_sno)
1078        entry_sno.grid(row=1, column=1, padx=18, pady=18)
1079        lbl_username = Label(usrmgtleftframe, text='user Name', font='segoeuiblack 14
    bold', bd=10)
1080        lbl_username.grid(row=2, column=0, padx=18, pady=18)
1081        entry_username = Entry(usrmgtleftframe, bd=10, width=16, font='rockwell 14 bold',
    textvariable=usrmgt_username)
1082        entry_username.grid(row=2, column=1, padx=18, pady=18)
```

```
1083        lbl_password = Label(usrmgtleftframe, text='Password', font='segoeuiblack 14 bold'
    , bd=10)
1084        lbl_password.grid(row=3, column=0, padx=18, pady=18)
1085        entry_password = Entry(usrmgtleftframe, bd=10, width=16, font='rockwell 14',
    textvariable=usrmgt_password)
1086        entry_password.grid(row=3, column=1, padx=18, pady=18)
1087        entry_usertype = ttk.Combobox(usrmgtleftframe, width=16, font='rockwell 14',
    textvariable=usrmgt_usertype)
1088        entry_usertype['values'] = ('owner','employee')
1089        entry_usertype.grid(row=4, column=1, padx=18, pady=18)
1090        lbl_usertype = Label(usrmgtleftframe, text='usertype', font='segoeuiblack 14 bold'
    , bd=10)
1091        lbl_usertype.grid(row=4, column=0, padx=18, pady=18)
1092        button_adduser = Button(usrmgtleftframe, text='Add User', bg='pink', font='
    segoeuiblack 14 bold italic', bd=8, command=add_user)
1093        button_adduser.grid(row=5, column=0)
1094        button_deleteuser = Button(usrmgtleftframe, text='Delete User', bg='pink', font='
    segoeuiblack 14 bold italic', bd=8, command=delete_user)
1095        button_deleteuser.grid(row=5, column=1)
1096        button_updateuser = Button(usrmgtleftframe, text='Update User', bg='pink', font='
    segoeuiblack 14 bold italic', bd=8, command=update_user)
1097        button_updateuser.grid(row=6, column=0)
1098        button_clearuser = Button(usrmgtleftframe, text='Clear ', bg='pink', font='
    segoeuiblack 14 bold italic', bd=8,command=usermgt_clear)
1099        button_clearuser.grid(row=6, column=1)
1100        button_exit = Button(usrmgtleftframe, image=EXIT, padx=40, pady=40, command=
    mainexit)
1101        button_exit.grid(row=7, column=0)
1102        button_back = Button(usrmgtleftframe, text='Back',font='segoeuiblack 14 bold italic
    ', bg='Black', fg='Red',padx=60, pady=20, command=back4)
1103        button_back.grid(row=7, column=1)
1104        records_users = ttk.Treeview(usrmgtrightframe, height=35,columns=('Sno','
    Username','UserType','Password'))
1105        sty = ttk.Style()
1106        sty.configure('Treeview', rowheight=20)
1107        scroll_x = ttk.Scrollbar(usrmgtrightframe, orient=HORIZONTAL, command=
    records_users.xview)
1108        scroll_y = ttk.Scrollbar(usrmgtrightframe, orient=VERTICAL, command=
    records_users.yview)
1109        records_users.configure(yscrollcommand=scroll_y.set, xscrollcommand=scroll_x.set)
1110        scroll_x.pack(side=BOTTOM, fill=X)
1111        scroll_y.pack(side=RIGHT, fill=Y)
1112        records_users.heading('Sno', text='Sno')
1113        records_users.heading('Username', text='User Name')
1114        records_users.heading('UserType', text='User Type')
1115        records_users.heading('Password', text='Password')
1116        records_users['show'] = 'headings'
1117        records_users.column('#0', width=100, minwidth=50)
```

```
1118        records_users.column('#1', width=150, minwidth=80)
1119        records_users.column('#2', width=150, minwidth=80)
1120        records_users.column('#3', width=150, minwidth=80)
1121        insert_users()
1122        records_users.pack(fill=BOTH, expand=YES,side=TOP)
1123        records_users.bind('<Double-1>',userinfo)
1124
1125 class Dashboard():
1126     def __init__(self, ifc, *pargs):
1127         self.ifc = ifc
1128         ifc.title('Shopkeeper Dashboard')
1129         ifc.geometry('858x600')
1130         ifc.resizable(width=FALSE, height=FALSE)
1131
1132         def command_orderhistory():
1133             keeperdashboard.pack_forget()
1134             for widget in keeperdashboard.winfo_children():
1135                 widget.pack_forget()
1136             orderhistory(ifc)
1137
1138         def back2():
1139             keeperdashboard.pack_forget()
1140             entry_username.delete(0, END)
1141             entry_password.delete(0, END)
1142             shopkeeperclass(ifc)
1143
1144         def framefordataanalysis():
1145             keeperdashboard.pack_forget()
1146             for widget in keeperdashboard.winfo_children():
1147                 widget.pack_forget()
1148             analysedata(ifc)
1149         def changepassword():
1150             keeperdashboard.pack_forget()
1151             for x in keeperdashboard.winfo_children():
1152                 x.place_forget()
1153             changepasswordclass(ifc)
1154         def command_itemsmgt():
1155             # keeperdashboard.pack_forget()
1156             for widget in keeperdashboard.winfo_children():
1157                 widget.place_forget()
1158             items_management(ifc)
1159         def command_order():
1160             for widget in keeperdashboard.winfo_children():
1161                 widget.grid_forget()
1162             keeperdashboard.pack_forget()
1163             orders_management(ifc)
1164         def command_usermanagement():
1165             for widget in keeperdashboard.winfo_children():
```

```
                widget.grid_forget()
            keeperdashboard.pack_forget()
            usermanagementclass(ifc)
        global keeperdashboard
        keeperdashboard = Frame(ifc, bg='black')
        keeperdashboard.pack(fill=BOTH, expand=YES)
        button_back2 = Button(keeperdashboard, image=img_logout, bd=0, command=back2)
        button_back2.grid(row=0, column=0)
        button_exit = Button(keeperdashboard, image=EXIT, padx=80, pady=10, border=0, command=mainexit)
        button_exit.grid(row=0, column=3, sticky=NE)
        button_change = Button(keeperdashboard, image=img_changepassword, command=changepassword, bd=0)
        button_change.grid(row=0, column=2)
        button_orderitems = Button(keeperdashboard, image=img_orderitems, bd=0, command=command_order)
        button_orderitems.grid(row=1, column=2)
        button_manageitems = Button(keeperdashboard, image=img_viewandmanageitems, bd=0, command=command_itemsmgt)
        button_manageitems.grid(row=2, column=2)
        button_analyzedata = Button(keeperdashboard, image=img_analyzedata, bd=0, command=framefordataanalysis)
        button_analyzedata.grid(row=4, column=2)
        button_orderhistory= Button(keeperdashboard,text=' View Order History', bd=0, command=command_orderhistory,padx=60,pady=15,bg='green',fg='purple')
        button_orderhistory.grid(row=3, column=2)
        for x in range(0,len(usercredentials['username'])):
            if usercredentials.loc[x,'username']==str(user_name):
                if usercredentials.loc[x,'password']==str(pass_word):
                    if usercredentials.loc[x,'usertype']=='owner':
                        global button_manageusers
                        button_manageusers = Button(keeperdashboard, image=img_manageusers, font='rockwell 20 bold', bd=0, bg='gold',fg='violet',command=command_usermanagement)
                        button_manageusers.grid(row=5, column=2)
                    else:
                        x+=1
                else:
                    x+=1

class shopkeeperclass():
    def __init__(self,ifc):
        self.ifc=ifc
        ifc.title('ShopKeeper Login')

        def back1():
            shopmgt.pack_forget()
            entry_username.delete(0, END)
```

```
1206            entry_password.delete(0, END)
1207            homepage(ifc)
1208        def submit():
1209            global user_name
1210            global pass_word
1211            user_name = str(entry_username.get())
1212            pass_word = str(entry_password.get())
1213            if len(user_name) == 0:
1214                if len(pass_word) == 0:
1215                    label_error = Label(shopmgt, text='Please Enter Credentials', font='
    sansberi 18 bold', bg='violet',
1216                                        fg='pink')
1217                    label_error.pack()
1218            else:
1219                cn = sql.connect(host='localhost', user='root', password='danger', database='
    project')
1220                c = cn.cursor()
1221                c.execute('select username,usertype,password from credentials')
1222                data = []
1223                for row in c.fetchall():
1224                    data.append(row)
1225                global usercredentials
1226                usercredentials = pd.DataFrame(data, columns=['username', 'usertype', '
    password'])
1227                display(usercredentials)
1228                global count
1229                count =usercredentials['username'].count()
1230                print(count)
1231                cn.close()
1232                i = 0
1233                for i in range(0, count):
1234                    if usercredentials.loc[i,'username']== user_name:
1235                        if usercredentials.loc[i,'password'] == pass_word:
1236                            shopmgt.pack_forget()
1237                            Dashboard(ifc)
1238                            break
1239                        else:
1240                            i += 1
1241                    else:
1242                        i += 1
1243                if i == count:
1244                    messagebox.showerror(message='Wrong credentials entered')
1245                    entry_username.delete(0, END)
1246                    entry_password.delete(0, END)
1247        global shopmgt
1248        shopmgt = Frame(ifc, bg='black')
1249        shopmgt.pack(fill=BOTH, expand=YES)
1250        global entry_username
```

```
1251        global entry_password
1252        entry_username = Entry(shopmgt, fg='red', bg='pink', width=50, borderwidth=5)
1253        entry_password = Entry(shopmgt, textvariable=password, show='*', fg='red', bg='
    pink', width=50, borderwidth=5)
1254        label_username = Label(shopmgt, text='Enter Username', font='sansberi 20 bold',
    bg='black', fg='violet')
1255        label_password = Label(shopmgt, text='Enter Password', font='sansberi 20 bold', bg
    ='black', fg='violet')
1256        button_exit = Button(shopmgt, image=EXIT, padx=80, pady=10, border=0, command
    =mainexit)
1257        button_submit = Button(shopmgt, image=img_submit, border=0, command=submit)
1258        button_back = Button(shopmgt, image=img_back, border=0, command=back1)
1259        label_username.pack()
1260        entry_username.pack()
1261        label_password.pack()
1262        entry_password.pack()
1263        button_submit.pack()
1264        button_back.pack()
1265        button_exit.pack()
1266 class homepage():
1267     def __init__(self,ifc):
1268         self.ifc=ifc
1269         ifc.geometry('800x600')
1270         ifc.title('Grocery Trade')
1271         ifc.configure(bg='black')
1272         ifc.minsize(0, 0)
1273         ifc.maxsize(2000, 1200)
1274 
1275         def command_login():
1276             Home.pack_forget()
1277             for widget in Home.winfo_children():
1278                 widget.pack_forget()
1279             shopkeeperclass(ifc)
1280 
1281         class Example(Frame):
1282             def __init__(self, master, *pargs):
1283                 Frame.__init__(self, master, *pargs)
1284                 self.image = pilimage.open(os.getcwd()+"\\resources\sum.png")
1285                 self.img_copy = self.image.copy()
1286                 self.background_image = ImageTk.PhotoImage(self.image)
1287                 self.background = Label(self, image=self.background_image)
1288                 self.background.pack(fill=BOTH, expand=YES)
1289                 self.background.bind('<Configure>', self._resize_image)
1290 
1291             def _resize_image(self, event):
1292                 new_width = event.width
1293                 new_height = event.height
1294                 self.image = self.img_copy.resize((new_width, new_height))
```

```
1295                self.background_image = ImageTk.PhotoImage(self.image)
1296                self.background.configure(image=self.background_image)
1297
1298        global Home
1299        Home = Frame(ifc, bg='black')
1300        Home.pack(fill=BOTH, expand=YES)
1301        Head = Label(Home, text='Welcome to D-Mart  Shop Management', font='sansberi
    28 bold', bg='black', fg='violet', bd=5,relief='sunken', padx=150, pady=8)
1302        Head.pack()
1303        shopkeeperlogin = Button(Home, image=LOGIN, border=0, command=command_login)
1304        shopkeeperlogin.pack()
1305        button_exit = Button(Home, image=EXIT, padx=80, pady=10, border=0, command=
    mainexit)
1306        button_exit.pack()
1307        imgfill = Example(Home)
1308        imgfill.pack(fill=BOTH, expand=YES)
1309
1310 ##########################################################################
    ##########Creating eventloop and entering into first frame
    +=========================================================================
    ==============
1311 homepage(ifc)
1312 ifc.mainloop()
1313
1314
```